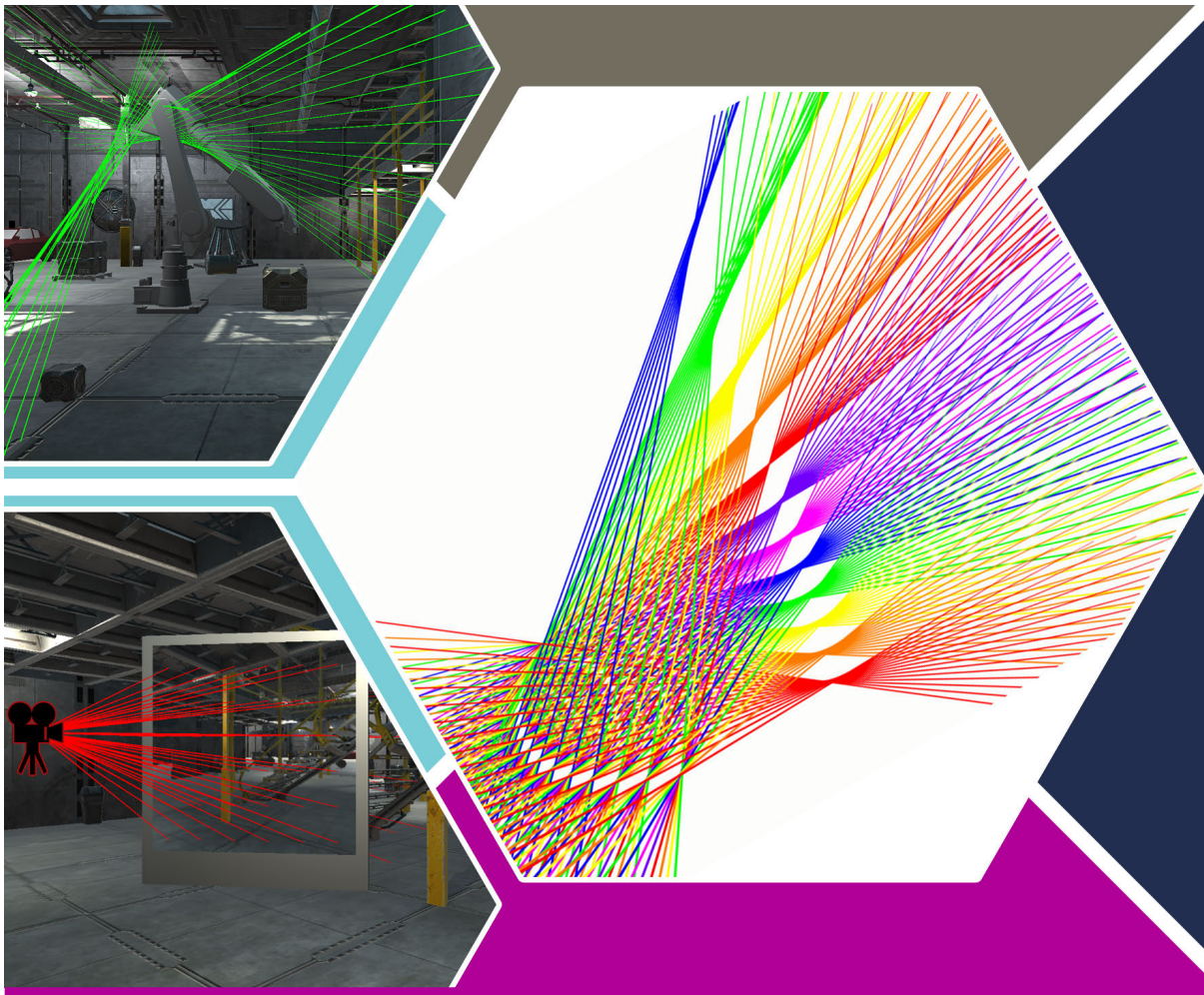
Gaussian Processes for 3D Measurements

On the usage of probabilistic machine learning methods for calibrating 3D measuring devices based on straight lines

Ing. Ivan De Boi



Supervisors **prof. dr. R. Penne — dr. P. Jorissen**

Thesis submitted in fulfilment of the requirements for the degree of doctor in applied engineering
Faculty of Applied Engineering — Antwerpen, 2024

University of Antwerp

Faculty of Applied Engineering

Gaussian Processes for 3D Measurements

On the usage of probabilistic machine learning methods for calibrating 3D measuring devices based on straight lines

Thesis submitted in fulfilment of the requirements for the degree of
doctor in applied engineering
at the University of Antwerp

Ing. Ivan De Boi

Antwerpen, 2024

Supervisors
prof. dr. R. Penne
dr. P. Jorissen

Jury

Chairman
prof. dr. P. Hellinckx, University of Antwerp, Belgium

Supervisors
prof. dr. R. Penne, University of Antwerp, Belgium
dr. P. Jorissen, Karel de Grote Hogeschool, Belgium

Members
prof. dr. W. Daems, University of Antwerp, Belgium
prof. dr. H. Araújo, University of Coimbra, Portugal
prof. dr. C. H. Ek, University of Cambridge, United Kingdom
dr. B. Ribbens, University of Antwerp, Belgium
dr. B. Bogaerts, Apixa, Belgium

Contact
Ing. Ivan De Boi

University of Antwerp
Faculty of Applied Engineering
InViLab Research Group
ivan.deboi@uantwerpen.be
Groenenborgerlaan 171, 2020 Antwerpen, Belgium

Acknowledgments

After over a decade of teaching bachelor students Multimedia Technology and Creative Technologies at the Karel de Grote University College, my professional life needed a new challenge. Getting a PhD has always been a dream of mine. Luckily at the time, I was surrounded by people who were more than willing to support me in this pursuit. I would like to offer my special thanks to all of them.

First, dr. Pieter Jorissen, a co-worker from the Karel de Grote University College and dear friend of mine, who introduced me to the academic world. Together, we developed a virtual reality application to train nurses and midwives called *ImmersiMed*. This became the subject for writing several papers and visiting conferences. I surely learned a lot from him during this period, which gave me a tremendous head start in my PhD. He was one of the first to support my PhD by offering to become one of my promotors. His guidance on a more personal level has truly been indispensable.

Second, prof. dr. Rudi Penne. Weirdly, our lives have crossed paths several times. He was my mathematics professor when I was getting my master's degree. I was a boy scout leader for his son. Years later, we briefly were co-workers at the Karel de Grote University College. We lost contact again when he joined the faculty of applied engineering at the University of Antwerp. A few years later, we started talking again after I emailed him with the question if a PhD was even feasible for me. At that moment in time, I was teaching an introductory course on artificial intelligence. He took this opportunity to introduce me to Gaussian processes. He must have believed in me somehow, because one thing led to another and he became my main promotor. It has been a wonderful experience reasoning with him on a blackboard and coming up with new ways to incorporate Gaussian processes into the world of line geometry. We always had more ideas than time to implement them. His vast expertise as a mathematician, a professor and an author has really boosted my progress in more ways than one.

Third, all the members of the research group InViLab. This is a very dynamic group of young and bright people who are always willing to help each other out, which is a most powerful combination. Special thanks to dr. Bart Ribbens, who introduced me to them and always made sure we all had what we needed.

Fourth, David Van Hoecke, friend and former colleague from the Karel de Grote University College. On multiple occasions, he intervened to enhance the artwork in my presentations, lectures, visualisations, and also the cover image of this thesis.

Fifth, my three very curious children: Boris, Lina and Edda. They share an unquenchable thirst for knowledge, each in their own way. They don't give up when they are trying to understand something. It's very inspiring to observe them like that.

And last but certainly not least, my wife Friederike, who has been the wind beneath my wings for as long as we are together. When I asked her if I could quit my day job to try to get a PhD, she did not have to think long. Even though this meant a considerable sacrifice for our family, she was immediately on board as she knew this was something I needed to do.

Having all these people by my side made things a lot easier for me. For which I'm eternally grateful.

Abstract

In this dissertation we explore the usage of probabilistic machine learning techniques for calibrating 3D measuring devices and applications. More specifically, we focus on Gaussian processes and line geometry. The research focuses on four main topics: galvanometric setup calibration, surface approximation, camera calibration, and the clinical application of mapping visuospatial neglect.

We present a semi-data-driven approach to calibrate galvanometric setups. We perform Gaussian process regression to learn the mapping from the two galvanometric controlled mirrors to the resulting laser lines. Purely data-driven methods bypass the underlying geometric model of the device completely. We re-introduce one geometric assumption: lasers are straight lines. Building upon this approach, we impose constraints on both the inputs and the outputs of the Gaussian process models. The angles of the two galvanometric controlled mirrors can be seen as points on the surface of a torus. The Plücker coordinates of the straight lines themselves obey a quadratic constraint. We formulate several ways to handle these constraints.

We implement Gaussian Process Latent Variable Models, which are a form of non-linear probabilistic dimensionality reduction, to approximate several surfaces. By relaxing the linear assumption of Principal Component Analysis, we can handle many more surfaces than classical methods.

The exploration extends to camera calibration, addressing checkerboard pattern detection through corner detection and sub-pixel refinement. A mapping from virtual perfect checkerboard corners to real image pixels is learned, providing a solution for inferring missing corners and correcting existing ones. This idea is taken further in camera calibration by reversing this mapping. Now we learn from pixels of real checkerboard corners to a perfect virtual checkerboard. This approach turns cameras into perfect pinhole models.

The research concludes by extending its scope to the application of visuospatial neglect assessment and treatment. Active learning methods are applied in a virtual reality environment to lower the number of measurements and thus the burden on patients suffering from this cognitive disorder.

In summary, the dissertation highlights significant contributions to the field of 3D measurements and applications, offering a variety of innovative techniques and practical applications for future exploration.

Samenvatting

In deze dissertatie verkennen we het gebruik van probabilistische machine learning-technieken voor het kalibreren van 3D-meetapparaten en toepassingen. Meer specifiek richten we ons op Gaussian processes en lijngeometrie. Het onderzoek richt zich op vier hoofdonderwerpen: kalibratie van galvanometrische opstellingen, oppervlaktebenadering, camera-kalibratie en de klinische toepassing van het in kaart brengen van visuospatieel neglect.

We presenteren een semi-data-gestuurde aanpak om galvanometrische opstellingen te kalibreren. We voeren Gaussian process regression uit om de mapping van de twee door galvanometers aangestuurde spiegels naar de resulterende laserlijnen te leren. Puur op data gebaseerde methoden omzeilen het onderliggende geometrische model van het apparaat volledig. We herintroduceren één geometrische veronderstelling: lasers zijn rechte lijnen. Op basis van deze aanpak leggen we beperkingen op zowel de invoer als de uitvoer van de Gaussian process-modellen. De rotatiehoeken van de twee door galvanometers aangestuurde spiegels kunnen worden gezien als punten op het oppervlak van een torus. De Plücker-coördinaten van de rechte lijnen zelf voldoen aan een kwadratische beperking. We formuleren verschillende manieren om met deze beperkingen om te gaan.

We implementeren Gaussian Process Latent Variable Models, die een vorm van niet-lineaire probabilistische dimensionale reductie zijn, om verschillende oppervlakken te benaderen. Door de lineaire veronderstelling van Principal Component Analysis los te laten, kunnen we veel meer oppervlakken aan dan klassieke methoden.

Het onderzoek strekt zich uit tot camera-kalibratie, waarbij de detectie van schaakbordpatronen wordt aangepakt door hoekdetectie en subpixel-verfijning. Een mapping van virtuele perfecte schaakbordhoeken naar pixels in echte beelden wordt geleerd, waardoor een oplossing ontstaat om ontbrekende hoeken af te leiden en bestaande te corrigeren. Deze gedachte wordt verder uitgewerkt in de camera-kalibratie door deze mapping om te keren. Nu leren we van pixels van echte schaakbordhoeken naar een perfect virtueel schaakbord. Deze benadering verandert camera's in perfecte pinhole-modellen.

Het onderzoek wordt afgerond door uit te weiden naar de toepassing van beoordeling en behandeling van visuospatieel neglect. Actieve leermethoden worden toegepast in een virtuele realiteitsomgeving om het aantal metingen te verminderen en daarmee de belasting voor patiënten met deze cognitieve stoornis te verminderen.

Samenvattend belicht de dissertatie significante bijdragen aan het veld van 3D-metingen en toepassingen, en biedt het een verscheidenheid aan innovatieve technieken en praktische toepassingen voor toekomstig onderzoek.

Contents

Chapter 1

Introduction

1.1 Context

Calibrating a measuring device is an initial and essential step before conducting any measurements. In this work, calibrating a device means formulating a model that can accurately produce a certain output when given some input variables. The output should be as close to the underlying physical truth as possible. The input variables are physical quantities we can directly observe or control. A running example in the first part of this text are galvanometric setups [37]. These devices aim a laser beam by means of two rotating mirrors, each mounted on a voltage-controlled galvanometer. The outputs are the laser lines themselves. The inputs are the two control voltages. A well calibrated galvanometric setup produces laser lines in the real world that do not differ significantly from the ones the underlying model predicts, given the two input voltages.

When formulating such a calibration model, several approaches are possible. First, one could try to mathematically model the physical reality by formulating equations that propagate the input parameters to the outputs. In our galvanometric setup example, this means modelling the mirrors as flat 2D planes in a 3D world and using geometry to calculate how a fixed incoming laser beam is reflected. An example is given in [100]. Calibration comes down to finding values for all parameters in the equations. In our galvanometric setup example, these are the positions of the mirrors, the distance between the mirrors, the angle of incidence between the incoming fixed laser and the first mirror, etc. The calibration of this device is in essence an optimisation problem, where we try to find a set of values for all the parameters that result in predicting the straight lines we observe as close as possible, given the input voltages. As the complexity of the model increases by introducing more nuanced equations, so does the number of variables, which means harder optimisation challenges. Besides having to deal with possibly many local optima, there is also the curse of dimensionality. This results in mathematical and physical calibration models that can be cumbersome to implement and hard to find solutions for. Moreover, these models often still make assumptions to simplify the equations, that might not always hold in reality. For instance, in the real world, mirrors are not always perfect flat planes.

Alternatively, one could bypass the underlying physical reality completely and turn the calibration challenge into a regression problem. That is, to learn the mapping from the input variables to the output directly. This is known as the data-driven approach. This has proved fruitful in a myriad of application domains such as spectroscopic calibration [97, 58], infrared spectroscopy [104], and LIDAR calibration [144].

Both the mathematical and the data-driven approach have pros and cons, but as we will advocate in the next chapters, a hybrid approach can be very beneficial when working with devices that measure a quantity from the 3D world we live in. When we re-introduce one or more geometrical assumptions into a data-driven calibration model, this approach becomes semi-data-driven. An example would be the fact that laser beams behave as straight lines in 3D.

The regression problem from the (semi-)data-driven approach can be solved by several machine learning methods, such as neural networks, support vector regression, polynomial regression and many more. This work is not a comparison between different machine learning techniques for calibration. We primarily focus on Gaussian processes, explained in Chapter 2 and in the book [130], as they come with several benefits in a calibration context:

- Gaussian processes work well in low data regimes. Calibration is usually a cumbersome and time consuming procedure. Being able to produce accurate calibration results on very few data points is an important plus.
- Training a Gaussian process means maximising a log-likelihood, which for Gaussian processes comes with a built-in mechanism against overfitting.
- As a Bayesian probabilistic method, Gaussian process predictions are Gaussian probability distributions. They consist of both an expected value and a variance. The latter can be seen as an uncertainty estimate. This can be further exploited by uncertainty propagation or techniques such as Bayesian optimisation and active learning.

All of these aspects of Gaussian processes will be addressed in much more detail in the following chapters.

We will mainly focus on 3D measuring devices that make use of straight lines in one way or another. We already mentioned the example of a galvanometric setup, as used in for instance laser Doppler vibrometers or laser scanners. We will also look at point clouds resulting from a 3D photogrammetric scanner and reformulate them as a set of normal lines with a point on them. Straight lines also appear in the field of camera calibration. Every pixel of an image taken by a camera is the result of an incoming ray of light, which is a straight line. Calibrating a camera means knowing which 3D line in the real world corresponds to which pixel in the image. Lastly, a person's field of view can also be modelled as a set of viewing directions, which can be seen as straight lines. Needless to say, straight lines play a vital role in many 3D measuring devices.

In summary, in this work, we will present novel probabilistic calibration and modelling techniques used in 3D measurements and applications. Moreover, we will explore how we can construct semi-data-driven approaches, in which Gaussian processes will predict straight lines.

1.2 Outline of this Dissertation

This dissertation consists of four main parts: galvanometric setup calibration, surface approximation, camera calibration and the clinical application of mapping visuospatial neglect.

Chapter 1 lays the foundation for this work, providing context for the research and outlining the structure of the dissertation. Furthermore, the publications resulting from this research are

summarised, including journal articles, conference proceedings, and a patent, highlighting the broader impact of this work.

Chapter 2 offers a comprehensive theoretical background, elucidating key concepts such as Gaussian processes, Gaussian process regression, and covariance functions. Additionally, the chapter delves into line geometry by introducing Plücker coordinates and screw theory, which is foundational for subsequent chapters.

Chapter 3 focuses on galvanometric setup calibration. By bypassing the geometric modelling of the setup completely, we obtain a purely data-driven method. However, we re-introduce one reasonable geometric assumption: laser lines are straight lines. Our method implements Gaussian processes to perform regression from the two mirror angles of the galvanometric setup to the Plücker coordinates of the resulting straight laser lines.

Chapter 4 builds on the approach of Chapter 3 by introducing constraints on both the inputs and the outputs of our Gaussian process models. The inputs are the two mirror rotation angles and as such, they can be seen as points on the Cartesian product of two circles. Topologically, these points form the surface of a torus. The outputs are the straight lines themselves. Plücker coordinates of straight lines in 3D space are by construction six-tuples. They are points in homogeneous 5-space that obey a quadratic constraint known as the Grassmann-Plücker relationship. We formulate several ways to handle this quadratic constraint on the outputs of the Gaussian process predictions.

In Chapter 5, the research shifts its focus to surface approximation by extending the known literature on line element geometry. We rewrite point clouds given by a 3D scanner into a set of normal lines with a point on them. This re-formulation opens the door for dimensionality reduction. Moreover, by relaxing the linear assumption of Principal Component Analysis, we can handle many more surfaces than classical methods. We base our method on Gaussian Process Latent Variable Models, which are a form of non-linear probabilistic dimensionality reduction.

Chapter 6 explores the domain of camera calibration. The detection of checkerboard patterns is addressed through corner detection and sub-pixel refinement. Our method learns a map from corners of a virtual perfect checkerboard to pixels in a real image. Missing corners can easily be inferred and existing corners can be corrected.

Chapter 7 turns the approach of chapter 6 around: now we learn the map from pixels of the corners of a real checkerboard to a perfect virtual checkerboard. This innovative approach transforms a camera into a perfect pinhole model. We explore this for various camera distortions.

Finally, Chapter 8 extends the research to address visuospatial neglect assessment and treatment. Active learning methods are applied to aid in the assessment and treatment of this cognitive disorder, with results and discussions highlighting the potential of AI and VR in this domain.

The dissertation concludes with a summary of the key findings and recommendations for future research in Chapter 9.

1.3 Publications

1.3.1 A1 Journal Articles

1. Ivan De Boi, Seppe Sels, and Rudi Penne. Semidata-driven calibration of galvanometric setups using Gaussian processes. IEEE Transactions on Instrumentation and Measurement, 71:1–8, 2021.
   https://doi.org/10.1109/TIM.2021.3128956

2. Ivan De Boi, Seppe Sels, Olivier De Moor, Steve Vanlanduit, and Rudi Penne. Input and output manifold constrained Gaussian process regression for galvanometric setup calibration. IEEE Transactions on Instrumentation and Measurement, 71:1–8, 2022.
   https://doi.org/10.1109/TIM.2022.3170968

3. Ivan De Boi, Carl Henrik Ek, and Rudi Penne. 2023. Surface approximation by means of Gaussian process latent variable models and line element geometry. Mathematics, 11, no. 2: 380.
   https://doi.org/10.3390/math11020380

4. Michaël Hillen, Ivan De Boi, Thomas De Kerf, Seppe Sels, Edgar Cardenas De La Hoz, Jona Gladines, Gunther Steenackers, Rudi Penne, and Steve Vanlanduit. 2023. Enhanced Checkerboard Detection Using Gaussian Processes. Mathematics 11, no. 22: 4568.
   https://doi.org/10.3390/math11224568

5. Ivan De Boi, Elissa Embrechts, Quirine Schatteman, Rudi Penne, Steven Truijen, and Wim Saeys. Assessment and treatment of visuospatial neglect using active learning with Gaussian processes regression. Artificial Intelligence in Medicine, 149:102770, 2024.
   https://doi.org/10.1016/j.artmed.2024.102770

1.3.2 P1 Conferences

1. Ivan De Boi. 2021. How to improve the accuracy of laser-based systems using straight lines. Metromeet - Conferencia Internacional sobre Metrología Industrial Dimensional. Bilbao, Spain.
   https://metromeet.org/ivan-de-boi/

2. Ivan De Boi, Stuti Pathak, Marina Oliveira, Rudi Penne. 2024. How to Turn Your Camera into a Perfect Pinhole Model. In: Vasconcelos, V., Domingues, I., Paredes, S. (eds) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP Conference 2023. Lecture Notes in Computer Science, vol 14469. Springer, Cham.
   https://doi.org/10.1007/978-3-031-49018-7_7

1.3.3 Patent

The work in Chapter 8 is protected by the following patent:

Computer Implemented Method And System For Mapping Spatial Attention
Inventors: Wim Saeys, Steven Truijen and Ivan De Boi
Publication Number WO/2022/122834

Publication Date 16.06.2022
International Application No. PCT/EP2021/084817
International Filing Date 08.12.2021
Priority Data 20212871.6 09.12.2020 EP

Chapter 2

Theoretical Background

2.1 Introduction

In this chapter we provide some theoretical background on the two important subjects in this work. First, we elaborate on Gaussian processes. This probabilistic machine learning technique follows the Bayesian paradigm and thus allows us to make predictions incorporating prior knowledge. As they have a built-in mechanism against overfitting, handle noise very well and can work with small datasets, they are very well suited for calibration. In short, they offer a probability distribution over functions that fit the data. For regression, we are mostly interested in the mean function. However, this distribution also comes with an uncertainty estimate, which serves as a measure of the quality of the prediction or can be exploited in algorithms such as Bayesian optimisation or active learning. Second, we introduce Plücker coordinates, which are a mathematically elegant way to describe straight lines in 3D space. Equipped with this formulation, we briefly explore the field of line geometry.

2.2 Gaussian Processes

2.2.1 Gaussian Process Regression

By definition, a Gaussian process (GP) is a continuous collection of random variables, any finite subset of which is normally distributed as a multivariate distribution. A more comprehensive treatment can be found in [130]. Here, we give a brief introduction with focus on practical implications.

Let $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ be a dataset of n observations, where $\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\right]^T$ is an $n \times d$ matrix of n input vectors of dimension d and $\mathbf{y} = \left[y_1, y_2, ..., y_n\right]^T$ is a vector of continuous-valued scalar outputs. These data points are also called training points. Regression aims to find a mapping $f : \mathbb{R}^d \to \mathbb{R}$,

$$y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2), \tag{2.1}$$

with ε being identically distributed observation noise. In stochastic numerics, this mapping is often implemented by a Gaussian process, which is fully defined by its mean $m(\mathbf{x})$ and *covariance*

function $k(\mathbf{x},\mathbf{x}')$. It is generally denoted as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}),k(\mathbf{x},\mathbf{x}'))$. Typically, the covariance function, also referred to as a *kernel* or *kernel function*, is parameterised by a vector of hyperparameters θ. By definition, a GP yields a distribution over a collection of functions that have a joint normal distribution [130],

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}m_{\mathbf{X}}\\ m_{\mathbf{X}_*}\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{\mathbf{X},\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{X}_*}\\ \mathbf{K}_{\mathbf{X}_*,\mathbf{X}} & \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*}\end{bmatrix},\right) \tag{2.2}$$

where $\mathbf{X}$ and $\mathbf{X}_*$ are the input vectors of the n observed training points and the n_* unobserved test points, respectively. The vectors of mean values for $\mathbf{X}$ and $\mathbf{X}_*$ are given by $m_{\mathbf{X}}$ and $m_{\mathbf{X}_*}$. The covariance matrices $\mathbf{K}_{\mathbf{X},\mathbf{X}}$, $\mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*}$, $\mathbf{K}_{\mathbf{X}_*,\mathbf{X}}$ and $\mathbf{K}_{\mathbf{X},\mathbf{X}_*}$ are constructed by evaluating k at their respective pairs of points. In practice, we do not have access to the latent function values directly, which are dependent on the noisy observations $\mathbf{y}$.

Putting it all together, the conditional (predictive posterior) distribution of the GP can be written as:

$$\mathbf{f}_*|\mathbf{X},\mathbf{X}_*,\mathbf{y},\theta,\sigma_\varepsilon^2 \sim \mathcal{N}(\mathbb{E}(\mathbf{f}_*),\mathbb{V}(\mathbf{f}_*)) \tag{2.3}$$

$$\mathbb{E}(\mathbf{f}_*) = m_{\mathbf{X}_*} + \mathbf{K}_{\mathbf{X}_*,\mathbf{X}}\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_\varepsilon^2 I\right]^{-1}\mathbf{f} \tag{2.4}$$

$$\mathbb{V}(\mathbf{f}_*) = \mathbf{K}_{\mathbf{X}_*,\mathbf{X}_*} - \mathbf{K}_{\mathbf{X}_*,\mathbf{X}}\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_\varepsilon^2 I\right]^{-1}\mathbf{K}_{\mathbf{X},\mathbf{X}_*} \tag{2.5}$$

Without loss of generality, the mean function m is often set to zero. This does not imply a zero mean posterior $\mathbb{E}(\mathbf{f}_*)$, which in fact does not depend on the mean of the training points at all. Note also that the variance $\mathbb{V}(\mathbf{f}_*)$ does not depend on the observations $\mathbf{y}$, but only on the location of the test points $\mathbf{X}_*$. This is a property of the Gaussian distribution

An extensive overview of kernels and how to implement and combine them can be found in [43]. Hyperparameters θ are usually learned by using a gradient based optimisation algorithm to maximise the log marginal likelihood,

$$\log p(\mathbf{y}|\theta,\mathbf{X}) \propto -\frac{1}{2}\mathbf{y}^T\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_\varepsilon^2 I\right]^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_\varepsilon^2 I|. \tag{2.6}$$

Unless stated otherwise, in our experiments, we use L-BFGS, a quasi-Newton method described in [94], to maximise this log marginal likelihood. We call Equation 2.6 the log marginal likelihood, as it is obtained through marginalisation over the latent function $\mathbf{f}$.

The terms in Equation 2.6 are a combination of a data fit term $\mathbf{y}^T\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_\varepsilon^2 I\right]^{-1}\mathbf{y}$ and complexity penalty term $\log|\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_\varepsilon^2 I|$. The data fit term is the only one that depends on the data itself. The complexity penalty term is based on the determinant of the covariance matrix. It is also sometimes called the capacity control term, because a larger determinant corresponds to a covariance matrix that captures more volume in data space. It can be interpreted as a larger ellipsoid in data space, where the principal axes are formed by the eigenvalues and eigenvectors of the covariance function. The determinant is the product of the eigenvalues. In addition, a larger determinant corresponds to a larger function space, as more functions can explain the data. Another way of looking at the determinant is regarding it as the spread of the multivariate Gaussian distribution we are modelling. Training a Gaussian process comes down to limiting the space of possible functions as much a possible, while still being able to explain the data through the data fit term. We search for hyperparameters that result in a covariance matrix with a small determinant. In other words, we search for a more narrow Gaussian distribution. Of course, this

is continuously traded-off with the data fit term. The training procedure automatically incorporates Occam's Razor [129]: *The simplest explanation is usually the best one.* This guards the GP against overfitting. The best solution is the simplest one. The trade-off between data fit and model complexity is automatic. There is no parameter controlling the trade-off.

For efficiency and numerical stability, the linear system $\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_\varepsilon^2 I\right]^{-1} \mathbf{y}$ is often calculated by first calculating the Cholesky decomposition factor L of $\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_\varepsilon^2 I\right]$ and then solving

$$\left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma_\varepsilon^2 I\right]^{-1} \mathbf{y} = L^T \setminus (L \setminus \mathbf{y}). \tag{2.7}$$

2.2.2 Covariance Functions

There exists a large variety of kernel functions, which have been extensively studied and reviewed in [60]. The de facto default kernel used in Gaussian processes is the squared exponential kernel (SE), also called the radial basis function kernel, the Gaussian kernel, or exponentiated quadratic kernel. It is applicable in a wide variety of situations because it is infinitely differentiable and thus yields smooth functions. Besides that, it only takes two hyperparameters to tune. It has the form:

$$k_{SE}(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\left(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{2l^2}\right), \tag{2.8}$$

in which σ_f^2 is a height-scale factor and l the length-scale that determines the radius of influence of the training points. For the squared exponential kernel, the hyperparameters θ_{SE} are $\left\{\sigma_f^2, l\right\}$.

Alternatively, a different length-scale parameter for every input dimension can be implemented. This technique is called automatic relevance determination (ARD) and allows for functions that vary differently in each input dimension [43]. The kernel has the form:

$$k_{SEARD}(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{j=1}^{d}\left(\frac{\left|\mathbf{x}_j - \mathbf{x}'_j\right|}{l_j}\right)^2\right), \tag{2.9}$$

in which l_j is a separate length-scale parameter for each of the d input dimensions.

In this work, all GP models are equipped with a squared exponential kernel, unless stated otherwise. Producing smooth functions is a property of this kernel that serves our purpose very well. The underlying reality of the devices investigated in this work is always a very smooth function from input variables to outputs. The galvanometric setup aims a laser beam by means of two rotating mirrors. There are no discontinuities in the function from rotation angles of the mirrors to the resulting laser beams. A similar reasoning can be given for the other applications in this dissertation. This smoothness also allows us to diminish the effect of outliers. However, as we will see in Chapter 6, it is crucial to exercise caution to avoid over-smoothing when working with heavily distorted images.

2.2.3 Training Recommendations

In this thesis, we relied on the use of the MATLAB implementation of Gaussian processes and GPy[1]. The latter is a Python framework, from the Sheffield machine learning group. Both allow for various options for training a Gaussian Process.

Unless stated otherwise, we standardised our inputs to zero mean and scaled them to have unit variance. Generally, this results in better conditioning of the covariance matrix. Most software packages do this automatically, unless explicitly disabled. Alternatively, one can initialise the hyperparameters to reasonable values based on prior knowledge of the problem at hand. For instance, by using the median pairwise distance for the length-scale. As for the noise variance estimate, one can start close to zero and allow the GP to learn the noise level from the data itself by increasing it.

To prevent overfitting, we can regularise the hyperparameters by restricting their values between certain bounds. We did this in Chapter 6 and 8. For the checkerboard corner detection in Chapter 6, the length-scale is determined by the size of the squares in the checkerboard and the level of distortion of the camera. In our code, this hyperparameter is tunable by setting boundaries. We also refer to the extensive documentation in our code base itself. Having a length-scale that is too big, results in a flattening of the checkerboard in the image. Having a length-scale that is too small, results in bad extrapolations outside the checkerboard, which makes it harder to detect new corners. Examples of this are given in our Python package PyCBD[2].

In Chapter 8, we put an upper bound on the length-scale to prevent the model from becoming overconfident after just a handful of measurements. In this context, the aim is to reduce the number of measurements as much as possible. The danger is that after, for instance, three consecutive and very similar measurements, the model would learn a very large length-scale, resulting in a generalisation of this measured value to the entire input space. This is detrimental to active learning, as the posterior uncertainty also collapses. In other words, the model becomes overly confident that this measurement value can be found everywhere.

As stated above, the L-BFGS optimiser is used throughout. Alternatives are available in both the MATLAB implementation and GPy, but are outside the scope of this thesis. However, this method can get stuck at a local minimum. A thorough explanation of this phenomenon is presented in the book [130].The solution for this is to perform several restarts of the optimisation process with a different set of initial values. We advise setting the parameter that controls the number of restarts to five. Reducing the number of restarts is permissible only under the condition that the optimisation algorithm consistently converges to the same set of hyperparameter values.

2.2.4 Dealing with Outliers

Severe outliers can have an impact on the outcome of any model. Generally speaking, Gaussian processes are well equipped against this by considering them *unlikely* in a Bayesian setting [130]. Although, in the case of extreme values or in the case of abundance of outliers, the models can lose their ability to cope with these. This problem has been well studied in literature. One can change the likelihood model from Gaussian to Student-t, which has fatter tails [143]. Alternatively, one

[1] `http://sheffieldml.github.io/GPy/`
[2] `https://pypi.org/project/PyCBD/`

could resort to deep Gaussian processes [29]. These layered models have the ability to warp the data from one layer to another, allowing to shift outliers to their more likely neighbours. More recent work studied the splitting of datasets into inliers and uniformly distributed outliers by incorporating this in the likelihood formulation [98]. The downside of these methods is the increase in training time due to added complexity. The main issue is the fact that the posterior is no longer Gaussian. This is why we did not opt for any of these in this work. We deal with outliers on several levels and in several ways. Each level has his own implications. The reader is advised to take these into consideration when applying our methods on their own application.

First, when working with straight lines, we do not observe them directly. We measure points in 3D space, on which we fit a best fit line. For the details, we refer the reader to Chapter 3 and 4. However, we will encounter that some of those points clearly do not belong to the line we are trying to find. In this case, we filter out those points by means of a RANSAC pre-processing step. What we end up with, is a best fit line on inliers. It is the aggregate of all this lines on which the various GP models are trained. We do not remove a single line from the dataset, unless not enough points can be found to perform a best fit on. This means that from the perspective of the Gaussian process, no *outlier-line* has been removed. It does mean that outlier removal takes place on the points instead of on the training data itself, i.e., the set of lines. This is a subtle but important nuance. It is application specific to whether or not this pre-processing step is justified.

Second, in Chapter 6 on checkerboard detection, we flip this reasoning around. We explicitly rely on the Gaussian process posterior prediction to deem a detected corner an outlier or not. In other words, a detected corner that does not fit the prediction is labelled an outlier and not retained in the set of valid corners of the checkerboard.

Third, when working with point clouds resulting from a real world 3D scanning procedure, we always have some points that are not of interest. In Chapter 5, we firstly cleaned up the models we are validating on. For instance, we removed the vertices of the table on which our object of interest lies. This is in fact removing data from the dataset and should only be done with some considerations in mind. Details are in the chapter itself and all models are available in the online material. In Section 5.3.3, we did include one experiment in which we deliberately inserted outliers in the data to assess how the models can deal with this.

Fourth, as mentioned above, in Chapter 8 we are trying to assess a patient's visuospatial neglect condition with as few measurements as possible. If one of those measurements is a false positive or negative, then the result is severely influenced in the wrong direction. In our application, we added the additional check, that a stimulus that is successfully spotted cannot be surrounded by stimuli that are not spotted in time. And vice versa, stimuli that are missed cannot be surrounded by stimuli that are successfully spotted. In other words, the heatmaps should not show islands. An island is a blind spot that is clinically not possible in the context of visuospatial neglect. This is why we consider these false positives or negatives as outliers, and we remove them from the training data. If we had an unlimited time budget, we could leave them in, and the Gaussian process would smooth them out. But in practice, this is not feasible when working with real patients.

2.3 Line Geometry

Our workspace is modelled by the Euclidean 3-space $\mathbb{E}^3$, both theoretically and in real-world applications. The reader is advised not to confuse this symbol with the expectation of a distribution. However, for geometric computations, it is more convenient to embed $\mathbb{E}^3$ in a larger real projective 3-space $\mathbb{P}^3$, adding the so-called points at infinity.

2.3.1 Plücker Coordinates

A straight line $\mathbf{L}$ in projective 3-space $\mathbb{P}^3$ can be represented by its direction $\mathbf{l}$ and a point $\mathbf{p}$ that on that line. The so-called moment vector for that line with respect to the origin, is obtained by

$$\bar{\mathbf{l}} = \mathbf{p} \times \mathbf{l}. \tag{2.10}$$

The six coordinates $(\mathbf{l}, \bar{\mathbf{l}}) = (l_1, l_2, l_3, l_4, l_5, l_6)$ are called the *Plücker coordinates* or the *homogeneous line coordinates* of the line $\mathbf{L}$ [122]. They are independent of the choice of $\mathbf{p}$. Since we are not concerned about the orientation, $(\mathbf{l}, \bar{\mathbf{l}})$ and $(-\mathbf{l}, -\bar{\mathbf{l}})$ describe the same line. Plücker coordinates do depend on the scale of $\mathbf{l}$, such that they must be considered as homogeneous coordinates. Some works emphasise this by denoting $\mathbf{L}$ as $(l_1 : l_2 : l_3 : l_4 : l_5 : l_6)$. However, we will not take that approach here.

For example, a line $\mathbf{L}$ is spanned by two given points x and y, possibly at infinity. By following the notation in [124], we write the homogeneous coordinates for x and y as $(x_0, \mathbf{x})$ and $(y_0, \mathbf{y})$ respectively. Then, the homogeneous Plücker coordinates for $\mathbf{L}$ are found as

$$\mathbf{L} := (\mathbf{l}, \bar{\mathbf{l}}) = (x_0\mathbf{y} - y_0\mathbf{x}, \mathbf{x} \times \mathbf{y}) \in \mathbb{R}^6. \tag{2.11}$$

A rigid body in $\mathbb{E}^3$ has six degrees of freedom: three for rotations and three for translations. Straight lines however, only have four degrees of freedom. Rotating a line around its direction or translating it along its direction, yields the same line. Since these coordinates are homogeneous, we can normalise them by having their direction vector at unit length:

$$\|\mathbf{l}\| = 1. \tag{2.12}$$

This normalisation removes one degree of freedom from the six-tuple. Moreover, the moment of a straight line is by definition perpendicular to its direction:

$$\mathbf{l} \cdot \bar{\mathbf{l}} = 0, \tag{2.13}$$

$$l_1 l_4 + l_2 l_5 + l_3 l_6 = 0. \tag{2.14}$$

This quadratic condition is called the *Grassmann-Plücker relation* or in some works the *Plücker constraint*. Only six-tuples that obey this condition are straight lines, so this also removes one additional degree of freedom. Plücker coordinates can also be interpreted as points in projective 5-space $\mathbb{P}^5$, which lie on the *Klein quadric* M_2^4. This quadric is a four-dimensional surface of degree two, given by Equation 2.13.

This is just a reformulation of the Grassmann-Plücker relation in Equation 2.13. Representing straight lines by Plücker coordinates has proved useful in a variety of problems. A more in-depth explanation of line geometry can be found in the book [124].

2.3.2 Screws

As mentioned above, not all six-tuples of real numbers represent a straight line in $\mathbb{P}^3$. In order to do so, they have to satisfy the Grassmann-Plücker relation. In general, a six-tuple C can be written as $(\mathbf{c}, \bar{\mathbf{c}})$. This is called a *screw centre* or just *screw*. The pitch ρ of C is defined as

$$\rho = \frac{\mathbf{c} \cdot \bar{\mathbf{c}}}{\|\mathbf{c}\|^2}. \tag{2.15}$$

This only holds for $\mathbf{c}$ not being the zero vector, in which case C represents a line at infinity. Straight lines are described by six-tuples with zero pitch.

According to the *Central Axis Theorem of Poinsot* [26], we can always write C as

$$C = (\mathbf{c}, \bar{\mathbf{c}} - \rho\mathbf{c}) + (\mathbf{0}, \rho\mathbf{c}) = A + (\mathbf{0}, \rho\mathbf{c}), \tag{2.16}$$

in which A is called the *Poinsot* or *central axis* of the screw centre C. The term $(\mathbf{0}, \rho\mathbf{c})$ represents the line at infinity where the planes perpendicular on A meet. Since A does satisfy the Grassmann-Plücker relation, it is a straight line in $\mathbb{P}^3$. This allows us to correct a six-tuple into a straight line A via

$$A = C - (\mathbf{0}, \rho\mathbf{c}). \tag{2.17}$$

We will find that using a Gaussian process to predict a six-tuple that is actually a straight line is not trivial. More often than not, we will end up with a prediction that is a screw and not a straight line. It will be application dependent whether the pitch of this screw is within a region of tolerance or that we have to invoke Equation 2.17 to correct the prediction. In Chapter 4 we will introduce more elaborate kernels to enforce the Grassmann-Plücker relationship on Gaussian process predictions.

Part I

Calibration of Galvanometric Setups

Chapter 3

Semi-data-driven Calibration

Traditionally, calibration of a galvanometric setup is based upon a mathematical model of an underlying physical reality. These models make a considerable number of assumptions and simplifications. Moreover, they tend to be non-generalisable and lead to non-convex optimisation problems encompassing many parameters. Alternatively, several data-driven statistical approaches have been proposed, in which any model of the underlying reality is completely bypassed. The often black-box model is trained purely on the data itself. Although some precautions for overfitting need to be kept in mind, it has been shown that this radically different approach can outperform the traditional mathematical models. On the other hand, some assumptions about the underlying physical truth are both reasonable and simple to implement. In this chapter, we propose to keep the best of both worlds and construct a semi data-driven calibration model with a single built-in assumption: rays exiting the galvanometric setup are straight lines. The data-driven approaches do not exploit this obvious fact. In this work, we focus on the intrinsic calibration of the galvanometric setup, i.e. finding the relationship between the input parameters that control the galvanometers and the Plücker coordinates of the laser beams exiting the device. We investigate four different models to predict lines and evaluate them in cross-validation, predicting intersection points on a validation plane and aiming the laser at a specific point in 3D space. We show that our approach outperforms a purely data-driven approach.

We published the findings of this chapter in [37]. We also gave a talk entitled *How to improve the accuracy of laser-based systems using straight lines* at the Metromeet 2021 Conferencia Internacional sobre Metrología Industrial Dimensional in Bilbao, Spain.
`https://metromeet.org/ivan-de-boi/`

3.1 Introduction

Galvanometric setups are widely used in many different applications, ranging from Laser Doppler Vibrometry [19] to bar-code readers [6], 3D laser scanners [45], medical imaging [112] and LIDAR systems [50]. They also play an important role in computer vision for self driving cars [167].

Traditionally, these setups consist of two rotating mirrors that guide a laser or light beam. The motors rotating the mirrors are called *galvanometers* [150]. More recently, progress has been made in the miniaturisation of the hardware and the elimination of moving parts by means of

phase-changing materials and broadband lasers, which can adjust diffraction angles by changing their frequency [82]. A detailed image of the two rotating mirrors of a Polytec OFV 056 vibrometer is given in Figure 3.1.

Many applications demand accurately calibrated setups. In this work we focus on the intrinsic calibration of the galvanometric itself, i.e. finding the relationship between the input parameters that control the galvanometers and the rays exiting the device. Extrinsic calibration, the localisation of the device in a reference system, has been investigated in [140]. We assume the availability of a calibrated 3D-sensor that has been intrinsically calibrated in a preprocessing phase. In Chapter 7, we will address a novel calibration method applicable for a general camera.

As explained in [100], the common pinhole model cannot be used for calibrating a galvanometric setup, because there is no single centre of projection. To overcome this, the authors propose a mathematical approach to model the geometry of laser reflections. In [101], a transformation of coordinate system is described to correct the effects of the geometric distortion of galvanometer scanners. The main drawback of these approaches is their complexity, resulting in difficult non-convex optimisation tasks. Furthermore, the models fail to account for all distortions such as the ones caused by non-planar mirrors.

Alternatively, calibration in general can be performed by a purely data-driven approach. Instead of constructing a model based upon mathematical assumptions about the underlying physical reality, the calibration challenge is transformed into a regression problem. This is proved fruitful in spectroscopic calibration [97, 58], infrared spectroscopy [104], and LIDAR calibration [144].

Data-driven calibration for galvanometric laser scanners was proposed in [162]. The authors implemented artificial neural nets (ANN), Support Vector Regression (SVR) and Gaussian processes (GP). These methods proved to outperform the mathematical model and look-up table-based calibration procedures.

However, their results also show that the models are sensitive to overfitting. Imagine a straight line intersecting several parallel planes. The intersection points are, by construction, collinear. Noisy real-world measurements will yield coordinate values for points that are not perfectly collinear. Training models on these noisy values will result in predictions of points that are also not collinear. Our approach exploits the knowledge that those predicted points have to be collinear. The data-driven models do not impose such a strong relationship between those points.

In this chapter, we propose four different models that can be used to intrinsically calibrate galvanometric setups. These models learn the mapping between the two input parameters that control the rotation of the mirrors and the straight lines that represent the rays of the laser beams exiting the setup. We evaluate our models using three different scenarios: cross-validation, predicting intersection points on a validation plane and aiming at a specific point in 3D space. We perform these evaluations on both real world and synthetically generated data and show that our approach outperforms the purely data-driven one.

This work is not a comparison between different machine learning techniques for calibration. As already investigated in [162], data-driven calibration for galvanometric laser scanners outperforms the mathematical model and look-up table-based calibration procedures. We restrict ourselves to comparing our methods only to the data-driven one. As mentioned in Chapter 1, we focus on Gaussian processes, as they are a flexible regression tool that provide a build in mechanism against overfitting. Training the model is achieved by optimising a set of hyperparameters by maximising the marginal likelihood. In this process, a complexity term is involved that penalises

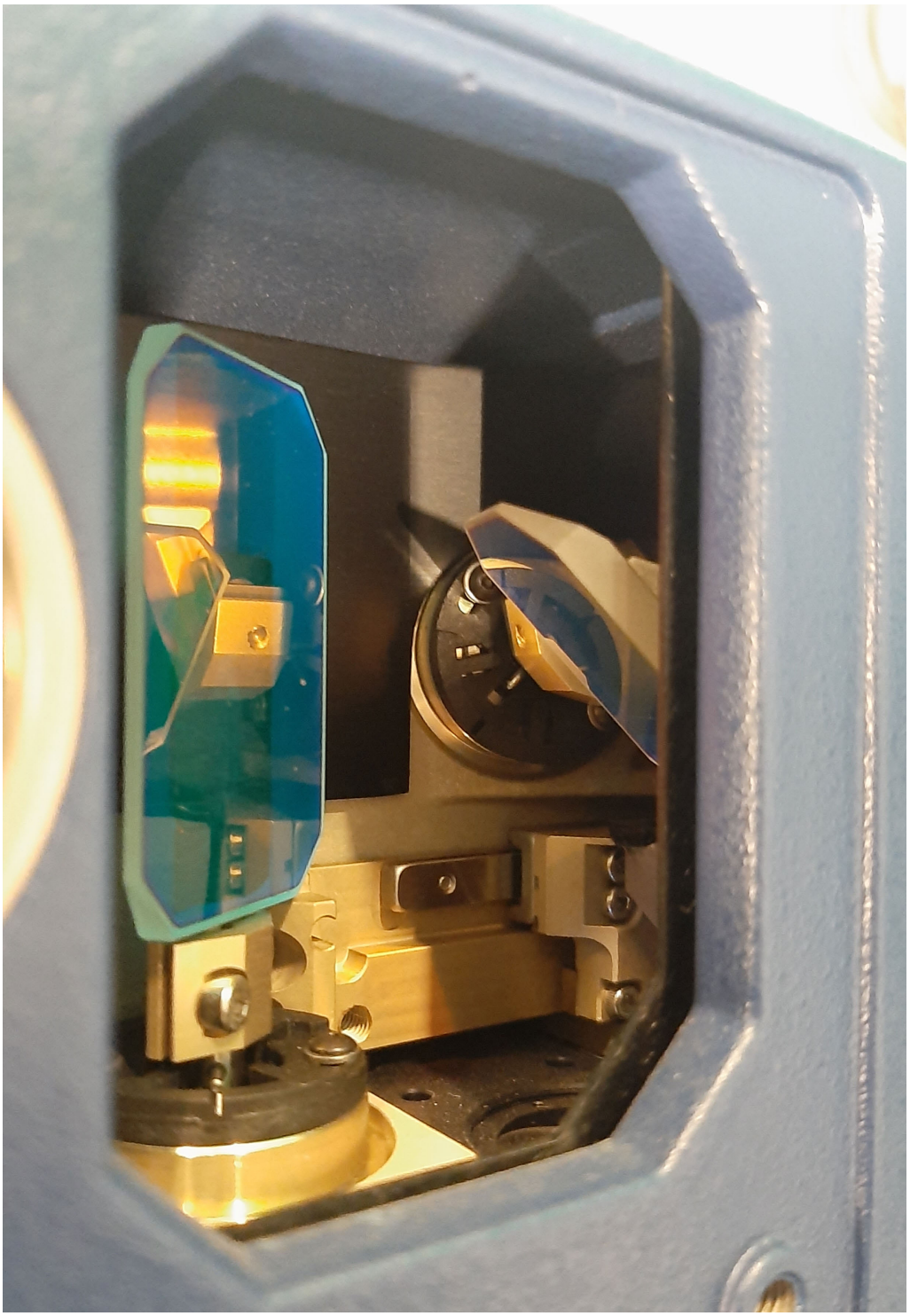


Figure 3.1: The two rotating mirrors inside a vibrometer.

over-complex models [130]. Furthermore, they follow a non-parametric Bayesian paradigm, in which the data speaks for itself.

The rest of this chapter is structured as follows. In the next section, we explain how we gathered the data. The third section describes the four models we investigate. In Section 3.4 we present the results from cross-validation, the ability of the models to predict a point on a validation plane and the assessment of the aiming capabilities of the models. In Section 3.5 we discuss these findings. Finally, we elaborate on our conclusions.

3.2 Gathering Data

In order to construct a dataset consisting of straight lines, a laser beam controlled by two rotating mirrors was directed towards a detection plane. This plane was moved into different positions. For every plane position and every pair of mirror rotations, the 3D coordinate of the intersection point of the laser with the plane was detected with a camera. A schematic overview is provided in Figure 3.2.

This setup was implemented in both the real world and a virtual environment. The latter allows us to perform validation on an exact ground truth that is free from the measurement error on the 3D coordinate detection.

3.2.1 Real World Data

The first dataset was recorded with a setup using a Polytec OFV 056 Vibrometer Scanner. The point where the laser hits the plane, forms a red dot. This dot was detected in an image taken by the RGB-camera from a Microsoft Kinect. We did not use the depth information of the Kinect. Additionally, a calibration pattern is detected and used to calculate the 3D coordinates of the detected dot. The procedure is explained in more detail in [142]. A picture of this setup can be seen in Figure 3.3. In this case, the two input parameters that control the rotations of the mirrors, are voltages. The galvanometers of the vibrometer rotate each mirror accordingly. They were ranged over 22 values each, resulting in 484 laser beams. The plane was put in 9 different positions, as can be seen in Figure 3.4. Points from one particular plane, which we will call the validation plane, were kept separate. For every pair of mirror rotations, the corresponding collinear points were filtered via RANSAC [51]. A straight line was fitted on the remaining inlier points (except the ones from the validation plane) using the least squares method described in [90]. A total of 226 lines were kept.

3.2.2 Synthetic Data

The second dataset was generated in a virtual world. We used the Unity game engine (version 2020.2.5f1) to recreate the real world setup. Again, two parameters control the mirrors that guide a laser beam towards a detection plane. However, in this case we have full control over the rotations directly. We used the built-in physics engine to calculate the rays, their reflections and the positions where the laser beams hit the detection plane. Figure 3.5 shows the virtual setup.

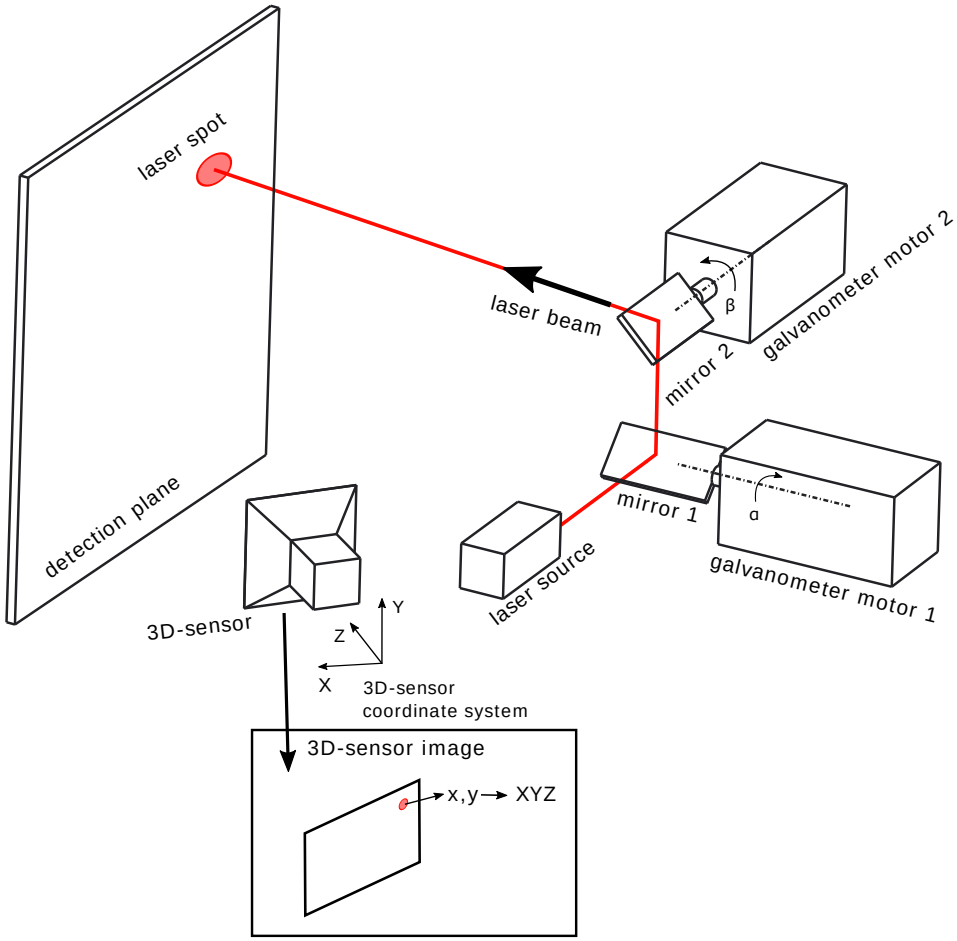


Figure 3.2: Overview of the galvanometric setup. The laser beam is controlled by two rotating mirrors. The rotations are represented by α and β. The 3D coordinate of the intersection point on the detection plane is measured by a 3D-sensor, which in our case is a camera.

The mirror rotations ranged over 20 values each, resulting in 400 laser beams. The plane was also put in 9 different positions, including a validation plane. Noise was added to each point, proportional to the distance of the origin. This corresponds to diminishing accuracy in measurements further away from the camera that is doing the 3D coordinate detection. The resulting points are shown in Figure 3.6. In the same manner as for the real world data, a line was fit on the RANSAC-filtered points. This resulted in 217 straight lines retained for the dataset.

3.3 Models

The models in this research aim to find a mapping (function) from the galvanometric setup input parameters to a set of straight lines. In this work, the inputs parameters are either the voltages

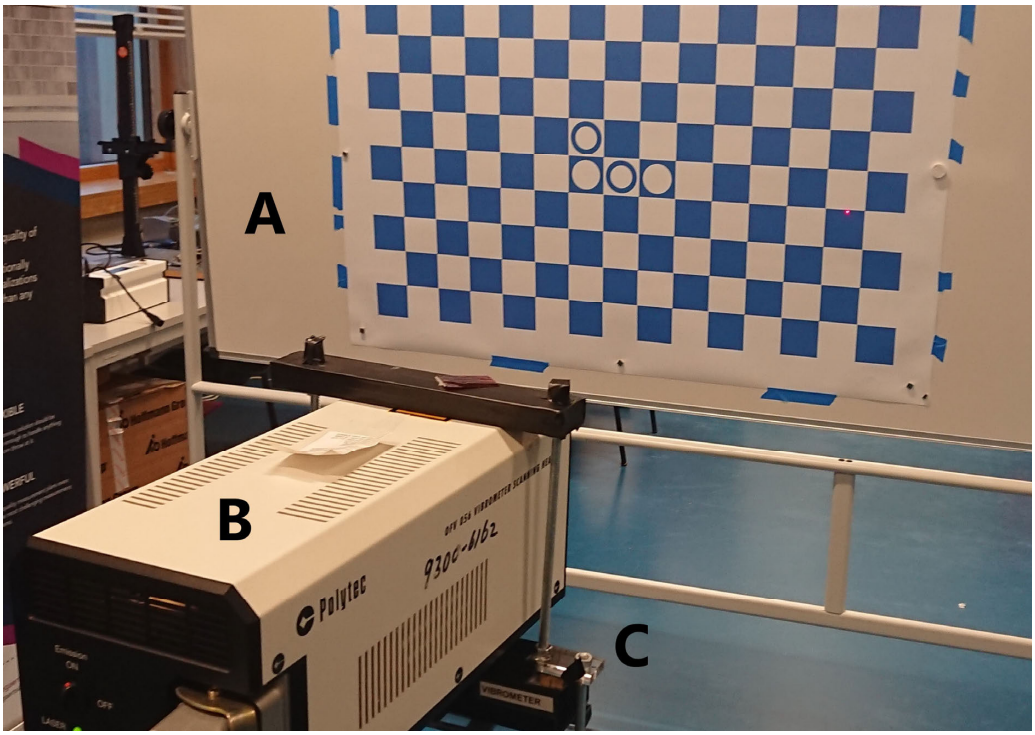


Figure 3.3: The setup in the real world. A: The detection plane with the calibration pattern. B: Polytec OFV 056 Vibrometer Scanner. C: Microsoft Kinect.

that control the real world galvanometers or the angles of the virtual mirrors themselves. In either case, we will label them α and β. As stated in the previous section, straight lines can be described by a six-tuple of numbers. Thus the mapping has the form $f : (\alpha, \beta) \rightarrow (l_1, l_2, l_3, l_4, l_5, l_6)$. We investigate and compare four models. Each model takes as input a value for α and β and produces a six-tuple, either directly, after correction or by construction. Both the correction and the construction have the purpose of fulfilling the Grassmann-Plücker relation.

Throughout, we base ourselves on a right-handed Cartesian coordinate system that is defined as follows. The Z-axis is pointing in the horizontal forward direction of the vibrometer. The Y-axis corresponds to the vertical axis. The X-axis is perpendicular to the YZ-plane. Laser beams leave the device in the positive z-direction. We use the reference frame of the 3D-sensor itself, meaning it is positioned at coordinate (0,0,0).

3.3.1 Six Distinct Gaussian processes

The most straightforward way to learn this mapping is to implement 6 distinct Gaussian processes, i.e. one for each output component. Alternatively, a single Gaussian process with an extra input for the number of the output dimension could be implemented [5]. An elaborate explanation on multi-output Gaussian processes can be found in [95]. However, we are assuming independent outputs. This means the covariance matrix has zero-valued entries at positions that correspond to outputs from different dimensions.

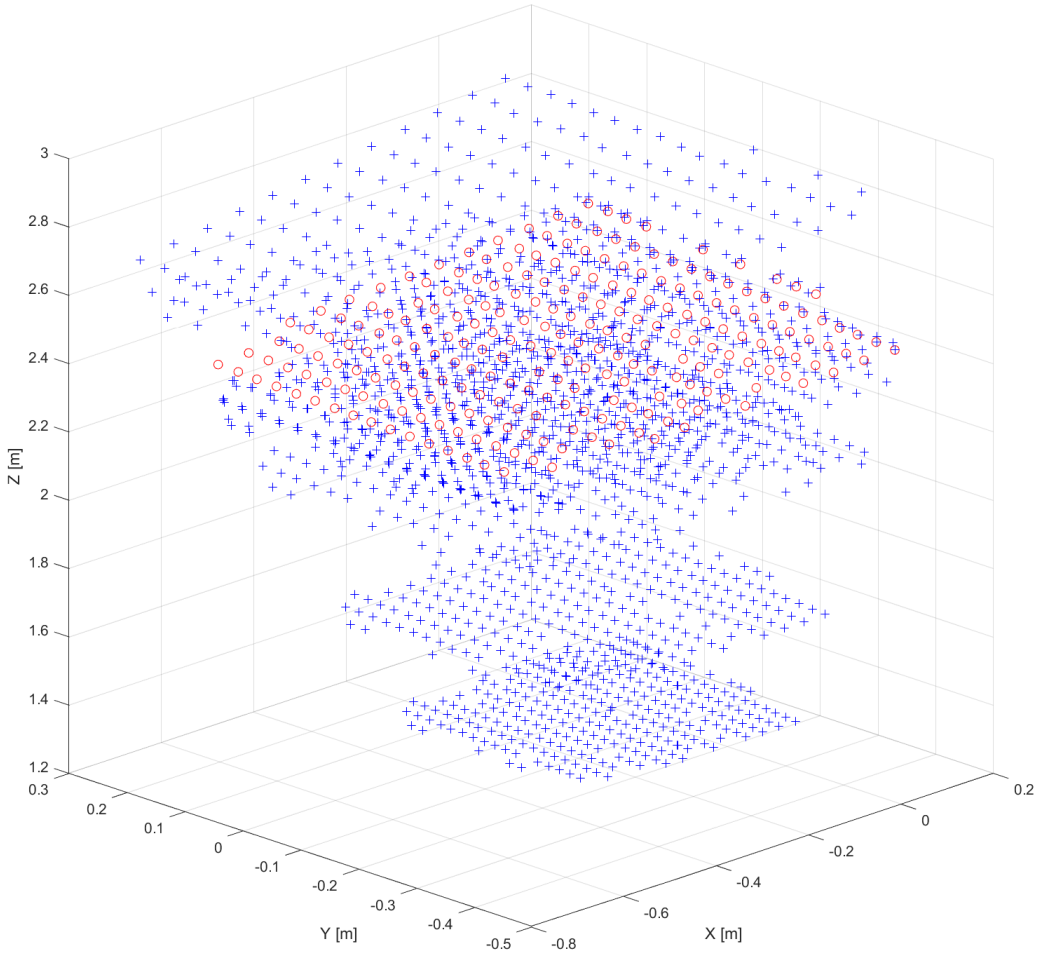


Figure 3.4: Measured points in the real world. The blue + were used to fit a straight line. The red o form a validation plane and were not used to construct the straight lines in de dataset.

Moreover, combining six Gaussian processes into one multi-output Gaussian process, results in a single set of hyperparameters, consisting of one height-scale σ_f^2 and two length-scales l_1 and l_2 (one for each input). Independent Gaussian processes on the other hand, each with their own height-scale and length-scales, allow for more flexibility in the model. The laser beams leaving the (real or virtual) vibrometer are all pointing in the forward direction of the device. This means the z-component of the direction of the straight line representing the laser varies very little in respect to the other components. This discrepancy can be captured by working with six distinct Gaussian processes.

3.3.2 Poinsot Axis

The method described in the previous section has a major drawback. The six Gaussian processes predict a six-tuple that does not necessarily obey the Grassmann-Plücker relationship in Equation 2.13. As mentioned in Section 2.3.2, we can correct the predicted six-tuple, generally a screw with

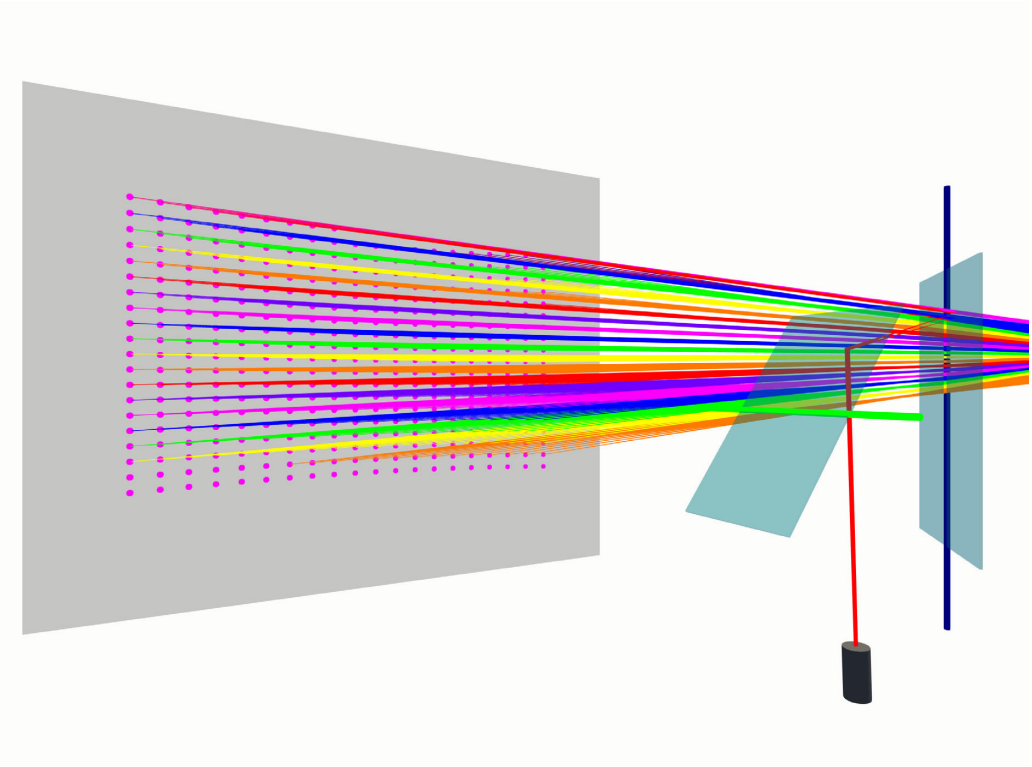


Figure 3.5: The setup in a virtual world created with the game engine Unity. The 3D coordinates of the pink dots are recorded.

a non-zero pitch, to its Poinsot axis A using Equation 2.17. This model is in effect the same as the previous one built on six independent Gaussian processes with an additional post-processing step of calculating the Poinsot axis based on the predicted six-tuple. We regard the Poinsot axis as the outcome of this model, and as such it represents a six-tuple of pitch zero, which is a straight line.

3.3.3 Direction and Intersection Point

An alternative way to ensure that the predicted six-tuples are straight lines, is by construction. In this third model we also use three distinct Gaussian processes for the direction $\mathbf{l}$. However, the moment $\bar{\mathbf{l}}$ is not learned directly. Instead, we find the intersection point of a straight line with direction $\mathbf{l}$ and a fixed plane. For simplicity, this plane goes through the origin. The intersection point $\mathbf{p}$, with homogeneous coordinates (p_0, p_1, p_2, p_3), of a line $(\mathbf{l}, \bar{\mathbf{l}})$ with a plane π, with homogeneous plane coordinates $(\pi_0, \pi_1, \pi_2, \pi_3)$, is given by

$$\mathbf{p} = ((\pi_1, \pi_2, \pi_3) \cdot \mathbf{l}, -\pi_0 \mathbf{l} + (\pi_1, \pi_2, \pi_3) \times \bar{\mathbf{l}}). \tag{3.1}$$

We train three Gaussian processes on the 3D coordinates of those intersection points. The moments $\bar{\mathbf{l}}$ of the lines are calculated via Equation 2.10. By construction, they are perpendicular to the direction $\mathbf{l}$ and thus always satisfy the Grassmann-Plücker relation.

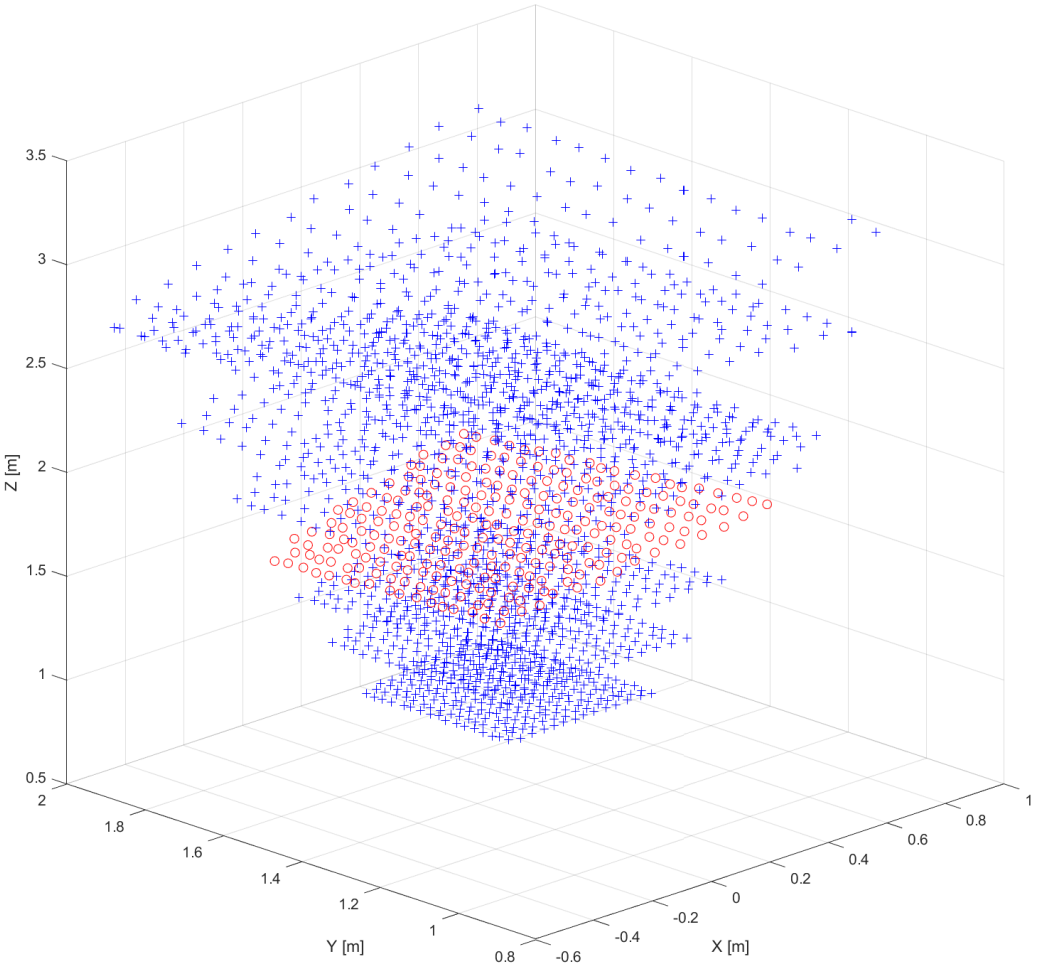


Figure 3.6: Measured points in the virtual world. Straight lines are fitted on the blue +, while the red o form a validation plane.

3.3.4 Two Intersection Points

Since straight lines in $\mathbb{P}^3$ only have four degrees of freedom, they can be represented locally by four-tuples. The lasers in our situation are finite lines (no lines at infinity), that are also never perpendicular to the Z-axis. We can achieve this representation by defining two parallel planes, each perpendicular to the Z-axis. Every line in our dataset intersects the two planes at specific 3D coordinates. The four-tuples are composed of the x- and y-coordinates of the two intersection points. Since the two planes are perpendicular to the Z-axis, all the z-coordinates of the corresponding intersection points are the same and do not add exploitable information to the model. They are discarded. We train four distinct Gaussian processes on these four-tuples. The resulting predictions of this trained model are intersection points on the original parallel planes. From those two points $\mathbf{p_1}$ and $\mathbf{p_2}$, the not normalised Plücker coordinates of a predicted line $\mathbf{L}$ are calculated by

$$\mathbf{L} = (\mathbf{p_2} - \mathbf{p_1}, \mathbf{p_1} \times (\mathbf{p_2} - \mathbf{p_1})) = (\mathbf{p_2} - \mathbf{p_1}, \mathbf{p_1} \times \mathbf{p_2}). \tag{3.2}$$

This fourth model also ensures that the predicted six-tuple is a straight line.

3.4 Results

We evaluated the models using three different methods. First, a cross-validation (CV) is performed on the set of lines. Second, we assess how well the models can predict a point in 3D space when a pair of input values are given. Third, we aim the laser beam at a specific point in 3D space.

The assumption of working with straight lines allows us to correct the measured points on the validation plane. We first determined the coordinates of the validation plane by performing a best fit on the measured points. Then we composed a set of validation points by intersecting this plane with the straight lines of the dataset. Our assumption is that intersection points are closer to the ground truth than the measured points. The corrected points are used in the evaluation method that predicts a point on a plane and the method that aims the laser beam at a given point. We do not use this corrected set to validate the purely data driven method, since in that case no assumptions of an underlying truth are made. The data driven method is only validated against the measured points. For the synthetic data, no correction is needed since the exact underlying truth is known. We acknowledge that this extra correction is up for debate. However, we feel justified in doing so because our initial assumption was that laser lines are straight lines in the real world. Moreover, the coordinates of the planes are still based on the original measurement values of the points. We leave the viability of this extra correction step to the assessment of the reader.

We performed the validations on every model with three different sizes for the dataset: 50 lines, 100 lines and either 226 or 217 for the real world and synthetic dataset respectively. The lines were sampled at random. We repeated the experiment ten times when using 50 lines and five times when using 100 lines. We trained the data-driven model with the same number of points on the validation to compare it with our models.

As already investigated in [162], data-driven calibration for galvanometric laser scanners outperform the mathematical model and look-up table-based calibration procedures. We restrict ourselves to comparing our methods only to the data-driven one.

3.4.1 Cross-validation

In a first evaluation method, we compare all models via 10-fold cross-validation. The data is split into ten blocks. The models are trained on nine of them. Predictions are made on the inputs from the remaining block and validated against the underlying ground truth. This process is repeated ten times, at every iteration with a different block for validation. This ensures that every straight line is included in the validation set exactly once. We compare the set of predicted lines to the set of underlying true lines. In the case of real world data, these are the best fit lines on the noisy measured points. In the case of synthetic generated data, these lines are fit on noiseless points and thus yield an accurate ground truth.

To measure the deviation between two straight lines, we follow the approach in [124], which describes the construction of a *line segment distance*. We do not consider the line itself, but only a finite oriented line segment with boundaries in a region of interest, i.e. values that are

	Real world data						Synthetic data					
	50		100		226		50		100		217	
	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]
6 GPs	1.939	1.355	1.709	1.009	1.695	1.327	5.819	15.725	2.655	3.143	1.266	0.684
Poinsot	1.927	1.329	1.693	0.985	1.687	1.315	5.822	15.950	2.614	3.018	1.250	0.661
Dir Point	1.874	1.266	1.589	0.927	1.626	1.297	4.918	12.908	2.300	2.381	1.263	0.668
2 Points	1.950	1.266	1.660	1.010	1.553	1.243	3.314	6.163	1.585	1.003	0.700	0.198

Table 3.1: The RMSE and variance σ^2 for the evaluation method that calculates the line segment distances. The data-driven model does not produce straight lines for cross-validation. The first row shows the number of data points used.

appropriate to the measured points of the dataset. Let $\mathbf{p_1}$ and $\mathbf{p_2}$ be the intersection points of lines $\mathbf{L_1}$ and $\mathbf{L_2}$ with the XY-plane. These lines also intersect with another plane, parallel to the xy-plane, at points $\mathbf{q_1}$ and $\mathbf{q_2}$. The distance d between the line segments is defined as

$$d^2 = (\mathbf{p_1} - \mathbf{p_2})^2 + (\mathbf{q_1} - \mathbf{q_2})^2 + (\mathbf{p_1} - \mathbf{p_2}) \cdot (\mathbf{q_1} - \mathbf{q_2}). \tag{3.3}$$

This distance can be seen as the aggregate of all distances of the pairwise points on the line segments integrated between the two planes. We used the 3D-sensor's reference frame, which is mounted directly under the vibrometer.

For the validation of our models, we are interested in the region of the validation plane. All measured points on that plane have z-coordinates that are well between 1.5 and 2 metres, making this a more than reasonable choice for the line segment boundaries. Here and elsewhere, we follow the notation used in [124] to describe homogeneous coordinates of points, lines and planes. The two intersection planes are described by the homogeneous plane coordinates $(-1.5, 0, 0, 1)$ and $(-2, 0, 0, 1)$.

For each model, we calculated the root mean squared error (RMSE) of these distances for all line segments. Exact numbers can be found in Table 3.1.

3.4.2 Predicting a Point on a Plane

In a second evaluation of the models, a pair of input values was given to predict the intersection point between the guided laser beam and a detection plane. To this end, we use the validation plane. Points from the dataset that lie on this plane were not used to train the models. The intersection point is calculated using Equation 3.1. The coordinates of the plane π are found by the best plane fit method explained in [90]. This evaluation method infers the 3D coordinates of a point when given a value for input parameters α and β (corresponding to a set of mirror rotations and thus a reflected laser beam) and the validation plane coordinates. For every line, we calculate the Euclidean distance between the ground truth point in the dataset and the by the model predicted point on the validation plane. This discrepancy serves as the error in the RMSE for every model.

We also benchmark our models to a purely data-driven approach as proposed in [162]. In order to do so, we trained three separate Gaussian processes on the 3D coordinates (one for each component). The same SE kernel with ARD (Equation 2.9) was used throughout. We performed a 10-fold CV on 50 points, 100 points and the full datasets, similarly to the CV method described above. The findings are summarised in Table 3.2.

	Real world data						Synthetic data					
	50		100		226		50		100		217	
	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]	RMSE [mm]	σ^2 [µm²]
6 GPs	1.278	0.515	1.054	0.321	0.963	0.258	4.135	9.068	1.725	0.793	1.523	0.552
Poinsot	1.275	0.511	1.047	0.314	0.961	0.256	4.017	8.284	1.735	0.807	1.526	0.555
Dir Point	1.274	0.517	1.042	0.308	0.958	0.242	3.302	4.599	1.761	0.818	1.566	0.568
2 Points	1.601	0.996	1.215	0.484	0.970	0.289	2.356	2.178	1.562	0.523	1.417	0.405
Data-driven	2.307	2.224	2.089	1.986	1.969	1.809	3.696	2.383	3.310	1.879	3.130	1.564

Table 3.2: The RMSE and variance σ^2 of the distances calculated between points on the validation plane and the predictions made by the models.

	Real world data						Synthetic data					
	50		100		226		50		100		217	
	RMSE [mV]	σ^2 [µV²]	RMSE [mV]	σ^2 [µV²]	RMSE [mV]	σ^2 [µV²]	RMSE [mdeg]	σ^2 [mdeg²]	RMSE [mdeg]	σ^2 [mdeg²]	RMSE [mdeg]	σ^2 [mdeg²]
6 GPs	7.582	18.536	6.180	10.990	5.664	8.903	65.104	2.344	26.380	0.189	22.943	0.127
Poinsot	7.579	18.492	6.162	10.869	5.664	8.895	62.439	2.053	26.330	0.188	22.922	0.126
Dir Point	7.579	18.812	6.119	10.548	5.664	8.354	50.476	1.083	27.019	0.200	23.674	0.130
2 Points	9.473	35.113	7.094	16.178	5.690	9.887	36.032	0.520	23.625	0.122	21.445	0.096
Data-driven	13.689	80.024	12.413	71.306	11.691	64.761	45.176	0.464	41.022	0.387	40.094	0.340

Table 3.3: The RMSE and variance σ^2 of the distances calculated between the pairs of input parameters of the points on the validation plane and the pairs of input parameters when aiming a laser beam towards those points.

3.4.3 Aiming at a Point on a Plane

As a third evaluation method, we use the opposite approach of the previous method that predicts the 3D coordinates of a point on the validation plane, when given some input values for α and β. Here, we are given the coordinates of the point and try to find the values for α and β that direct the laser beam to that point.

Aiming a laser beam accurately at a specific point in 3D space is an important application of galvanometric setups. In order to assess this aspect of the models, the points on the validation plane were used as targets. We minimise the distance between the point on the validation plane we are aiming at and the point predicted by varying the input parameters α and β. Our MATLAB R2020b implementation used `fminsearch`, a Nelder-Mead Simplex Method as described in [83]. We used MATLAB's default options for tolerances and stopping criteria.

We calculate the 2D Euclidean distance between the α and β values for the target points and the points for which a minimum deviation from the targets was found. This 2D distance is considered the error in the RMSE calculated for each model. For the real world data datasets, α and β are voltages. For the synthetic dataset, α and β are degrees. Again, for the purely data-driven one, we performed a 10-fold CV in the same manner as described above. Table 3.3 shows the results.

3.5 Discussion

When cross-validating the predicted lines, all models perform similarly on real world data. This is due to the fact that they are trained and validated on noisy data. The difference between the models is lost in the accuracy of the measurements. When adding more data points, all models perform better and the two intersection points model slightly outperforms the other ones. It is notable that increasing the amount of data points from 50 to 100 results in a change in accuracy that is larger than when increasing even further to 226 data points. This suggests that adding even more data points would result in marginal improvements, perhaps not worth the extra computing time as GPs have a $\mathcal{O}(n^3)$ complexity in the number of data points [130]. Of course this is highly dependant on the application itself.

The synthetic dataset does allow for validation against an exact ground truth. The results of the cross-validation of the line segment distances show that the two intersection points model reaches higher accuracy. The differences between the models decrease when adding more data points.

The difference between the six-GP-model and the Poinsot axis model is almost negligible for the real world dataset. However, there is a slight increase in accuracy on the synthetic dataset when predicting points on the validation plane or aiming at them. Both datasets can be considered dense, meaning there are relatively many data points in a small region. The straight lines are all pointed towards the front of the device and only vary slightly within small angle ranges for the mirrors. This results in line directions that, when represented as 3D points on a unit sphere, only cover a small fraction of its surface. The same reasoning holds for the moments. As a result, the norms and pitches of the predicted lines do not deviate significantly from their ideal values.

The synthetic data also shows that training on datasets that are relatively small, results in bad predictions for the 6 GP, Poinsot axis and direction intersection point model. These models are the most flexible because they are composed of more GPs. More data points are needed to outperform the data-driven approach. This is also the case for the evaluation method that aims a laser beam at a point on the validation plane.

Alternatively, one could implement a different data-driven method for aiming the laser beam at a given point in 3D space as follows. Two distinct GPs are trained on the dataset that maps the 3D coordinates to α and β respectively. Then a pair of angles could be inferred for every new input of 3D coordinates. So this method works the other way around. We did not include it in this work, because no comparison can be made between the location of the laser hitting the validation plane and the target points themselves. These are the same points, since the targets are the input for the two GPs. This alternative method does not suit our purposes here, since the explicit goal in validation was to compare those two sets of points for all models.

In this work, we solely focused on Gaussian processes, due to their inherent robustness against overfitting [130]. Earlier stated works make comparisons between different regression techniques for calibrating galvanometric setups [162]. We leave exploring the semi data-driven method implemented with other machine learning techniques, such as artificial neural networks, for future work.

An important benefit of working with Gaussian processes is that a predicted value is in fact a Gaussian distribution, i.e. determined by a mean and a variance. The latter can be interpreted as a measure of uncertainty accompanying the mean. In applications where risk assessment plays a vital role, such as predictive maintenance or medical treatments, this extra information about the

prediction is of great consequence.

To calculate the similarity between two pairs of input parameter values $a_1 = (\alpha_1, \beta_1)$ and $a_2 = (\alpha_2, \beta_2)$, we use the Euclidean distance. This distance is key in the covariance function. Clearly, this method is sub-optimal, as a_1 and a_2 are not points in $\mathbb{E}^2$. In the synthetic dataset, these are rotation angles of the two mirrors. The real-world dataset uses voltages, which are transformed linearly to angles in the galvanometers. In [53] these points are considered to be lying on the surface of a sphere so a spherical distance is implemented in the kernel. Alternatively, a_1 and a_2 can be interpreted a lying on a torus. However, the range of the angles in our dataset is very limited. In our case, the rays are all pointing in the forward direction of the (either real or virtual) device. They do not cover the whole hemisphere. Hence the difference between the spherical or circular distance to the Euclidean distance is negligible. In applications where the rays cover a significant part of the hemisphere or even the whole sphere, e.g. in LIDAR 3D scanning, this approximation would no longer hold. In this case, large differences in angles could result in small Euclidean distances. In the next chapter, we will address this issue in more detail.

3.6 Conclusion

Mathematical or physical calibration models can be cumbersome to implement and hard to find solutions for. Data-driven approaches don't exploit obvious facts about the underlying truth, bypassing any model completely. We have shown that our semi data-driven approach combines the best of both worlds. The assumption that rays leaving the galvanometric setup are straight lines, is more than reasonable and easy to work with. We proposed several methods to construct models that can learn to predict these straight lines, given the two input parameters that govern the rotation of the galvanometer mirrors. We have presented three scenarios to assess their performance and have demonstrated that they outperform the data-driven approach.

Chapter 4

Input Output Constraints

In Chapter 3, we introduced a semi-data-driven approach to calibrate a galvanometric setup. We also described the challenges of predicting Plücker coordinates for straight lines that follow the Grassmann-Plücker relationship and presented several construction methods to ensure a six-tuple with zero pitch. In this chapter we take that approach a step further. Both the inputs (the pairs of rotations of the two mirrors) and outputs (the straight lines) lie on a manifold. We can incorporate this prior knowledge in the model via constraints built in the formulation of a covariance function. We propose two constrained models: one in which a linear constraint on the direction vector is written as a differential equation and one in which a quadratic constraint is imposed by a reparametrisation of the line coordinates. We compare them to data-driven and unconstrained model-based approaches. We show that enforcing constraints improves the quality of the predictions significantly and thus the accuracy of the calibration. We validate our findings against real world data by predicting points on validation planes, calculating line segment distances, considering the training times for the models and assessing how much a predicted line resembles an actual straight line. We published the findings of this chapter in [36].

4.1 Introduction

The six components of the Plücker coordinate of a straight line are not correlated. However, they are not fully independent either. Information about one of the outputs does not imply knowledge about any of the other outputs. But together, the outputs do follow certain rules or constraints. As these Plücker coordinates are homogeneous, knowledge about all the outputs except one, does encode the underlying value for the missing one. For example, assume that a 3D direction vector is normalised to a length of one. Knowledge about one of the components does not reveal anything about the other two. There is no correlation between the components. However, knowing the values for the x- and y-component allows for the calculation of the z-component via $x^2+y^2+z^2=1$. The components are not independent. In the case of normalised Plücker coordinates, these constraints are given by Equation 2.12 and 2.13. The first equation is simply to normalise the direction of the line to unit length. The second equation is the Grassmann-Plücker relation, which states that the direction of a straight line is perpendicular to its moment.

We can exploit these constraints via covariance functions or kernels of our GPs. Over the past years, several methods for incorporating constraints in kernels have been formulated in [75, 136]. An elaborate overview, including methods beyond those two, is given in [151].

In this chapter, we investigate both the input and output space of the dataset, which consists of the mapping between input angles of the two rotation mirrors and the laser beams, represented by Plücker coordinates of a straight line. First, we observe that the input space is not Euclidean, but topologically a manifold that is a torus. This leads us to propose a periodic kernel in which we separate the two input dimensions (i.e. the two mirror rotations). Second, we argue that straight lines can be described by points in a five-dimensional projective space $\mathbb{P}^5$ that live on a four-dimensional manifold M_2^4. This is a quadric, which means its points satisfy a quadratic equation. To enforce this quadratic constraint on the predictions of our model, we propose two different approaches:

1. Reformulate the constraint as a linear differential equation
2. Parameterise the quadratic constraint and solve a matrix factorisation

Both approaches are achieved by rewriting the covariance function of the underlying GPs.

We propose two models, in which we incorporate prior knowledge about the data in the kernels. By doing so, we enforce constraints on the predictions that improve their quality. We compare these models to the mathematical model of [100], the data-driven model proposed in [162] and the semi-data-driven approach by [37]. All models are validated on real world data gathered with a scanning laser Doppler vibrometer (type: Polytec PSV-QTec).

The rest of this chapter is structured as follows. In the next section, we explain how we gathered the data. The third section describes the models we investigate. In a fourth section we present the results. In Section 4.5 we discuss these findings. Finally, conclusions are provided.

4.2 Gathering Data

For this chapter, we composed a new dataset consisting of straight lines. We used a Polytec PSV-QTec scanning laser vibrometer instead of the Polytec OFV 056 used in the previous chapter. Its laser beam is also controlled by two rotating mirrors. As described in the manufacturer's device manual, we first performed a 2D and 3D alignment to establish a reference coordinate system. We aimed the beam at a detection plane, on which we assigned a point that serves as the origin of the coordinate system. The Z-axis points from the origin towards the device horizontally. The Y-axis points upwards and the X-axis from left to right from the perspective of the device. Then we moved the plane in thirteen different positions and orientations. The 3D-coordinate of the intersection point of the laser with each detection plane was calculated by the built-in 3D scanning capabilities of the vibrometer itself. From the thirteen planes, we kept four aside as validation planes. Points on these planes were not used to train the models. A schematic overview of the setup is provided in Figure 3.2.

The data includes 11 values for the first mirror rotation angles and 21 for the second. This results in 231 laser beams. Collinear points, generated by a particular pair of mirror rotations and thus a single laser beam, were filtered via RANSAC [51]. Straight lines are fitted on the inlier points using the least squares method described in [90]. Points on any of the validation planes were not included. A plot of the detected points can be seen in Figure 4.1.

Furthermore, we took the same RANSAC and best fit approach to find the homogeneous coordinates of the planes themselves. We calculate the intersection point of the found plane and every

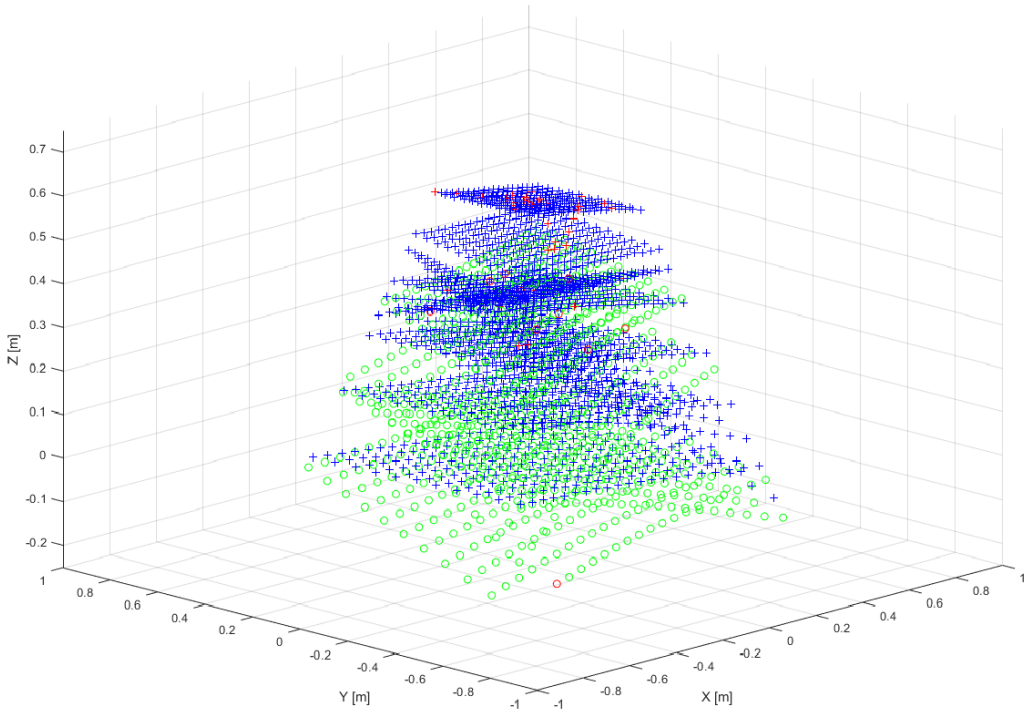


Figure 4.1: Points where the laser beam hit the detection plane, which was moved to thirteen different positions. Blue crosses are points that are used in the best fit approach to construct a dataset of lines. Green circles are points on the four validation planes. Red circles are outlier points that are removed from the planes.

best fit line. These intersection points are considered the corrected versions of the measured points and are used in the validation of models that have the assumption of straight lines. For the data-driven approach, which does not make such an assumption, this correction was not used.

In contrast to the previous chapter, we now only work with real world generated data. No synthetic data was used throughout. In the following sections, we explain how these measured intersection points of the laser with each detection plane can be exploited in a semi-data driven calibration method.

4.3 Models

The objective of the calibration of the galvanometric setup, is to find a mapping from the input parameters to the set of straight lines. In our case the input parameters are the rotation angles of the two mirrors. We will label them α for the first mirror and β for the second mirror. As stated in the previous section, straight lines can be described by a six-tuple of numbers. Thus the mapping has the form $f : (\alpha, \beta) \rightarrow (l_1, l_2, l_3, l_4, l_5, l_6)$. Since the aim is to find straight lines in Euclidean 3D-space, we want these six-tuples to satisfy Equation 2.12 and Equation 2.13. In other words, the output of the prediction should be a point on the manifold M_2^4. These are the constraints we

impose on the outputs of the GPs. We propose two models that incorporate these constraints and compare them to both mathematical and data-driven models.

4.3.1 Mathematical Model

In [100] a mathematical model was proposed to calibrate the galvanometric setup. This model consists of parameters that describe the distance between the mirrors, their position, the tilt angle between them, the position of the origin of the laser beam, the initial direction of the laser beam and the conversion parameters between the inputs and the actual angles of the mirrors. These values are found by minimising the difference between a set of lines generated by these parameter values and the set of measured lines. We followed the procedure described in the paper and will refer to this model as Mathematical. The initial guess for the parameters were based on measurements on the device itself.

4.3.2 Data-driven

We implemented a purely data-driven approach as explained in [162], by defining three GPs for every detection plane. Each GP was trained on either the x-, y- or z-component of the 3D-coordinates of the intersection points of the laser beam and the plane. The covariance matrices were constructed with the SE ARD kernel described in Equation 2.9.

4.3.3 Six Gaussian Processes

To learn the Plücker coordinates directly, we implement 6 distinct Gaussian processes, i.e. one for each output component in $(l_1, l_2, l_3, l_4, l_5, l_6)$. The kernel used, is the SE with ARD as described in Equation 2.9. To calculate the similarity between two pairs of input parameter values $\mathbf{x} = (\alpha, \beta)$ and $\mathbf{x}' = (\alpha', \beta')$, the Euclidean distance is used.

4.3.4 Six Gaussian Processes with Periodic Kernel

The usage of Euclidean distances in the kernel is sub-optimal, as $\mathbf{x}$ and $\mathbf{x}'$ are not points in $\mathbb{E}^2$. They are pairs of rotation angles. Both input components live in a space that is periodically curved on itself. From a topological point of view, the input space can be interpreted as the Cartesian product of two circles $\mathbb{S}^1 \times \mathbb{S}^1$, which is homeomorphic to the torus $\mathbb{T}^2$. As a first improvement on the kernel from the previous model, we implement a periodic kernel (PER), proposed by [99], which extended with ARD and a period of 2π for both α and β, has the from

$$\begin{aligned} k_{PER}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{2}{l_\alpha^2}\sin^2\left(\frac{|\alpha - \alpha'|}{2}\right)\right) \\ \cdot \exp\left(-\frac{2}{l_\beta^2}\sin^2\left(\frac{|\beta - \beta'|}{2}\right)\right), \end{aligned} \tag{4.1}$$

in which σ_f^2 is the height-scale and l_α^2 and l_β^2 are the length-scales for each input dimension. In this kernel, the inputs are periodic and the dimensions are separated from each other.

4.3.5 Direction and Point on a Plane

As described in the previous chapter, the Grassmann-Plücker relation can be enforced by construction. Instead of training a GP on each of the six components of the Plücker coordinates, we train on the directions of the straight lines and on the intersection points of those lines with a known plane parallel to the XY-plane. From these intersection points and the directions, the moments of the lines can be calculated via Equation 2.10. This ensures zero-pitch lines according to Equation 2.3.2, since the calculated moment is always perpendicular to the direction. For this approach three GPs are needed for the three components of the direction vectors and two GPs for the 3D-coordinates of the intersection with the plane (the z-component is fixed). We refer to this model as Dir IPP.

4.3.6 Linear Constraint on the Direction

The three GPs in the model proposed above, whose predictions together form the direction vectors for the straight lines, must follow Equation 2.12, i.e. their norms must equal one. To put this another way, all these direction vectors can be seen as points on a unit sphere $\mathbb{S}^2$. This allows for the implementation of a constraint: the predicted output (a three-vector) should always lie on a unit sphere. To achieve this constraint, we perform two steps:

First, we incorporate the index of the three output dimensions as an extra input. This strategy is explained in [5]. Another elaborate explanation on multi-output Gaussian processes can be found in [95]. The input of the GP can be written as $\mathbf{x} = (\alpha, \beta, d)$, in which d is the index of the output dimension. In our case, this is either one, two or three. This results in three times as many data points. The output of the GP is still a scalar, but we can combine three GP-predictions, each for a different index of output dimension, in such a way that we get a three-vector. The kernel in this model is extended to take in an extra input. For independent outputs, the covariances for inputs that do not have the same index for their output dimension, are zero. When inputs are of the same output dimension, the kernel k_{PER} in Equation 4.1 is used. This leads to a 3×3 block covariance matrix. The altered kernel now has the form

$$k_{PER3D}(\mathbf{x}, \mathbf{x}') = \begin{cases} k_{PER}([\alpha, \beta], [\alpha', \beta']), & \text{for } d = d' \\ 0, & \text{for } d \neq d'. \end{cases} \tag{4.2}$$

Second, we adjust the kernel following a method described in [75]. The authors describe a way to have linear constraints be built in the kernel function itself. An example of those would be a linear differential equation. We refer to their supplementary material for a full description. Here, we explain how this can be implemented in our context. Even though our constraint itself is quadratic in nature (outputs of the GPs describe points on the surface of a unit sphere $x^2 + y^2 + z^2 = 1$), we can rewrite this as a linear differential equation.

After a reparametrisation from the two input angles to three Cartesian coordinates for points on the unit sphere, following the convention depicted in Figure 4.2, we can write

$$f_1 : x = \cos(\alpha)\sin(\beta), \tag{4.3}$$
$$f_2 : y = \sin(\alpha)\sin(\beta), \tag{4.4}$$
$$f_3 : z = \cos(\beta). \tag{4.5}$$

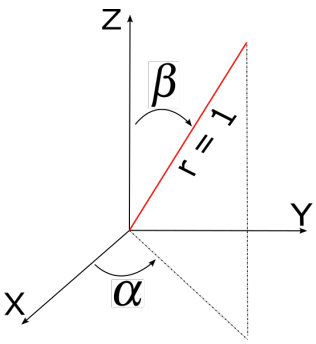


Figure 4.2: Convention for the parametrisation.

This allows us to formulate the following linear constraint

$$f_1 - \frac{\partial f_2}{\partial \alpha} + \frac{\partial f_3}{\partial \alpha} = 0. \tag{4.6}$$

This can be written more compactly as

$$\mathscr{F}_{\mathbf{x}}[\mathbf{f}(\mathbf{x})] = \mathbf{0}, \tag{4.7}$$

in which $\mathbf{x} = [\alpha, \beta]^T$, $\mathbf{f} = [f_1, f_2, f_3]^T$ and $\mathscr{F}_{\mathbf{x}}$ is a linear operator mapping the function $\mathbf{f}(\mathbf{x})$ to another function $\mathbf{g}(\mathbf{x})$. We can also relate $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ via another linear operator $\mathscr{G}_{\mathbf{x}}$ such that

$$\mathscr{G}_{\mathbf{x}}[\mathbf{g}(\mathbf{x})] = \mathbf{f}(\mathbf{x}). \tag{4.8}$$

The constraint in Equation 4.6 can now be written as

$$\mathscr{F}_{\mathbf{x}}[\mathscr{G}_{\mathbf{x}}[\mathbf{g}(\mathbf{x})] = \mathbf{0}. \tag{4.9}$$

Reformulating the linear operators as matrix-vector multiplications yields

$$\mathscr{F}_{\mathbf{x}}\mathscr{G}_{\mathbf{x}} = \mathbf{0}. \tag{4.10}$$

This reformulation allows us to impose the constraint on the operator $\mathscr{G}_{\mathbf{x}}$ instead of on the GP for $\mathbf{f}(\mathbf{x})$. With $\mathbf{g}(\mathbf{x})$ sampled from a GP with mean $\mu_g(\mathbf{x})$ and covariance $K_{\mathbf{g}}(\mathbf{x}, \mathbf{x}')$, this means that

$$\mathbf{f}(\mathbf{x}) = \mathscr{G}_{\mathbf{x}}\mathbf{g} \sim \mathcal{GP}(\mathscr{G}_{\mathbf{x}}\mu_{\mathbf{g}}, \mathscr{G}_{\mathbf{x}}K_{\mathbf{g}}\mathscr{G}_{\mathbf{x}'}^T). \tag{4.11}$$

A complete explanation on how to find $\mathscr{G}_{\mathbf{x}}$ for any given $\mathscr{F}_{\mathbf{x}}$ using the nullspace of the latter, can be found in [75]. In our case, we find

$$\mathscr{F}_{\mathbf{x}} = \left[1, -\frac{\partial}{\partial \alpha}, \frac{\partial}{\partial \alpha}\right], \tag{4.12}$$

$$\mathscr{G}_{\mathbf{x}} = \left[\frac{\partial}{\partial \alpha}, 1, 0\right]^T, \tag{4.13}$$

which leads to

$$\mathscr{G}_{\mathbf{x}}\mathscr{G}_{\mathbf{x}'}^T = \begin{bmatrix} \frac{\partial^2}{\partial \alpha \partial \alpha'} & \frac{\partial}{\partial \alpha} & 0 \\ \frac{\partial}{\partial \alpha'} & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.14}$$

Operator $\mathscr{G}_{\mathbf{x}}\mathscr{G}_{\mathbf{x}'}^T$ can only be applied on functions that are at least twice differentiable. To that end, we need to combine the two cases in the covariance function in Equation 4.2. Furthermore, the absolute value signs can be dropped, since the sine function is an odd function which is then squared: $\sin^2(-\theta) = \sin^2(\theta)$. We propose the formulation

$$\begin{aligned} k_{COM}(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp\left(-\frac{2}{l_\alpha^2}\sin^2\left(\frac{\alpha-\alpha'}{2}\right)\right) \\ \cdot \exp\left(-\frac{2}{l_\beta^2}\sin^2\left(\frac{\beta-\beta'}{2}\right)\right) \\ \cdot \exp\left(-\frac{(d-d')^2}{\varepsilon^2}\right), \end{aligned} \tag{4.15}$$

in which ε is an arbitrary small value. This results in the last factor to be either one for equal dimension index d or (very close to) zero for different dimensions. Applying Equation 4.14 on Equation 4.15 yields a covariance function

$$k_{DIF} = \mathscr{G}_{\mathbf{x}} k_{COM} \mathscr{G}_{\mathbf{x}'}^T = \begin{bmatrix} A & B & 0 \\ C & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} k_{COM}, \tag{4.16}$$

in which, with $\gamma = \frac{\alpha-\alpha'}{2}$ (for brevity in notation),

$$A = -\frac{\sin^2(\gamma)\cos^2(\gamma)}{l_\alpha^4} + \frac{\cos^2(\gamma)}{l_\alpha^2} - \frac{\sin^2(\gamma)}{l_\alpha^2}, \tag{4.17}$$

$$B = -\frac{2}{l_\alpha^2}\sin(\gamma)\cos(\gamma), \tag{4.18}$$

$$C = \frac{2}{l_\alpha^2}\sin(\gamma)\cos(\gamma). \tag{4.19}$$

This describes a covariance function that imposes the quadratic constraint, written as a linear differential equation, which states that the norm of the 3D vector predicted by the GP should always be equal to one. We apply this for the prediction of the direction of the line. We refer to this model as DirCon IPP.

4.3.7 One Gaussian Process with Quadratic Constraints

In this section, we derive an alternative kernel to the one proposed above. The aim is to impose the constraints in Equation 2.12 and Equation 2.13 simultaneously. We follow the method described in [136] by applying the following steps.

First, let $\mathbf{z} = (l_1, l_2, l_3, l_4, l_5, l_6)$ and $\mathbf{Q} = \mathbf{z}^T\mathbf{z}$, which encodes all the quadratic terms. This is a 6×6 symmetric matrix and thus it is fully determined by its upper triangular part. We now formulate a training point $\mathbf{y} \in \mathbb{R}^{21}$ as the concatenation of those upper triangular elements

$$\mathbf{y} = [\mathbf{Q}_{11}, ..., \mathbf{Q}_{ij}, ..., \mathbf{Q}_{66}]^T, \text{with } i \leq j. \tag{4.20}$$

Second, by incorporating the index of the output dimension as an input, we apply the same method as explained above to construct a multi-output GP. Again, the input of the GP can be written as $\mathbf{x} = (\alpha, \beta, d)$, in which d is the index of the output dimension. Here d ranges from 1 to 21 and

thus the dataset is twenty-one times larger. The covariance function for this model is given by Equation 4.2.

As stated and proved in [136], a set of training examples $\{\mathbf{y}_1, ..., \mathbf{y}_n\}$ that satisfy the linear constraints $\mathbf{A}\mathbf{y}_i = \mathbf{b}$ will result in a GP for which the mean prediction $\mu(\mathbf{x}_*)$ also satisfies $\mathbf{A}\mu(\mathbf{x}_*) = \mathbf{b}$. The authors of [136] demonstrate their findings on a similar challenge, namely a rotation (quaternion) of unit norm. In our case, $\mathbf{A}$ is a 2×21 matrix and $\mathbf{b} = [1,0]^T$.

However, we are interested in predictions in the form of the six dimensional $\mathbf{z}_*$ and not the twenty-one dimensional $\mathbf{y}_*$. Generally, this is achieved by casting this problem to a matrix factorisation problem and minimising the Frobenius norm between the factorisation and the output of the GP. The solution can be obtained in closed form as follows. In our case, we first compose the matrix

$$\mathbf{Q}_* = \begin{bmatrix} y_{*1} & y_{*2} & \cdots & y_{*6} \\ y_{*2} & y_{*7} & \cdots & y_{*11} \\ \vdots & \vdots & \vdots & \vdots \\ y_{*6} & y_{*11} & \cdots & y_{*21} \end{bmatrix} = \mathbf{V}\Sigma\mathbf{V}^T, \tag{4.21}$$

in which $\mathbf{V}$ is a one column matrix and Σ is just a scalar. We then calculate

$$\mathbf{z}_* = \sqrt{\Sigma}\mathbf{V}^T. \tag{4.22}$$

The solution to the factorisation is not unique. There is a sign ambiguity that arises when taking the square root. As proposed in [136], this can be solved by taking into account information from the context. In our, case we know that the z-component of the direction vector of the predicted lines should be negative. If this is not the case, the line points in the opposite direction and the Plücker coordinate can be multiplied by -1 to reverse the signs of its six components.

4.4 Results

We compare all models via 5-fold cross-validation (CV). The data described above is split into five blocks. The models are trained on four of them. Predictions are made on the inputs from the remaining block and validated against the blocks used for training. This process is repeated five times, at every iteration with a different block for validation. This ensures that every block is included in the validation set exactly once. We trained all models on four different sizes for the dataset: 50, 100, 150 and 200 lines. We reran the training ten times for each size. To assess the quality of the models, four evaluation methods are considered.

First, for every input pair, we predict points on the validation planes. The data-driven model predicts those via three GPs (one for each 3D-coordinate component), while the other models calculate an intersection point between a predicted line and each validation plane. To assess the models, the root mean squared error (RMSE) of the Euclidean distance between the predicted points and the measured points (after correction as described in the Gathering data section) is calculated. An overview can be found in Table 4.1. This validation method assesses the practical validity of our models. In a real world scenario, one would be interested in where the laser beam exactly hits an object under inspection. Working with a correct set of predefined inputs (the mirror rotations) is of utmost importance when accuracy is being pursued.

Methods	n = 50	n = 100	n = 150	n = 200
Mathematical	21221.57	19410.13	19037.90	17599.49
Data-driven	12538.62	12585.36	10733.94	6078.32
6 GP	35.09	22.38	117.95	27.25
6 GP PER	8.79	1.22	0.46	0.26
Dir IPP	6.71	1.68	1.47	0.61
DirCon IPP	5.36	1.68	1.37	0.62
QCon	15.84	1.17	0.42	0.27

Table 4.1: The RMSE in [µm] of the distances calculated between points on the validation planes and the predictions made by the models.

Methods	n = 50	n = 100	n = 150	n = 200
Mathematical	35207.23	32456.77	31800.85	29527.41
Data-driven	NA	NA	NA	NA
6 GP	51.46	86.04	284.63	51.00
6 GP PER	13.00	2.21	0.82	0.44
Dir IPP	11.33	2.85	0.87	1.01
DirCon IPP	8.93	2.82	0.83	1.00
QCon	22.34	1.96	0.70	0.43

Table 4.2: The RMSE in [µm] of the line segment distances in µm between the measured lines and the lines predicted by the models. The data-driven model does not produce lines.

Second, for the line-based models, the distance between the predicted line and the measured line is calculated as described in [124]. We do not consider the line itself, but only a finite oriented line segment with boundaries in a region of interest, i.e. values that are appropriate to the measured points of the dataset. Let $\mathbf{p_1}$ and $\mathbf{p_2}$ be the intersection points of lines $\mathbf{L_1}$ and $\mathbf{L_2}$ with the xy-plane. These lines also intersect another plane, parallel to the xy-plane, at points $\mathbf{q_1}$ and $\mathbf{q_2}$. The distance d between the line segments is defined as

$$d^2 = (\mathbf{p_1} - \mathbf{p_2})^2 + (\mathbf{q_1} - \mathbf{q_2})^2 + (\mathbf{p_1} - \mathbf{p_2}) \cdot (\mathbf{q_1} - \mathbf{q_2}). \tag{4.23}$$

This distance serves as the error in the RMSE of the line segment distances in Table 4.2.

Third, we record the total time it takes to train a model for all cross-validation iterations and all reruns. The results can be found in Table 4.3. Finally, in a fourth evaluation method, we consider the pitch of the predicted line for the line-based models. These findings are summarised in Table 4.4.

4.5 Discussion

As thoroughly discussed in [162], the mathematical model is based on a hard to solve non-convex high-dimensional optimisation problem. The procedure requires a qualitative initial guess. Moreover, several time-consuming reruns are needed to find an optimal solution. The amount of training time highly depends on the stopping criteria for the optimisation process. For every laser beam, this model calculates its intersection point with the second mirror and a direction. As such,

Methods	n = 50	n = 100	n = 150	n = 200
Mathematical	566.52	1193.43	1713.65	2351.08
Data-driven	66.50	91.43	174.89	265.00
6 GP	108.59	191.22	303.26	529.85
6 GP PER	14.35	41.73	93.16	160.16
Dir IPP	10.89	29.86	68.59	122.64
DirCon IPP	29.69	104.68	214.90	370.89
QCon	642.56	2652.05	3612.35	3744.28

Table 4.3: The time in [s] it took to train each model. These are the accumulations of the times for each cross-validation iteration and each rerun.

Methods	n = 50	n = 100	n = 150	n = 200
Mathematical	0	0	0	0
Data-driven	NA	NA	NA	NA
6 GP	5.23	34.49	58.02	5.17
6 GP PER	0.89	0.26	0.13	0.05
Dir IPP	0	0	0	0
DirCon IPP	0	0	0	0
QCon	0.53	0.05	0.02	0.01

Table 4.4: The RMSE in [µm] of the pitches of the lines predicted by the models. The data-driven model does not produce lines. The mathematical model, the Dir IPP and DirCon IPP model produce lines with a pitch that is exactly zero, apart from machine accuracy.

the pitch of the generated line is always zero, even though this does not result in more accurate predictions.

The data-driven approach is free from the optimisation problem of the mathematical model, as the model is bypassed completely. However, removing all assumptions about the underlying truth was shown to be too restrictive in [37]. The semi-data-driven 6 GP model, in which six fully independent GPs are implemented, performs better than the mathematical or data-driven approach. However, training six GPs instead of three takes twice as long.

The most notable improvement is the introduction of a kernel in the 6 GP PER model that takes into account that the inputs are not points in Euclidean space, but live on a torus. Even though the kernel is slightly more complex, the actual training time for this model is lower. The training of a GP implies finding a set of hyperparameters that maximises the log marginal likelihood [130]. This process converges sooner for a kernel that describes the underlying reality or dataset better. Furthermore, we exploit the fact that the period for both dimensions is exactly 2π. These hyperparameters do not have to be learned during the training of the GPs.

In contrast to the data-driven, the 6 GP and the 6 GP PER model, the Dir IPP model produces lines that do fulfil the Grassmann-Plücker. This is achieved by construction. For the smallest dataset, this approach resulted in better predictions. For larger datasets, the gain in accuracy in comparison to the other models vanishes. Furthermore, this model requires only five GPs, which take less time to train than the six GPs of the 6 GP PER model.

The model with the linear differential equation constraint, also always produces lines with a

pitch of zero. Moreover, the constraint on the norm of the direction of the line, as described in Equation 2.12, yields predictions that are slightly better than those of the models without the constraint. This effect diminishes as the datasets grow. Since these datasets have lines with a unit norm for their direction vectors, the models all produce lines with a direction vector with a norm close to one. This constraint only has an effect on a minor defect in the prediction. Thus making the overall gain from this constraint minimal. Only when the dataset is relatively small, and the predicted directions tend to deviate from points on the unit sphere does this model outperform the other ones.

The model with the quadratic reparametrisation is trained on both constraints simultaneously. Therefore, it finds an optimum that satisfies both. This results in sacrificing one constraint in favour of the other one. When trained on the smallest dataset, it is outperformed by the other models. When trained on bigger datasets, it outperforms the other models significantly.

In order to construct multi-output GPs, the index of the output dimension is transformed into an input variable. The new set of outputs are the components of the original dataset stacked in a single column of scalars. For the model DirCon IPP, which is based on the differential equation, this yields a dataset that has three times as many entries. The dataset for the model QCon, with the quadratic reparametrisation, is composed of twenty-one times more entries. This is an important point of concern. With n the number of data points in a dataset, the standard complexity of the Gaussian process is $\mathcal{O}(n^3)$ for computation and $\mathcal{O}(n^2)$ for storage [161]. Herein lies the biggest trade-off of the models proposed. There is a gain in quality of the predictions, but at a significant time penalty.

In the last two decades, several approaches have been proposed to overcome the hurdle of large datasets for Gaussian processes. An overview can be found in [161]. This would be a fruitful area for further work, especially for the QCon model. Furthermore, in this work, we made no assumption about the relationship between the straight lines. The implemented covariance functions assume smoothness in the Plücker coordinates when varying the rotation angles of the mirrors. However, this is still a naive approach that can be taken further by incorporating the relationship between the straight lines into the model. Although, when doing so, the model becomes more complex and less data-driven, which was one of the strong shoots of the approach taken. More research is needed to assess where the optimum lies of the hybrid mix between fully data-driven and purely mathematical models.

We also implemented the methods of this and the previous chapter in the work of Penne et al. [115], where we compared these findings to the mathematical calibration methods based on line calculus. In that work, Penne introduced one more geometrical assumption. Not only are the laser lines considered straights lines, they are additionally constrained to a set of lines we dubbed the *2-mirror congruence*. In summary, the Gaussian process method was inferior to the line calculus methods for small grid sizes and for limited point measure noise levels. Only when the noise level is represented by a standard deviation of 8 mm or more and if a basegrid is used of size at least 8×8 lines, then the GP-method performs more accurately and more precisely. This is due to the fact that a Gaussian process filters out the noise by fitting smooth functions.

4.6 Conclusion

We presented two constrained models that, given a pair of rotation angles for the mirrors of a galvanometric setup, predict a straight line. Improving the kernel to take into account the manifolds on which the inputs and outputs live, leads to better predictions. This is of great importance for everyone who works with setups that require an accurate calibration to guide a laser beam.

However, this comes with a considerable extra computational time for the training of the GP(s), as the kernels become more complex and the datasets are re-parameterised into larger volumes. Even though this increase in training time is only an issue during the calibration process itself. It is not an issue at inference time when predictions are being made.

By means of cross-validation, we investigated our models on real world data in order to improve the calibration of a Polytec PSV-QTec scanning laser Doppler vibrometer. After calibration, we investigated the accuracy of the predicted laser beams and points those beams hit in a real world scenario.

Part II

Surface Approximation

Chapter 5

Surface Approximation by means of GPLVM

In this chapter we move away from galvanometric setup calibration and focus on another 3D measurement aspect, namely the surface that can be constructed by points obtained by a 3D scanner. In order to do so, we reformulate the point cloud as a set of normal lines, each with a corresponding point of the point cloud on them. The resulting *lines with a point on them* are called *line elements*. The close relationship between spatial kinematics and line geometry has been proven to be fruitful in surface detection and reconstruction. However, methods based on this approach are limited to simple geometric shapes that can be formulated as a linear subspace of line or line element space. The core of this approach is a principal component formulation to find a best fit approximant to a possibly noisy or impartial surface given as an unordered set of points or point cloud. We expand on this by introducing the Gaussian process latent variable model, a probabilistic non-linear non-parametric dimensionality reduction approach following the Bayesian paradigm. This allows us to find structure in a lower dimensional latent space for the surfaces of interest. We show how this can be applied in surface approximation and unsupervised segmentation to the surfaces mentioned above and demonstrate its benefits on surfaces that deviate from these. Experiments are conducted on synthetic and real world objects. We published the findings of this chapter in [32].

5.1 Introduction

Extracting structural information as shapes or surfaces from an unordered set of 3D coordinates (point cloud) has been an important topic in computer vision [64]. It is a crucial part of many applications such as autonomous driving [24], scene understanding [11], reverse engineering of geometric models [156], quality control [4], simultaneous localisation and mapping (SLAM) [146] and matching point clouds to CAD models [3]. Over the last decade, hardware developments have made the acquisition of those point clouds more affordable. As the availability, ease of use and hence the popularity of various 3D sensors increases, so does the need for methods to interpret the data they generate.

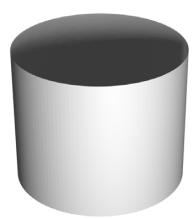
However, in this chapter, we mainly focus on detecting simple geometrical surfaces such as planes, spheres, cylinders, cones, spiral and helical surfaces, surfaces of revolution, etc. as described in [119]. Examples of these surfaces can be found in Figure 5.1. In [119], the close

relation between these shapes, spatial kinematics and line geometry is formulated. A point cloud, as a set of noisy points on a surface, is transformed into a set of normals (also referred to as normal lines or normal vectors) that show exploitable properties for that surface. For instance, the normals of a sphere intersect in a single point, the normals of a surface of revolution intersect in an axis of rotation and the normals of a helical surface can be seen as path normals of a helical motion. These insights led to applications in surface reconstruction and robotics [120, 119]. Later, their method was refined in [89, 125] to address pipe surfaces, profile surfaces and developable surfaces in general. In [121], the authors introduced principal component analysis (PCA) to approximate the set of normals. This laid the groundwork for a more general approach in [69] using so called *line elements*. These are constructed for every point of the point cloud. They are formed by the (Plücker) coordinates of the normal line and the surface point which lies on that line. The key insight of their work, is that the line elements of simple geometric surfaces lie on a linear subspace in $\mathbb{R}^7$, which can be found by solving an ordinary eigenvalue problem. We elaborate more on this approach in Section 5.2.2.

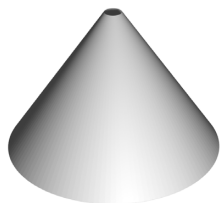
Although a mathematically very elegant way to describe 3D surfaces, this approach does have several drawbacks. First, the surface classification is strict. This means that only very primitive shapes can be treated. Real world objects do not always fall neatly into one of these categories. For example, imperfect shapes like a slightly bent plane, a sphere with a dent or shapes found in nature or human anatomy. Blindly following this method results in a misrepresentation of the data. Second, because PCA minimises an L2-norm, it is very sensitive to outliers. This can be mitigated by iterative RANSAC or by down-weighting the outliers. However, this comes at the cost of increasing computation time. Third, real world point clouds show various other imperfections like non-uniform sampling, noise, missing regions, etc. This highly affects the quality of the representation. Fourth, the authors of [89, 125, 121, 69] propose to look for *small* eigenvalues. The obvious question arises: when is an eigenvalue small? Even though some guidelines are provided, these threshold values remain domain specific and are challenging to set.

Most of these drawbacks can be attributed to the eigenvalue problem (or its PCA formulation) used to find an appropriate linear subspace in $\mathbb{R}^7$. In essence, this is a linear dimensionality reduction from the seven dimensional space of line elements to a lower dimensional latent space. In this work, we build on that method by introducing the Gaussian process latent variable model (GPLVM) [86] as an alternative. This allows for a non-linear relationship between a latent space and a higher dimensional data space, where observations are made. We implement a multi-output Gaussian process (seven outputs in this case) and try to find a mapping from a lower dimensional latent space to the line elements. Several variants on this theme exist, which we explain in more depth in Section 5.2.4. No longer confined by the linearity of PCA, our models can handle a wider range of shapes. Moreover, Gaussian processes handle noise very well, even in low data regimes [130]. The GPLVM places a Gaussian process prior over the mapping from the latent to the observed space. By exploiting strong priors we can significantly reduce the amount of data needed for equally accurate predictions.

In our approach, we can handle shapes (or surfaces) that fall in the categories described in [120, 119, 89, 125, 121, 69] but also shapes that significantly deviate from these. For instance, a surface of revolution whose central axis is not a straight line or an imperfect plane with one of the corners slightly bent. In fact, we drop the strict classification and allow for shapes that can be seen as somewhere in between the categories. This makes our methods more appropriate for handling surfaces that can be found in nature or when modelling the human body. Moreover, our formulation can handle multiple types of subsurfaces at once. This means we can perform segmentation in the latent space.



(a) Cylinder of revolution



(b) Cone of revolution

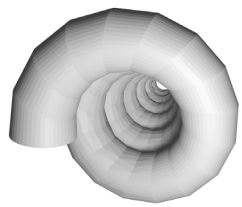
(c) Spiral cylinder

(d) Cylinder without revolution

(e) Cone without revolution

(f) Surface of revolution

(g) Helical surface



(h) Spiral surface

Figure 5.1: Examples of equiform kinematic surfaces

For completeness, we mention that in the recent years various deep learning techniques have been successfully introduced to discover objects in point clouds. A thorough overview can be found in [12] and more recently in [62, 64]. These techniques vary from ordinary multi-layer perception models (MLP) to convolutional- and graph-based models. Numerous datasets have been made public to further encourage the field to develop new models (e.g.: ScanObjectNN [154], ShapeNet [21], ScanNet [27], KITTI Vision Benchmark Suite [56], ModelNet40 [148, 132], ...). Generally, these data driven models are trained on more complex shapes: vehicles, urban objects, furniture, ... These models are specifically designed for detecting obstacles such as vehicles in autonomous driving, but not so for accurate object reconstruction from detailed 3D scanning. In this work, we focus on the latter. Moreover, whenever a new shape has to be learned, the underlying model has to be trained again.

To summarise, for every point on a given point cloud, we can formulate a so called line element. Dimensionality reduction on the set of line elements reveals the characteristics of the surface

captured by that point cloud. Existing methods rely on PCA, which is a linear mapping. In contrast, our model is built on the Gaussian process latent variable model, which allows for a non-linear mapping. This results in a more nuanced way to represent the surface. The main contributions of this work are the following:

- We expand existing methods based on kinematic surfaces and line element geometry by introducing GPLVM to describe surfaces in a non-linear way.
- We apply our method to surface approximation.
- We test our method to perform unsupervised surface segmentation.
- We demonstrate our method to perform surface denoising.

All of our 3D models, sets of line elements, trained GPLVM, notebooks with code and many more experiments and plots can be found on our GitHub repository[1].

The rest of this chapter is structured as follows. In the next section we give some theoretical background on line element geometry, kinematic surfaces, approximating the data, Gaussian processes and Gaussian process latent variable models in particular. The third section describes the results of our method applied to surface approximation, surface segmentation and surface denoising. Section 5.4 presents a discussion on our findings.

5.2 Materials and Methods

5.2.1 Line Elements

Plücker coordinates of a line in $\mathbb{E}^3$ can be extended to *line elements* by adding a specific point x on that line [69]. Neither this line or this point are at infinity. In order to do so, we start by choosing an orientation for the unit direction vector $\mathbf{l}$ of the line. A seventh coordinate λ is needed to locate x on that line, which can be defined as

$$\lambda = \mathbf{x} \cdot \mathbf{l}. \tag{5.1}$$

Notice that the norm of $\mathbf{l}$ matters, which is why we work with the (normalised) unit direction vector. This yields the seven-tuple $(\mathbf{l}, \bar{\mathbf{l}}, \lambda)$ of coordinates for a line element based on a line and a point, in which $\|\mathbf{l}\| = 1$ and $\mathbf{l} \cdot \bar{\mathbf{l}} = 0$.

Each point on a smooth surface Φ of a 3D volume has an outward unit normal vector $\mathbf{n}$. For every point x on that surface, a line element can be defined as $(\mathbf{n}, \mathbf{x} \times \mathbf{n}, \mathbf{x} \cdot \mathbf{n})$. These line elements constitute an associated surface $\Gamma(\Phi)$ in $\mathbb{R}^7$. An important property of many simple geometrical shapes in $\mathbb{R}^3$ (planes, spheres, cones, ...), is that their $\Gamma(\Phi)$ is contained in a linear subspace of $\mathbb{R}^7$. We will see in Section 5.2.2 that this aspect can be exploited in surface approximation, surface segmentation and surface denoising.

[1] `https://github.com/IvanDeBoi/Surface-Approximation-GPLVM-Line-Geometry`

5.2.2 Kinematic Surfaces

Rigid body motions can be seen as a superposition of rotations and translations. These can be extended by adding a scaling, making them the family of *equiform motions*, also known as *similarities* [89]. Such a one-parameter motion $M(t)$ is either a rotation, translation, a central similarity, a spiral motion or a combination of any of them. The velocity vector field of $M(t)$ is constant (time independent) and can be written as

$$\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \gamma\mathbf{x} + \mathbf{c} \times \mathbf{x}, \tag{5.2}$$

where $\bar{\mathbf{c}}$, $\gamma\mathbf{x}$ and $\mathbf{c} \times \mathbf{x}$ are the translation, scale and rotation component of the velocity vector $\mathbf{v}$ at a point x. A curve undergoing an equiform motion forms an *equiform kinematic surface*.

As defined in [69], a *linear complex of line elements* is the set of line elements whose coordinates $(\mathbf{l}, \bar{\mathbf{l}}, \lambda)$ satisfy the linear equation

$$\bar{\mathbf{c}} \cdot \mathbf{l} + \mathbf{c} \cdot \bar{\mathbf{l}} + \gamma\lambda = 0, \tag{5.3}$$

where $(\mathbf{c}, \bar{\mathbf{c}}, \gamma)$ is de coordinate vector of the complex. The following theorem from [109] shows the relation between linear complexes of lines and equiform kinematics:

Theorem 1 *The surface normal elements of a regular C^1 surface in $\mathbb{R}^3$ are contained in a linear line element complex with coordinates $(\mathbf{c}, \bar{\mathbf{c}}, \gamma)$ if and only if the surface is part of an equiform kinematic surface. In that case, the uniform equiform motion has the velocity vector field as given in Equation 5.2.*

Here, we will give an overview of such motions $M(t)$ and their corresponding surfaces Φ. For a thorough explanation on these (and multiple applications), we refer the reader to the works [123, 119, 89, 125, 121, 69, 109].

- $\gamma = 0$:
  - $\mathbf{c} = 0, \bar{\mathbf{c}} = 0$: $M(t)$ is the identical motion (no motion at all).
  - $\mathbf{c} = 0, \bar{\mathbf{c}} \neq 0$: $M(t)$ is a translation along $\bar{\mathbf{c}}$ and Φ is a cylinder (not necessarily of revolution).
  - $\mathbf{c} \neq 0, \mathbf{c} \cdot \bar{\mathbf{c}} = 0$: $M(t)$ is a rotation about an axis parallel to $\mathbf{c}$ and Φ is a surface of revolution.
  - $\mathbf{c} \neq 0, \mathbf{c} \cdot \bar{\mathbf{c}} \neq 0$: $M(t)$ is a helical motion about an axis parallel to $\mathbf{c}$ and Φ is a helical surface.
- $\gamma \neq 0$:
  - $\mathbf{c} \neq 0$: $M(t)$ is a spiral motion and Φ is a spiral surface.
  - $\mathbf{c} = 0$: $M(t)$ is a central similarity and Φ is a conical surface (not necessarily of revolution).

Examples of these surfaces can be found in Figure 5.1.

This alternative way of describing surfaces as linear complexes of line elements opens up a new way of studying them, as explained below.

5.2.3 Approximating the Complex

Suppose a scanning process results in a set of points X (a point cloud) which are the results of a scanning process. The aim is to determine the type of surface Φ on which these points lie. This knowledge would allow us to reconstruct the surface using its underlying geometrical properties. For instance, if we know our points result from the scan of a surface of revolution, we could determine the central axis etc. So, we are interested in the (linear) complex (of line elements) that best describes the given points. Its coordinates $(\mathbf{c}, \bar{\mathbf{c}}, \gamma)$ determine the type of surface [69].

First, we calculate the unit normal vectors from the point cloud at every point. This topic has been very well documented in the literature. We refer the reader to [103] for a more in-depth discussion. For each x_i in X with $i = 1, 2, ..., N$ we obtain a unit normal vector $\mathbf{n_i}$. From these normal vectors and corresponding points, we calculate their line elements $(\mathbf{n_i}, \mathbf{x_i} \times \mathbf{n_i}, \mathbf{x_i} \cdot \mathbf{n_i})$.

Second, according to Equation 5.3, a complex with coordinates $(\mathbf{c}, \bar{\mathbf{c}}, \gamma)$ that best fits these line elements minimises

$$F(\mathbf{c}, \bar{\mathbf{c}}, \gamma) = \sum_{i=1}^{N} (\bar{\mathbf{c}} \cdot \mathbf{l_i} + \mathbf{c} \cdot \bar{\mathbf{l}}_\mathbf{i} + \gamma \lambda_i)^2, \tag{5.4}$$

under the condition $\mathbf{c}^2 + \bar{\mathbf{c}}^2 + \gamma^2 = 1$. We follow the notation used in [121], in which $\mathbf{a}^2 = \mathbf{a} \cdot \mathbf{a}$. In order for this condition to make sense, we normalise our point cloud such that $\max \|x_i\| \approx 1$. We also centre it around the origin. We can rewrite this as

$$F(\mathbf{c}, \bar{\mathbf{c}}, \gamma) = (\mathbf{c}, \bar{\mathbf{c}}, \gamma) M (\mathbf{c}, \bar{\mathbf{c}}, \gamma)^T, \tag{5.5}$$

where $M = \sum_{i=1}^{N} (\bar{\mathbf{l}}_\mathbf{i}, \mathbf{l_i}, \lambda_i)^T (\bar{\mathbf{l}}_\mathbf{i}, \mathbf{l_i}, \lambda_i)$. This is an ordinary eigenvalue problem. The smallest eigenvalue of M corresponds to an eigenvector $(\hat{\mathbf{c}}, \hat{\bar{\mathbf{c}}}, \hat{\gamma})$ which best approximates Equation 5.3 for the given $(\mathbf{l_i}, \bar{\mathbf{l}}_\mathbf{i}, \lambda_i)$.

Some surfaces are invariant under more than one one-parameter transformation [123, 119, 89, 125, 121, 69, 109]. In that case, k small eigenvalues appear as solutions to Equation 5.4. The corresponding eigenvectors can be seen as a basis for a subspace in $\mathbb{R}^7$. We list the possibilities below:

- $k = 4$: Only a plane is invariant to four independent uniform motions.
- $k = 3$: A sphere is invariant to three independent uniform motions (all rotations).
- $k = 2$: The surface is either a cylinder of revolution, a cone of revolution or a spiral cylinder.
- $k = 1$: The surface is either a cylinder without revolution (pure translation), a cone without revolution (central similarity), a surface of revolution, a helical surface or a spiral surface.

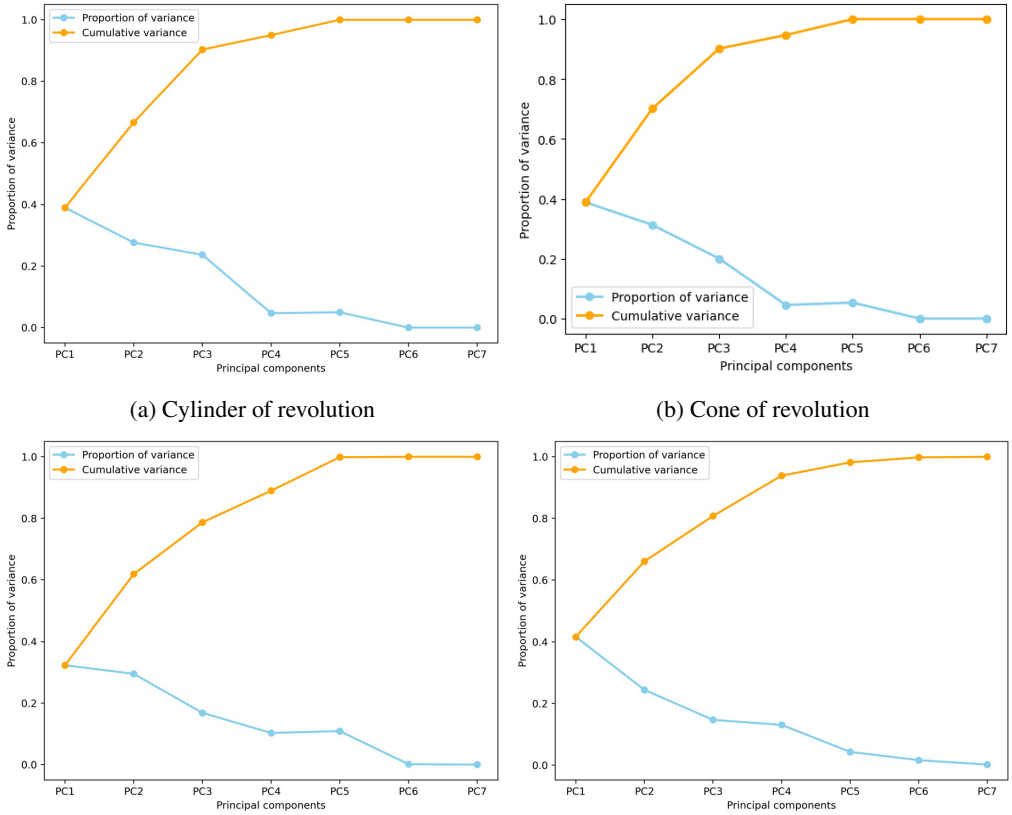
Further examination of the coordinate $(\mathbf{c}, \bar{\mathbf{c}}, \gamma)$ determines the exact type of surface, as described in Subsection 5.2.2. Multiple examples, applications and variations on this theme can be found in the above mentioned references. In Figure 5.2, we visualise the scree plots of the PCA method applied on the primitive surfaces shown in Figure 5.1.

Although this is a very elegant and powerful approach, some issues are discussed in the works listed above. First, this method is very sensitive to outliers. A solution proposed by the authors is

to apply a RANSAC variant or to iteratively downweigh the outliers by

$$M = \frac{1}{\sum \sigma_i} \sum_{i=1}^{N} \sigma_i (\bar{\mathbf{l}}_\mathbf{i}, \mathbf{l}_\mathbf{i}, \lambda_i)^T (\bar{\mathbf{l}}_\mathbf{i}, \mathbf{l}_\mathbf{i}, \lambda_i). \tag{5.6}$$

This obviously results in longer computation times. Second, numerical issues can arise calculating the eigenvalues (especially for planes and spheres). Third, some shapes do not fall into the classification of these simple geometric forms. This is certainly the case for organic surfaces that can be found in nature or when modelling the human body. Reconstructing a surface based on a simple geometric shape is obviously only valid if the surface resembles the shape well. Fourth, some shapes are either a combination or a composition of the elementary simple shapes (e.g. pipe constructions). Moreover, the shape of interest might show small but not zero invariances under a one-parameter transformation. For instance, a helical surface that only has one revolution instead of many more. In this case, the helical character is not very outspoken, which results in steadily increasing eigenvalues. this can be seen in the scree plots of Figures 5.2 and 5.3. The question arises of what constitutes as a small eigenvalue and where to draw the line. Even though some guidance is given in the literature, these thresholds are often application specific parameters.



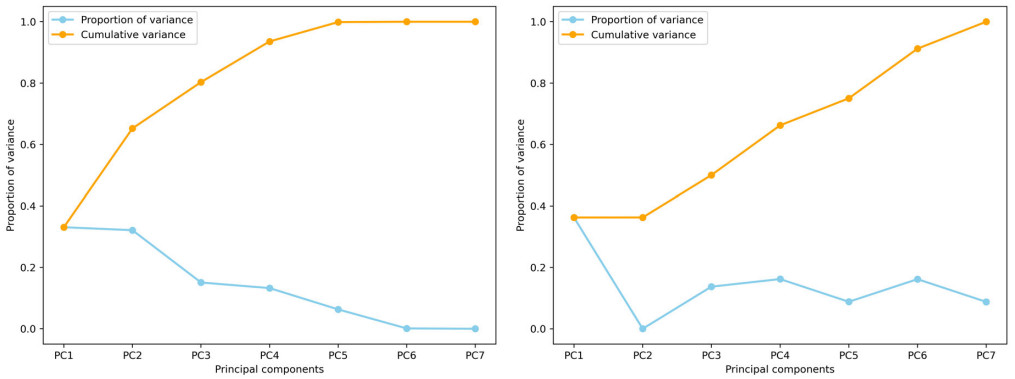
(a) Cylinder of revolution

(b) Cone of revolution

(c) Spiral cylinder

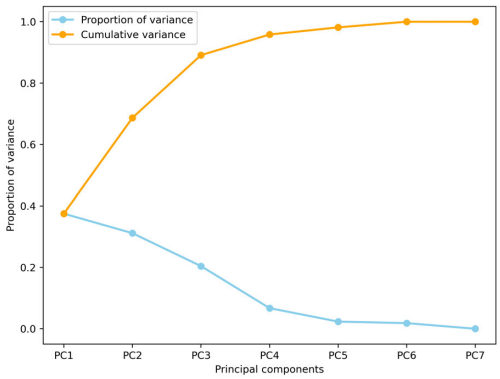
(d) Cylinder without revolution

Figure 5.2: PCA scree plots for the examples of equiform kinematic surfaces given in Figure 5.1(a)-(d). For (a)-(c), we have $k = 2$. For (d), $k = 1$.

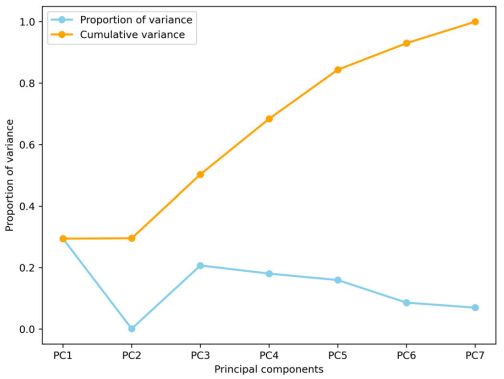


(a) Cone without revolution

(b) Surface of revolution



(c) Helical surface



(d) Spiral surface

Figure 5.3: PCA scree plots for the examples of equiform kinematic surfaces given in Figure 5.1(e)-(h), for which $k = 1$.

Our approach provides a solution for these issues by finding a representative lower dimensional latent space for the line elements in a more flexible non-linear way. This is no longer a linear subspace in $\mathbb{R}^7$. We address the issue of strict classification in shape primitives and the problem of the small eigenvalues cutoff in Section 5.3.1. We also demonstrate how this can be applied on surface segmentation in 5.3.2. Outliers and denoising is handled in Section 5.3.3.

5.2.4 Gaussian Process Latent Variable Models

Principal component analysis (PCA) transforms a set of data points to a new coordinate system, in which the greatest variance is explained by the first coordinate (called the first principal component), the second greatest variance by the second coordinate, etc. This reprojection of the data can be exploited in dimensionality-reduction by dropping the components with the smallest variance associated. The result will still contain most of the information of the original data. This method can also be explained as a statistical model known as probabilistic PCA (PPCA) [152], which implies that the principal components associated with the largest variance also maximise the likelihood of the data.

In dimensionality-reduction, the representation of the original data by its retained principal components, can be interpreted as a latent variable model, in which the n latent variables $\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\right]^T$ are of a dimension k that is lower than the dimension d of the original data. PPCA requires a marginalisation of those latent variables and an optimisation of the mapping from the latent space to the data (observation) space. For n d-dimensional observations $\mathbf{Y} = \left[\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n\right]^T$ we can write

$$\mathbf{y}_i = \mathbf{W}\mathbf{x}_i + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2), \tag{5.7}$$

in which $\mathbf{x}_i$ is a k-dimensional latent variable with $k < d$, $\mathbf{W}$ is a $d \times k$ matrix representing the mapping and ε is observation noise.

In [85, 86], a dual approach is proposed by marginalising the mapping $\mathbf{W}$ and optimising the latent variables $\mathbf{X}$. This approach is called the Gaussian process latent variable model (GPLVM) and is achieved by maximising the Gaussian process likelihood $\mathcal{L}$ with respect to the latent variables. We optimise

$$\mathcal{L}(\mathbf{X}) = -\frac{dn}{2}\log 2\pi - \frac{d}{2}\log|\mathbf{K}| - \frac{1}{2}\mathrm{tr}(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^\top), \tag{5.8}$$

with respect to $\mathbf{X}$. It is proved in [86] that this approach is equivalent to PCA when using a linear kernel to compose $\mathbf{K}$, which can be written as

$$k_{linear}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'. \tag{5.9}$$

However, by choosing a nonlinear kernel, we can establish a nonlinear relationship between the latent variables $\mathbf{X}$ and the observations $\mathbf{Y}$. This relationship can also be seen as placing a Gaussian process prior on each column of the $n \times d$ matrix $\mathbf{Y}$ and allows for a more flexible mapping between latent and data space.

In the original GPLVM, the unobserved inputs are treated as latent variables which are optimised. Another approach is to variationally integrate them out and compute a lower bound on the exact marginal likelihood of the non-linear variable model. This is known as the Bayesian Gaussian process latent variable model (BGPLVM) [153]. It is more robust to overfitting and can automatically detect the relevant dimensions in the latent space. These dimensions are characterised by a larger Automatic Relevance Determination (ARD, see also Section 2.2.1) contribution, which are the inverse of the length-scales l_j in Equation 2.9. Every component in data space is a vector whose components are formed by as many Gaussian processes as there are input dimensions. These ARD contributions determine the weight of the outcomes of each of the Gaussian processes and thus its corresponding input dimension. Less relevant dimensions result in longer length-scales and are pruned out. In this work, we exploit this by performing this Bayesian nonlinear dimensionality reduction on the seven-dimensional line elements. For most shapes in 3D a lower dimensional representation in a latent space can be found, as we show below.

The GPLVM is a map from latent to data space. As such, it preserves local distances between points in the latent space. However, this does not imply that points closeby in the data space are also closeby in the latent space. To incorporate this extra feature, one can equip the GPLVM with a so called back constraint [87]. This constraint is accomplished by introducing a second distance preserving map from the data to the latent space. We refer to this model as the back constrained Gaussian process latent variable model (BCGPLVM). A thorough review on GPLVM and its variants, including more than the ones mentioned here, can also be found in [28, 91] and more recently in [30, 84].

5.2.5 Our Approach

In this section, we explain how all of the above mentioned concepts come together in our approach. To recap, we can represent a straight line in 3D space by Plücker coordinates, which are six-tuples. By adding a seventh component, we can specify a point on that line. We can do this for every n points in a given point cloud. The line we choose through each point, is the normal line to the surface that is captured by that point cloud. We thus obtain a set of seven-dimensional line elements, that captures the information about the surface we want to examine.

The theory of kinematic surfaces links the line elements that are contained in a linear line element complex to an equiform kinematic surface. Finding this complex comes down to solving an ordinary eigenvalue problem. The dimensionality of the linear subspace, in which the line elements live, and the resulting eigenvalues determine the type of surface. In essence, this is dimensionality reduction of the seven-dimensional line elements via PCA.

However, PCA is a linear mapping. In contrast, our model is built on the Gaussian process latent variable model (GPLVM), which allows for a non-linear mapping. This results in a more nuanced way to represent the surface. Our model is only given the seven-dimensional line elements and finds the mapping from a latent (unobserved) space to these line elements. Each of the seven dimensions of the line elements is assigned to a Gaussian process (GP). The outputs (predictions) of those GPs are the components of the line elements. The inputs (training data) are the latent points, which are treated as variables of the overall model and are optimised (or integrated out in the Bayesian formulation).

In the next section, we will describe in more detail how we compose the datasets and elaborate on our experiments.

5.3 Results

All 3D models, datasets, plots and trained GP models described below can be found in our GitHub repository.

In order to assess the latent representation of various 3D shapes, we composed a collection of both synthetically generated point clouds and real world scanned objects. An overview can be found in Table 5.1. The synthetically generated point clouds are based on objects drawn in the free and open source Blender 3.3 LTS[2]. Point clouds resulting from real world scans were created on an Android mobile phone using the photogrammetry KIRI engine[3] and imported in Blender, where they are cleaned up by dissolving disconnected points, removing the background and subsampling using standard Blender tools. However, they still contain some overlapping triangles and other mesh irregularities. Synthetic models were made noisy by first subdividing the mesh several times and then applying the Blender Randomise tool on the vertices. This breaks the lattice structure of the vertices. Moreover, this makes them resemble a real world scan, where imperfections are inevitable. In this chapter, we restrict ourselves to one noise level and leave the effect of the amount of noise on our point clouds as future work. The noiseless and noisy bent torus are the same as their ordinary torus variant with a Simple Deform modifier of 45° applied to it. All

[2] `https://www.blender.org/`
[3] `https://www.kiriengine.com/`

models are exported in the Polygon File Format format (.ply), resulting in files consisting of points and unit normals for those points.

The line elements are calculated in Matlab R2020b and exported as comma-separated values (.csv). These serve as the data space for the GPLVM models, which are implemented using the python GPy library[4]. Some point clouds were subsampled uniformly for performance reasons. All GPLVM models were initialised with the results from a PCA. The BCGPLVM models were implemented with a MLP mapping with 5 hidden layers. The details are in Table 5.1. All code for training the models as well as the trained models themselves are available via notebooks in the GitHub repository.

5.3.1 Surface Approximation

We trained a Bayesian Gaussian process latent variable model for all examples of the equiform kinematic surfaces listed in Section 5.2.2. As can be seen by the ARD contributions in Figures 5.4 and 5.5, these surfaces can be represented by a lower dimensional representation. These dimensions are characterised by the largest ARD contribution and thus the smallest length-scales. In contrast, the PCA method does not show these clear latent dimensions in the scree plots in Figure 5.2 and 5.3. A 2D plot of the surface points in their GPLVM latent space is given by

[4] `http://sheffieldml.github.io/GPy/`

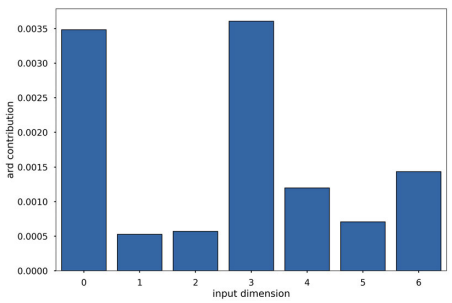
	Model	Vertices	Train	Noise	Inducing	Restarts
Cylinder of rev	BGPLVM	2176	2176	1.00E-04	15	10
Cone of rev	BGPLVM	2176	2176	free	50	10
Spiral cylinder	BGPLVM	2210	2210	free	25	10
Cylinder w/o rev	BGPLVM	1717	1717	free	50	10
Cone w/o rev	BGPLVM	2210	2210	free	50	10
Surface of rev	BGPLVM	2816	2500	free	50	10
Helical surface	BGPLVM	2582	2500	free	25	10
Spiral surface	BGPLVM	1842	1842	free	50	10
Torus	BGPLVM	2048	2048	free	50	10
Torus bent	BGPLVM	2048	2048	free	25	10
Pear	BGPLVM	6356	5000	free	50	5
Mixture 1	BCGPLVM	10653	1000	1.00E-06	NA	3
Mixture 2	BCGPLVM	6555	1000	1.00E-06	NA	3
Mixture 3	BCGPLVM	15681	1000	1.00E-04	NA	3
Hinge	BGPLVM	22282	5000	free	50	3
Torus bent noisy	BGPLVM	8192	8192	free	50	3

Table 5.1: An overview of the surfaces and their corresponding GPLVM and properties. The word revolution is abbreviated to rev. Larger point clouds are subsampled uniformly to a smaller set. The number of points retained for training are given in Train. Noise is the Gaussian noise variance hyperparameter for the GPLVM model, which is either a fixed value or a value that has to be learned along with the other hyperparameters. The number of inducing points for the Bayesian GPLVM are shown in Inducing. To make sure we do not end up in local minima, we restarted the training of the model a number of times given in Restarts.

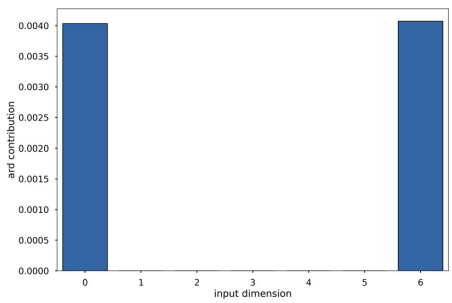
Figure 5.6.

All surfaces show a clear structure in their latent space. Note that the number of small eigenvalues does not necessarily correspond with the number of relevant dimensions in latent space. The latter are the result of an optimisation algorithm in which both the latent points and the kernel hyperparameters are found. This can be seen in the ARD contribution plot for the helical surface. The plot for the cylinder of revolution even has a significant value for all seven dimensions. This effect can be thought of as overfitting [84], as the model attributes importance to more latent dimensions than needed. In our experiments, we tried to lower this effect by making the model less flexible. We added a fixed noise term to the hyperparameters and lowered the number of inducing points. For details per surface we refer to Table 5.1. All trained models are available in the supplementary material on the GitHub repository.

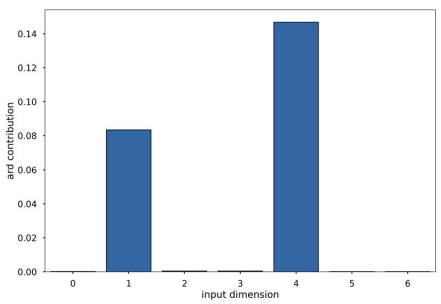
Another important remark is that the mapping from latent to data space is non-linear. Care must be taken when interpreting the 2D latent space plots. For instance, the spiral surface clearly has a one-dimensional subspace. However, the 2D plot shows a scattered cloud of points. This is an artefact of the visualisation. The ARD plot indicates that only dimension 1 has a significant contribution. Another example is given by the cylinder without revolution. Its subspace in $\mathbb{R}^7$ is one-dimensional, which manifests itself as a curve-like trail of points in the 2D latent space.



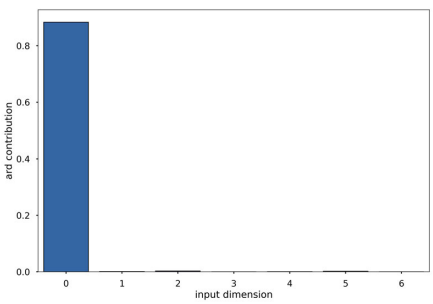
(a) Cylinder of revolution



(b) Cone of revolution

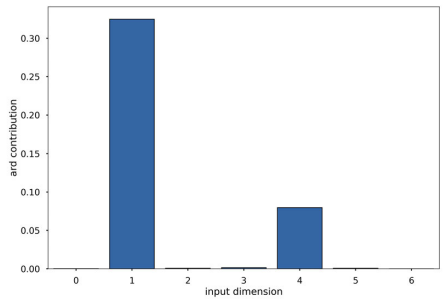


(c) Spiral cylinder

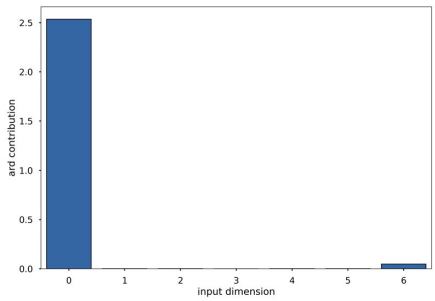


(d) Cylinder without revolution

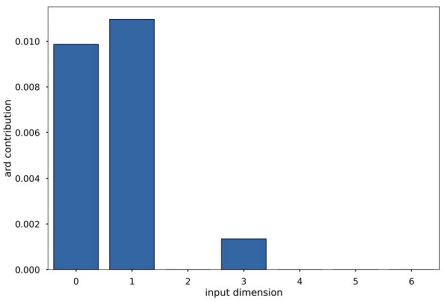
Figure 5.4: ARD contributions for the dimensions of the latent space for the examples of equiform kinematic surfaces given in Figure 5.1(a)-(d).



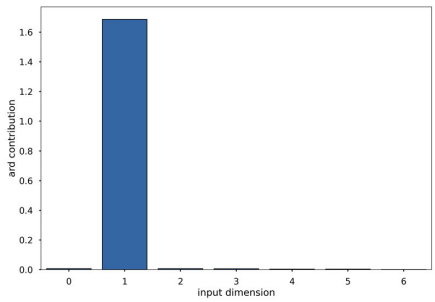
(a) Cone without revolution



(b) Surface of revolution



(c) Helical surface



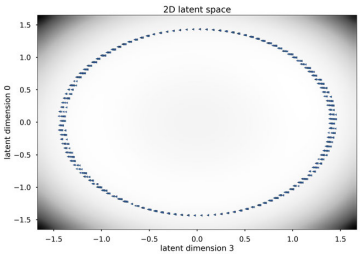
(d) Spiral surface

Figure 5.5: ARD contributions for the dimensions of the latent space for the examples of equiform kinematic surfaces given in Figure 5.1(e)-(h).

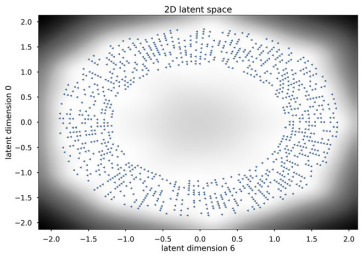
So far, nothing is gained by this new Bayesian GPLVM way of representing surfaces. The difference with the approach described in Section 5.2.2, is that we are no longer restricted to the simple geometric surfaces of Figure 5.1 and their linear subspaces of $\mathbb{R}^7$. We can now also describe surfaces that do not fall in the categories listed above. We investigate two cases.

First, we apply our method to a bent torus. This is a surface of revolution which we altered using the Simple Deform modifier in Blender to bend it 45° around an axis perpendicular to the axis of rotational symmetry of the original torus. This removes the rotational symmetry altogether. The results can be seen in Figure 5.7. We notice that the BGPLVM only deemed one dimension as significant. The 2D plot reveals the latent structure.

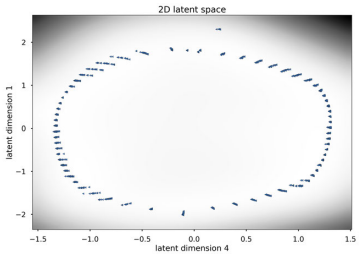
Second, we look at the surface of the point cloud obtained by scanning a pear as described above. This is an organic shape, so it possesses the same challenges as working with shapes that can be found in other items from nature or when modelling the anatomy of humans and animals. We are only interested in the shape of the body, so we removed the stalk and the bottom part when cleaning up the 3D model. In this case, the 3D shape resembles a surface of revolution, but the axis is bent irregularly and the rotational symmetry is broken (not all normals intersect the axis of rotation). The results can be seen in Figure 5.8. The darker region of the 2D latent plot indicates more posterior uncertainty. In the latent space we observe a set of points similar to what we saw for a cylinder of revolution with an additional distortion in a third latent dimension.



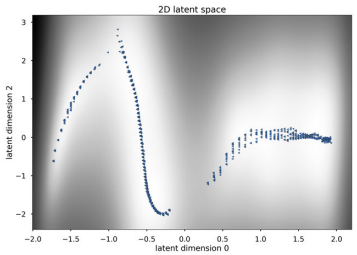
(a) Cylinder of revolution



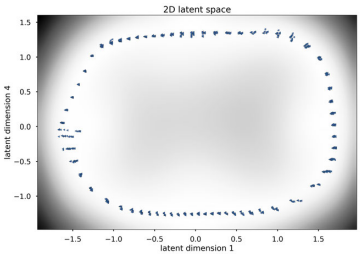
(b) Cone of revolution



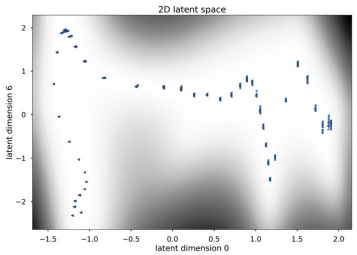
(c) Spiral cylinder



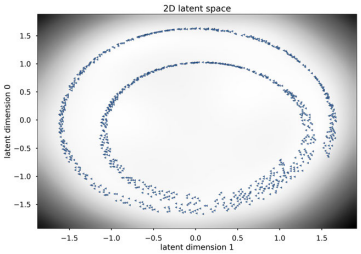
(d) Cylinder without revolution



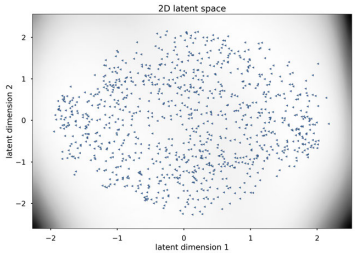
(e) Cone without revolution



(f) Surface of revolution

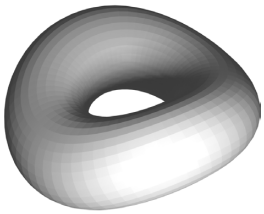


(g) Helical surface

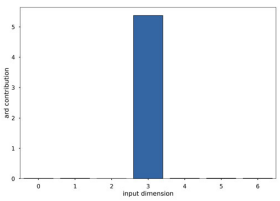


(h) Spiral surface

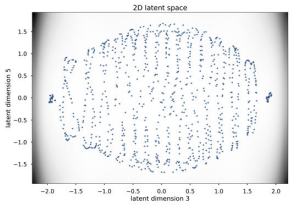
Figure 5.6: A 2D representation of the points of the kinematic surfaces in their latent space. The amount of black in the background indicates the posterior uncertainty of the BGPLVM.



(a) A bent torus



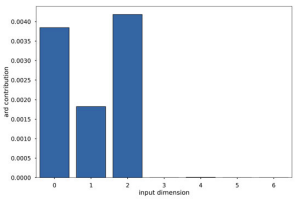
(b) ARD contributions



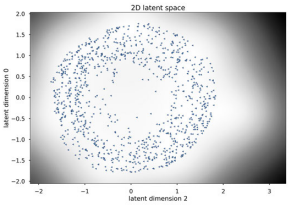
(c) 2D latent space

Figure 5.7: Results for a bent torus. One latent dimension is found to be dominant.

(a) A real world pear



(b) ARD contributions



(c) 2D latent space

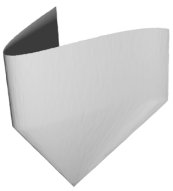
Figure 5.8: Results for a real world pear scanned with a mobile phone app.

Both the bent torus and the pear can not be described by a equiform kinematic surface. Applying the methods of Section 5.2.2 (i.e. approximating by a linear complex of line elements) for classification is numerically still possible. A small set of eigenvalues can be found. However, their interpretation would be faulty. The bent torus shows one small eigenvalue, $\mathbf{c} \neq (0,0,0)$, $\gamma = 0$ and $\mathbf{c} \cdot \bar{\mathbf{c}} \approx 0$. Unsurprisingly, these values fit a surface of revolution. They resemble the values for the torus or the torus with noise. However, blindly using the methods from [89] would result in a perfect surface of revolution. The same reasoning can be applied to the scanned pear's point cloud. Below, we show how to exploit our newfound GPLVM representation in surface approximation, surface segmentation and surface denoising.

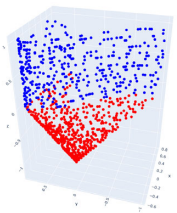
5.3.2 Surface Segmentation

A major challenge in point cloud classification is the segmentation of sub regions within that cloud. Once points are grouped together in simpler shapes, the underlying structure can be found via either our method or the methods described in [123, 119, 89, 125, 121, 69, 109]. In these works, several approaches are described for discovering the sub regions. Mostly, they are based on time consuming trial and error RANSAC. Here, we show that working in a latent space can be beneficial. The challenge is to group points together, whose line elements show similar behaviour.

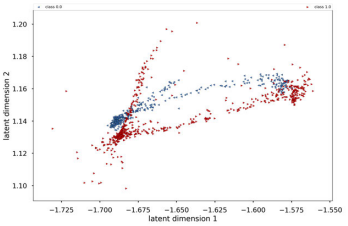
As we want to separate coherent groups of points in latent space, we care about their local distances. Points closeby in the latent space should be closeby in the data space as well. Therefore,



(a) Mixture 1

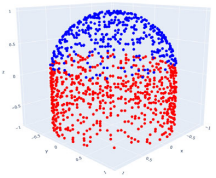


(b) Labelled points

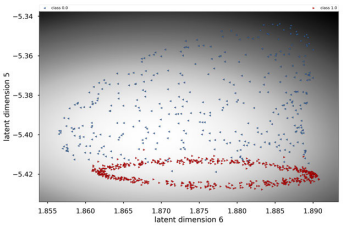


(c) Latent space

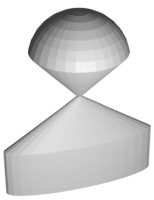
(d) Mixture 2



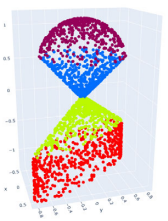
(e) Labelled points



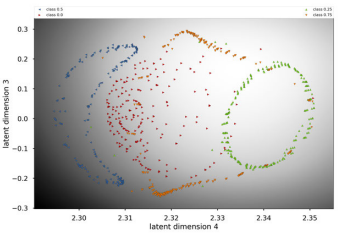
(f) Latent space



(g) Mixture 3



(h) Labelled points



(i) Latent space

Figure 5.9: Three synthetically generated 3D models by combining primitive surfaces. The BCG-PLVM is able to show distinguishable structures for points in latent space.

we expand our GPLVM with a back constraint as described in Section 5.2.4. We implement a multi-layer perceptron (MLP) mapping to capture the back constraint [87, 84]. The details for the different 3D models can be found in Table 5.1. As before, all code is available in the GitHub repository. The notebooks also include 3D plots made with the python open source graphing library Plotly[5], that allow user interaction such as 3D rotations. By rotating the viewpoint, we can clearly see how separable the latent points are.

To demonstrate our approach, we first designed three objects composed of different simpler geometric shapes. They can be found in Figure 5.9. The parts of these three models fall under the different categories described in Section 5.2.2. The aim of surface segmentation is to find those parts in an unsupervised manner.

[5] `https://plotly.com/python/`

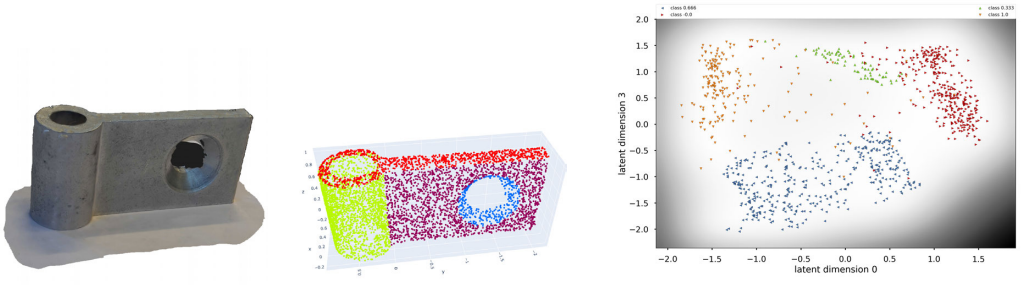
First, we created a 3D model called Mixture 1, which consists of a cylinder and cone, neither without rotational symmetry. Both of those shapes individually show one small eigenvalue and a clearly distinguishable curve in their 2D latent space, as can be seen in Figure 5.6. Combined, their latent space looks like two curves, shown in Figure 5.9. Notice that the 3D points that lie both on the cylinder and the cone, fall into both categories. Moreover, their normal is inconsistent with either of the the two shapes. For this cylinder, all normals are horizontal. For the cone, normals for points on a line connecting the cone's apex and its generating curve, are parallel. For points on the intersection of the cylinder and the cone, the normals are weighted with their neighbouring points. This results in a latent space that is not easily separable by clustering.

Second, the 3D model named Mixture 2 consists of a noisy cylinder of revolution where one end is closed by a demi-sphere. The former is characterised by two small eigenvalues and the latter by three. Again, this behaviour can be clearly observed in the latent space. Notice how the BCGPLVM formulates latent shapes for each part that are consistent with the kinematic surface described in Section 5.2.2. For the sphere, we observe a 2D shape. For the cylinder of revolution, an annulus can be seen. The supplementary material includes a interactively rotatable 3D plot where this cloud of latent point can be observed in more detail. We also see that the region for the tip of the demi-sphere has a darker background in the 2D latent plot, indicating more uncertainty in this region of the posterior. This can be explained by the fact that the normals of a sphere all intersect the centre of that sphere. As such, no normals are parallel. This results in line elements whose vector components vary more. Points with normals that lie in parallel planes, as is the case for a cylinder, have more similarity in their direction components of their line elements. Moreover, the hyperparameters in the mapping from latent to data space are optimised globally. This means for all latent points simultaneously. The strong structure in the cylinder part renders the large variations in the tip of the demi-sphere part as less likely. Hence the larger posterior variance.

Finally, in the 3D model Mixture 3 we grouped together the upper half of a sphere, a cone of revolution, a cone without revolution and a cylinder without revolution. These parts have three, two, one and one small eigenvalues respectively. As this model consists of four different parts, the segmentation is more complex. Nonetheless, the BCGLVM is able to find distinct substructures in the latent space, even in just two dimensions.

For a real world and more challenging example, we scanned a metal hinge, as described above. It can be found in Figure 5.10. The original 3D model and the cleaned up one can be found in the supplementary material. The 3D model is a collection of a cylinder of revolution, two planes and a cone-like aperture. It is important to notice that the scan itself is of poor quality, mainly due to the shininess of the metal and the lack of distinct features. There are holes and bumps in the surface, even after cleaning up the model in Blender. Moreover, the cone-like aperture does not have a lot of vertices (the region around the apex is completely missing). The latent space still shows the formation of clusters, especially when three dimensions are taken into account.

Once a latent space is found, the segmentation can be done via either manual selection of the latent points or a form of unsupervised learning. In the case of separable clusters, we can perform the well studied k-means clustering algorithm or drawing (hyper)planes determined by support vector machines (SVM). The details of these are outside the scope of this work. The reader is referred to [1] and [20] respectively. This segmentation can then be the basis to fit simple geometric surfaces to each cluster of points. As we can observe from the plots, some of the latent points do not belong to any of the found substructures. In practice, these can be ignored or filtered out. We are left with enough points to perform a best fit. Afterwards, we can determine whether or not such a



(a) Render of hinge (b) Labelled points (c) Latent space

Figure 5.10: The results for a real world scanned metal hinge. Again, the BGPLVM is able to separate the points in latent space.

rogue point belongs to the best fit subsurface or not.

5.3.3 Surface Denoising

In general, a Gaussian process can handle noise very well, even in low data regimes [130]. This means our technique is beneficial to denoise point clouds. Once a mapping is found from a latent space to a data space, it can be queried to predict new points in the data space. This can be used to handle missing data [30, 84]. Here, we take advantage of this feature by correcting noisy points in the lower dimensional latent space and predicting their counterparts in the data space. The smooth mapping allows re-predicting the line elements for every latent point.

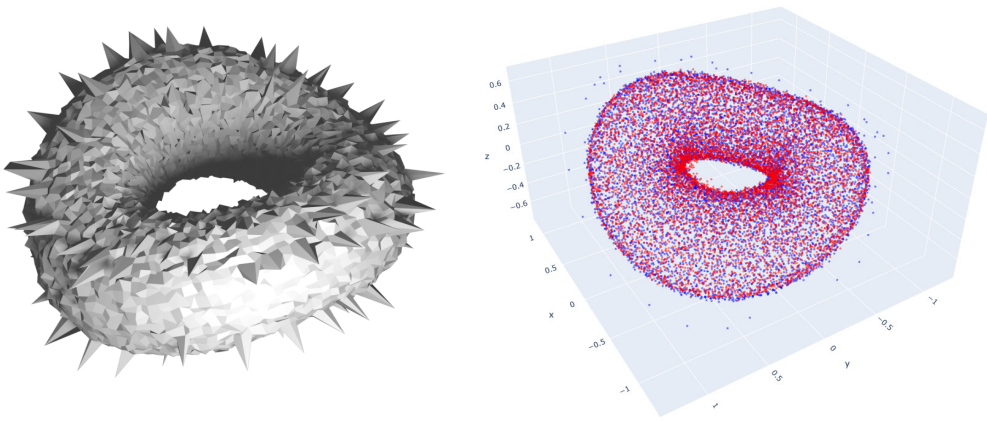
From a predicted line element $(\mathbf{l}, \bar{\mathbf{l}}, \lambda)$, with $\|\mathbf{l}\| = 1$, we can calculate a corresponding 3D coordinate $\mathbf{x}$ for a point x using

$$\mathbf{x} = \mathbf{l} \times \bar{\mathbf{l}} + \lambda \mathbf{l}. \tag{5.10}$$

To demonstrate this, we again work on the bent torus model. We introduce random noise with the Blender Randomise tool and select a hundred vertices at random which we translate to simulate shot noise. The results can be seen in Figure 5.11. The 3D model, the .ply file with the point coordinates and unit normal vectors, the .csv file with the line elements and notebook with the executed code for the BGPLVM can be found in the supplementary material. Once the BGPLVM is trained on the noisy point cloud, we use it to predict line elements for the latent point. From these line elements, we extract 3D coordinates for points via Equation 5.10. We observe that the BGPLVM is able to smooth out the translated vertices. This approach can also be used to detect and remove outliers.

5.4 Discussion

This chapter presented the first findings for this new GPLVM approach to describe 3D surfaces. In this chapter, we wanted to focus on the theoretical principles themselves and not overload this



(a) Deformed bent torus

(b) Denoised bent torus

Figure 5.11: A bent torus. Noise is added to the entire surface. Moreover, a hundred vertices were translated so simulate outiers. (b) The BGPLVM is able to smooth out the surface. Blue are the noisy and outlier points. Red are the denoised points.

chapter with additional research questions that determine the limits of this idea. Even though these are both interesting and important in real world applications, we leave them for future work.

We have shown surface segmentation for surfaces that are a combination of a few different simpler geometrical shapes. The question remains how many sub regions can be detected and what the complexity of those regions can be.

We presented the Bayesian GPLVM and the GPLVM with back constraints. There are more variations on this topic investigated in the literature. A recent paper describes a generalised GPLVM with stochastic variational inference [84]. They also present models for applying these techniques on larger dataset. This would be most applicable on larger point clouds that are often obtained in real world applications.

A line element is formed by a line and a point on that line. By working with normal lines for points on a surface, we effectively introduced a second so called *view* for those points, where we follow the terminology used in [28, 30]. These works present a multi-view unsupervised learning technique called manifold relevance determination (MRD), which offers another worthwhile approach.

The prediction as seven-tuples made by the model do not automatically follow the Grassmann-Plücker relationship in Equation 2.13 for their direction and moment vector. This leads to faulty line elements. In other words, the first six components of a line element vector are the Plücker coordinates of the line where the point of the line element lies on. In chapter 3 and 4 we described several ways of dealing with this.

5.5 Conclusions

We provided a theoretical introduction to kinematic surfaces and showed how they can be used to perform surface detection. Many simple geometric shapes manifest themselves as linear subspaces of line or line element space. This approach is limited by the linearity of the underlying eigenvalue problem. We expanded on this by reformulating this as a probabilistic non-linear non-parametric dimensionality reduction technique known as the Gaussian process latent variable model. We showed how this can be applied to many simple geometric surfaces, as well as surfaces that do not fall in any of these categories. Moreover, we showed the benefits of unsupervised surface segmentation and surface denoising. We presented findings on synthetically generated surfaces and scanned real-world objects.

The main goal of the current study was to determine the feasibility of applying the Gaussian process latent variable model to line element geometry. Even though several experiments are explained, and several more are included in the supplementary material, considerably more work will need to be done to determine the limits of this method. For instance, it remains an open question how noise affects the overall representation in the latent space. Moreover, we did not implement any optimisations on the training part of the underlying models, which is paramount for real world settings. We leave this as future work.

Another natural progression of this work is to further exploit the found latent space in the case of missing data. Point clouds sometimes have missing regions, caused by bad lighting conditions, occluded areas or areas that simply cannot be reached by the scanning device. Finding the 3D coordinates for the missing points is a classic example of the missing data problem. In our case, it manifests itself as a region in the latent space that is missing values. If the found structure in the latent space is enough to reconstruct those missing latent points, then the according data space points can also be inferred by the Gaussian process latent variable model.

Part III

Camera Calibration

Chapter 6

Checkerboard Detection

In an image taken by a camera, every pixel is the result of an incoming ray of light. Calibrating a camera comes down to determining which pixel corresponds to which ray. In other words, finding the values for the parameters in the camera model that produces the lines. The most simple camera model is a so called *pinhole camera*, where all rays go through a central point. However, many more complex cameras and camera systems have been developed over the last few decades. We elaborate more on camera types and their calibration in the next chapter. In this chapter, we focus on one particular step in the most common camera calibration method, namely the detection of the corners of a checkerboard. These corners have a known structure in the real world, which can be exploited to determine the camera parameters.

A variety of checkerboard detectors have been developed in the past decades. While some detectors are able to handle partially occluded checkerboards, they fail when a large occlusion completely divides the checkerboard. We propose a new checkerboard detection pipeline for occluded checkerboards that has a robust performance under varying levels of noise, blurring, and distortion and for a variety of imaging modalities. This pipeline consists of a checkerboard detector and checkerboard enhancement with Gaussian processes. By having a Gaussian process learn a mapping from the coordinates of the corners of a perfect virtual checkerboard to image pixel coordinates of corners detected in a real image, we can fill in occluded corners, expand the board beyond the image borders, allocate detected corners that do not fit an initial grid and remove noise on the detected corner locations. We show that our method outperforms other publicly available state-of-the-art checkerboard detectors, both in accuracy and in the number of corners detected. Our code and datasets are made publicly available on GitHub[1]. The checkerboard detector pipeline is contained within our Python checkerboard detection library called PyCBD. The pipeline itself is modular and easy to adapt to different use cases.

This chapter is also published in [68].

6.1 Introduction

Checkerboard detection is a fundamental tool in computer vision applications such as camera calibration [96, 22, 166, 79, 141, 2], projector-camera systems [149, 77], simultaneous localisation

[1] `https://github.com/InViLabUAntwerp/PyCBD`

and mapping (SLAM) [73], and robotics in general [47, 63, 155]. This topic is of such high importance that it has received a large amount of attention from the community over the past decades and a large variety of detection methods have been developed.

Generally, checkerboard detection involves corner detection, corner refinement, and checkerboard structure recovery [57, 22, 166]. The locations of the inner corners of a checkerboard pattern in an image are found with a corner detector. In many cases, additional refinement of the found corner locations is necessary to achieve sub-pixel accurate locations. The structure of the checkerboard is recovered by utilising the locations and structure of the detected corners, and the detected corners are mapped to local checkerboard coordinates.

A wide variety of automatic checkerboard detectors already exist. OpenCV's [78] standard checkerboard detector, *findChessboardCorners*, is based on the work of [158]. An improved version of this detector is proposed in [133], which is better suited for blurred and distorted images. More recently, a new detector called *findChessboardCornersSB* was added to OpenCV [41], which is more robust to all sorts of noise and runs faster on large images compared to *findChessboardCorners*. The major drawbacks of the previously mentioned detectors are that they require the size of the checkerboard and a thick white edge around the checkerboard is necessary for optimal performance. A checkerboard detector that does not require the size of the checkerboard was introduced in [57]. Their structure recovery algorithm can't cope with occlusions, which is particularly problematic when an occlusion occurs near the centre of the checkerboard. OCPAD [54], which is a successor to ROCHADE [118], and the newer version of OCamCalib [139] are both able to detect occluded checkerboards. Convolutional neural networks (CNNs) are becoming more ubiquitous for a large variety of detection tasks, including checkerboard corner detection [23, 22, 159, 163, 166, 79]. Both [23] and [166] have developed CNN-based checkerboard detectors. Chen et al. improved upon the structure recovery algorithm by Geiger et al., so it can be used to detect occluded checkerboards. At the time of writing, both OpenCV implementations, OCamCalib and the detector by Geiger et al. are publicly available. Zhang et al. share their code and data, but the trained weights for the CNN are omitted.

Many real world applications that rely on checkerboard detection suffer from low quality images, drastically impacting the corner detection performance. This could be due to low resolution, artefacts in the lenses of the camera used such as scratches or faults in the glass, contamination on the lens itself, heavily warped images from fisheye lenses, etc. An example of where these factors accumulate can be seen in Figure 6.7. The image was taken by a camera used in endoscopy. Besides the fisheye warping and the contamination in the lens, we can also see some corners being missed due to the specular reflection. The result is a checkerboard that is only partly detected.

In this chapter, we address the above-mentioned problems by proposing a new checkerboard detection pipeline with additional checkerboard enhancement using Gaussian processes (GP) [130], which can be performed after standard checkerboard detection. This machine learning technique is used to learn a mapping from local board coordinates to pixel coordinates, which can be used to predict the image coordinates for undetected or occluded corners and to smooth out the checkerboard corner image locations.

All of our code and the used datasets are publicly available on the above mentioned GitHub. The checkerboard detector and enhancement algorithms are contained within our Python checkerboard detection library called **PyCBD**. This library is modular and easy to adapt to different use cases.

The rest of this chapter is structured as follows: The next section describes the methodology im-

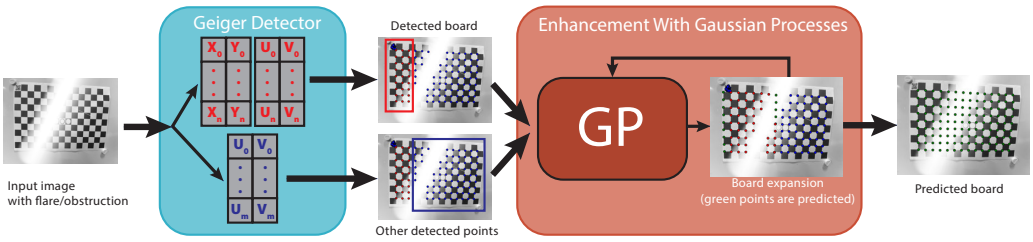


Figure 6.1: The proposed processing pipeline for enhancing a detected checkerboard using GP.

plemented in the enhancement using Gaussian processes. Section 6.3 explains our experimental setup. We discuss them in Section 6.4. Finally, we formulate our conclusions in Section 6.5.

6.2 Methodology

The complete pipeline is summarised in figure 6.1. In this section, we will explain the Gaussian process enhancement.

6.2.1 Allocation of Detected Corners

Many checkerboard detectors perform a checkerboard corner detection followed by a structure recovery algorithm that allocates these detected corners to positions in a grid, based on their location and topology. It is possible that not all detected corners are allocated to this grid because they do not fit according to the structure recovery algorithm. These detected corners could either be false positives, i.e., detected corners that are not checkerboard corners, or checkerboard corners that are somehow missed by the structure recovery step. The latter, for instance, could be the result of a large flare across the image, splitting the checkerboard into two distinct islands of ordered corners. An example of this is given in Figure 6.2.

In this section, we describe an algorithm to expand a partially detected checkerboard with detected corners that are part of the checkerboard pattern, but were not assigned to the grid by the structure recovery algorithm. By learning the map from local board coordinates to pixel coordinates, we can make predictions for missing grid points, which in turn can be used to add unassigned corners to the grid. This section relates to the last (red) box of our method in Figure 6.1. Our Algorithm is given in pseudo-code in Algorithm 1.

Our algorithm takes as input a partially detected checkerboard, which has two sets of coordinates for each corner: a set of image coordinates (boardUV) and a set of local grid coordinates (boardXY), and the image coordinates of the unassigned detected corners (cornersUV), which don't have local grid coordinates. The aim of the algorithm is to find the corresponding local xy-coordinates for any remaining corners, filter out corners that are not part of the checkerboard and find occluded corners. Each of these three sets could be empty. This is not known at the start.

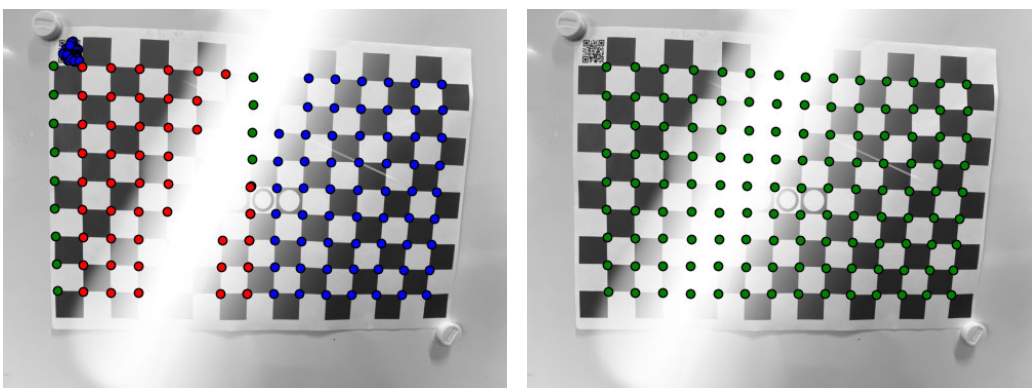
The central idea behind our algorithm is the mapping from local xy-coordinates of points on a perfect virtual checkerboard to uv-coordinates of pixels of points in a given image. This mapping is learned via Gaussian processes. We implemented two distinct GPs in parallel to perform

Algorithm 1 Algorithm to allocate all detected corners

Require: *boardXY*, *boardUV*, *cornerUV*
  maxNrOfIterations := 10
  currIteration := 1
  repeat
    GPs ← *boardXY*, *boardUV*
    Expand local grid of corners with *newXY*
    Predict *newUV* for given *newXY*
    Match *newUV* with existing *cornerUV*
    Augment *boardXY*, *boardUV*
    currIteration++
  until *cornerUV* = *boardUV* **or**
    currIteration > maxNrOfIterations **or**
    no *newUV* in image ▷ leave loop
  No more corners to be allocated
  return *boardUV* ▷ possibly augmented

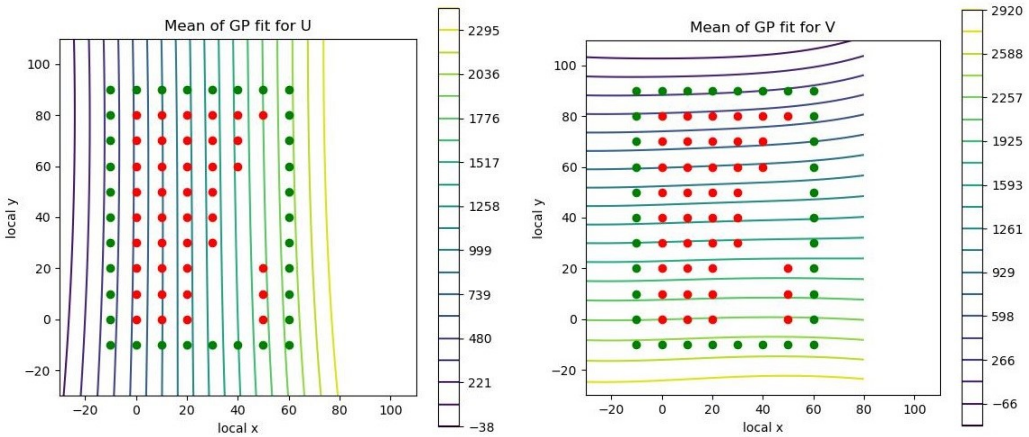
regression from xy-coordinates to both the u-coordinates and the v-coordinates of detected corners. They both take a 2D point as input and produce a scalar value each. In Algorithm 1, the image coordinates of the corners are called *cornerUV*. The image and local coordinates of the corners that are already on a (possibly impartial) checkerboard are called *boardUV* and, *boardXY* respectively.

In the second step, for each of the predicted corners, we search for a viable detected corner. If a corner is found, then it is added to the list of checkerboard corners. Its xy-coordinates are the input of the GP and its uv-coordinates are the detected values (not the predicted ones). If no corner can be found within a reasonable distance, which is a fraction of the distance between



(a) (b)

Figure 6.2: (a) Visualisation of iteration 4: red are corners for which a local coordinate is found, blue are the remaining corners without a local coordinate and green are the predicted corner locations. Notice how the original corner detection also detected corners in the QR code in the upper left corner of the image. As these corner coordinates do not correspond to any predicted values of the GP, they are deemed as false positive and filtered out. (b) The entire re-predicted checkerboard in green.



(a) (b)

Figure 6.3: An example of a situation given by the algorithm after four iterations. The red dots are the locations of detected checkerboard corners on a perfect checkerboard. The green dots are locations where we predict a value for the u and v coordinate of those corners, in (a) and (b) respectively. The values for the coordinates are given by the contour lines. Notice how these contour lines capture the curvature of the checkerboard in the real world. A flat checkerboard would result in straight lines for the contour lines. The deviation from this is the result of the bending of the paper checkerboard as seen in Figure 6.2.

the checkerboard corners, then this prediction is dropped. This fraction is a hyperparameter that can be tuned for the specific application. If the value is too large this might result in more false positives being accepted or corners being attributed to the wrong position in the grid. Naturally, this is less of a problem when there are few or no false corner detections. If the value is too small it might result in too few corners being added to the set of accepted corners. The latter manifests itself more when we are dealing with heavily warped images.

We repeat these steps, each time with a list of corners that is extended with the newly found ones. If no new corners can be found, then we expand the local xy-coordinates with two rows or columns instead of one. This allows us to bridge large gaps.

We keep iterating these steps until a maximum of steps have been reached, no new points are within the image or no more corners are without corresponding xy-coordinates (all corners are accounted for). Corners not close to a predicted corner are deemed false positives or outliers. These are falsely detected corners and are ignored on the checkerboard.

At the end of the algorithm, all detected corners are either part of the checkerboard or considered a false positive. A visualisation of a situation after a few iterations is given in Figure 6.3.

6.2.2 Gaussian Process Refinement

After running the algorithm described in Section 6.2.1, we retrain the GPs one more time on all found corners. This allows us to exploit the GP in two ways.

First, we make predictions for local xy-coordinates of corners that are without corresponding uv-coordinates. These are locations in the image where no corner is detected, even though the checkerboard tells us there should be one. This enables us to fill in occluded corners and even corners that are outside the borders of the image. An example can be seen in Figure 6.2, where a large flare hides some corners.

Second, we re-predict the uv-coordinates for all detected corners. In this work, we implemented a squared exponential kernel, which yields smooth functions that are infinitely many times differentiable [130]. This results in an extra refinement step which removes noise from those predicted coordinates. This is due to the fact that the GP utilises the uv-coordinates of all corners to predict a single one and does not base its prediction only on the surrounding pixels.

6.3 Experimental Setup

6.3.1 Dataset Generation

In this work, two datasets are generated. A synthetic dataset where checkerboards are generated artificially and drawn on random background images. The background images are generated through DALL-E 2 and made to resemble a typical laboratory environment. When creating the checkerboards onto the background images, the ground truth points are known. When augmenting the images, these reference points are transformed in the same way, thus obtaining the actual ground truth for the checkerboard locations. The second dataset, referred to as the real dataset, is generated using two types of cameras: a thermal infrared (IR) camera (Xenics Ceres) and a snapshot multispectral imaging (MSI) camera (Photonfocus MV0-D2048x1088-C01-HS02-G2). The multispectral and thermal infrared cameras are specifically chosen because they pose a challenge to current checkerboard detection methods due to heavy blurring or limited contrast between checkers.

When using a synthetic dataset, the exact sub-pixel location of the corners can be easily determined with high accuracy. This level of precision is difficult to obtain when working with real datasets. Several different types of distortions are used to create realistic environmental conditions [17]. These techniques include Gaussian blur, shot noise, optical distortions such as pincushion, moustache and barrel distortion, perspective transformations, and rotational transformations, as is common in the literature [23]. For the Gaussian blur, multiplicative noise and projective distortions, different levels of augmentations are used, from slightly distorted to heavily distorted. For each category and level of augmentation, 100 images are generated and evaluated.

6.3.2 Evaluation Methods

For the synthetic dataset, two metrics are used to evaluate the different methods: the amount of fully detected checkerboards and the pixel error of the corner locations. A fully detected checkerboard refers to cases where all the checkerboard corners are correctly detected by the algorithm. A checkerboard corner is considered to be correctly detected if its predicted location is within a Euclidean distance of 2 pixels from the true location. The pixel error is computed as the average Euclidean distance between the predicted and true locations of all the checkerboard corners in the image. The lower the value of this metric, the better the accuracy of the algorithm

in predicting the location of the checkerboard corners. The subsequent processing of the image will yield better results.

The results of both datasets, simulated and real, are discussed separately below. Four different methods are compared to each other: The method as proposed by Geiger with and without Gaussian enhancement (Geiger, Geiger + GP) and the industry standards OpenCV (OpenCV) and OpenCV Sector Based approach (OpenCV SB) [41]. We opted for the Geiger detector for two main reasons. First, it is widely used in the field. Second, the source code is publicly available. The Geiger detector we used is a lightly modified version of the implementation in the *libcbdetect* library[2]. It should be noted that the simulated images are tailored for use with OpenCV: there are clear white and black edges and a thick white border around the checkerboard and the checkerboard are always completely in the image. Not doing so would result in a lower detection rate.

6.3.3 Synthetic Data Results

The results of the synthetic images, with varying degrees of blur, shot noise and perspective transformations are displayed in Figure 6.4, 6.5 and 6.6. Each of these figures is composed of two graphs. The left graph displays the amount of fully detected checkerboards, as a percentage of

[2] `https://github.com/ftdlyc/libcbdetect`

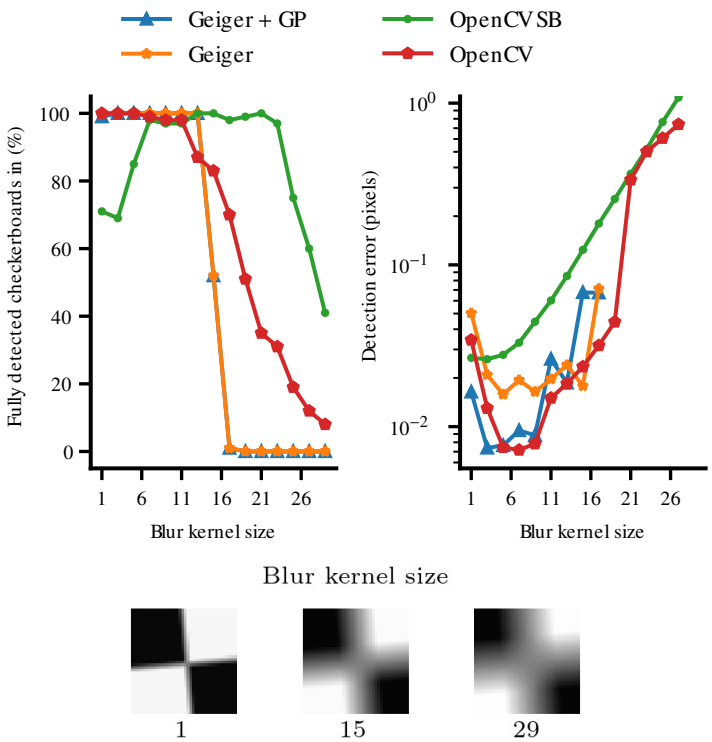


Figure 6.4: Testing different checkerboard detection methods on checkerboard blur.

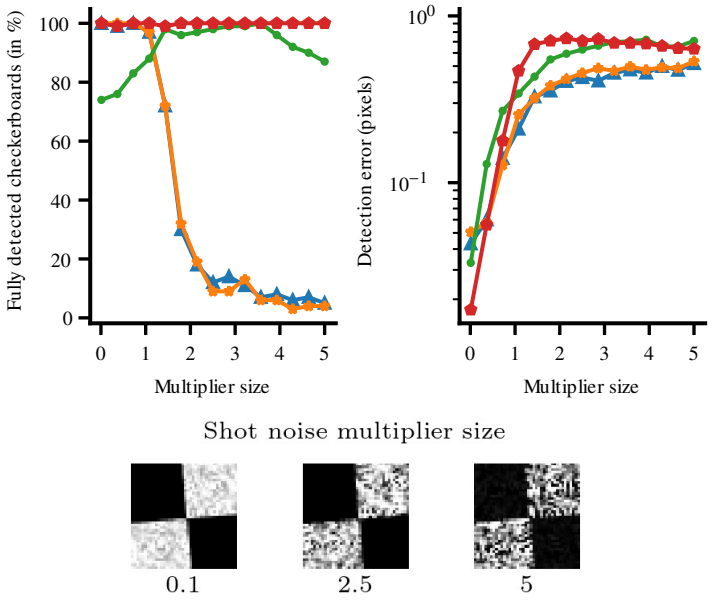


Figure 6.5: Testing different checkerboard detection methods on shot noise.

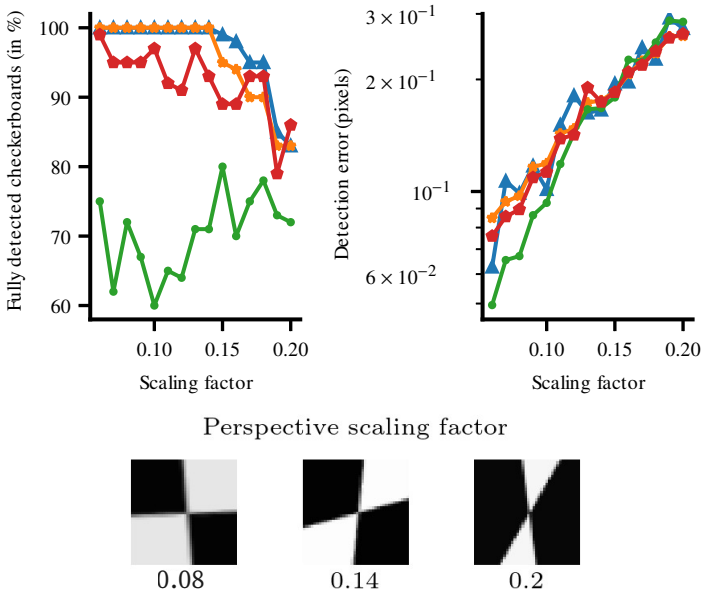


Figure 6.6: Testing different checkerboard detection methods on perspective distortion.

the total. The right graph shows the average error for each method. To obtain a fair comparison between the methods, the average of a single checkerboard is only taken into account when four or more methods find all the points of the respective checkerboard.

Fully detected checkerboards [%]				
Type of distortion	**Geiger**	**Geiger + GP**	**OpenCV**	**OpenCV SB**
Barrel	100	100	100	65
Mustache	100	100	100	49
Pincushion	85	100	100	85
Average corner error [pixels]				
Type of distortion	**Geiger**	**Geiger + GP**	**OpenCV**	**OpenCV SB**
Barrel	0.078	0.069	0.051	0.032
Mustache	0.085	0.066	0.063	0.048
Pincushion	0.179	0.178	0.170	0.140

Table 6.1: Results for different types of distortion used in the synthetic dataset

The Geiger detector, and by extension the proposed Geiger + GP method, fails to detect corners under a lower degree of blurring and shot noise compared to the OpenCV detectors, as can be seen in the left graphs in Figure 6.4 and 6.5. The right graphs in Figure 6.4 and 6.5 show that the GP corner refinement is able to reduce the corner position error when blur and shot noise is introduced into the images. The Geiger method is able to detect more corners under higher degrees of perspective transformation compared to the OpenCV methods, and the GP enhancement is able to improve this further for higher degrees of perspective scaling. The OpenCV SB method performs notably worse for all degrees of perspective transformation. This is shown in the left graph of Figure 6.6.

When different types of optical distortion are applied all checkerboard detection methods are able to detect most if not all checkerboards, except for OpenCV SB, as shown in Table 6.1. The mean pixel error is similar for all methods but worst for the Geiger detector. The GP corner refinement is able to slightly improve this result but not enough for it to be better than the OpenCV methods, as shown in Table 6.1.

When occlusions are added to the images, we observe that the GP enhancement step allows us to obtain fully detected checkerboards. The average pixel error for the occluded pixels is still sub-pixel range, as can be seen in Table 6.2.

Number of missing corners	**Average corner error [pixels]**
1	0.0226
3	0.0589
5	0.1023

Table 6.2: Results for increasing amounts of occluded corners introduced into the synthetic dataset. Only the results for the Geiger + GP method is shown here. The OpenCV methods are not included as they are unable to cope with occlusions. The reported pixel error is only for specific occluded pixels.

Detected corners [%] Real World Dataset				
Detection Method	**IR**	**IR Inverted**	**MSI Large CB**	**MSI Small CB**
OpenCV	12.65	96.15	14.23	58.40
OpenCV SB	6.44	46.15	24.21	25.60
Geiger	67.25	84.61	22.54	13.43
Geiger + GP	95.77	96.15	77.64	88.80

Table 6.3: Results for the different checkerboard detection methods for both the infrared (IR) images and the multispectral images (MSI). For the IR images, one set of results is without inverting the images and one with inverting the images. For the MSI images, a large and small checkerboard is evaluated.

6.3.4 Real Data Results

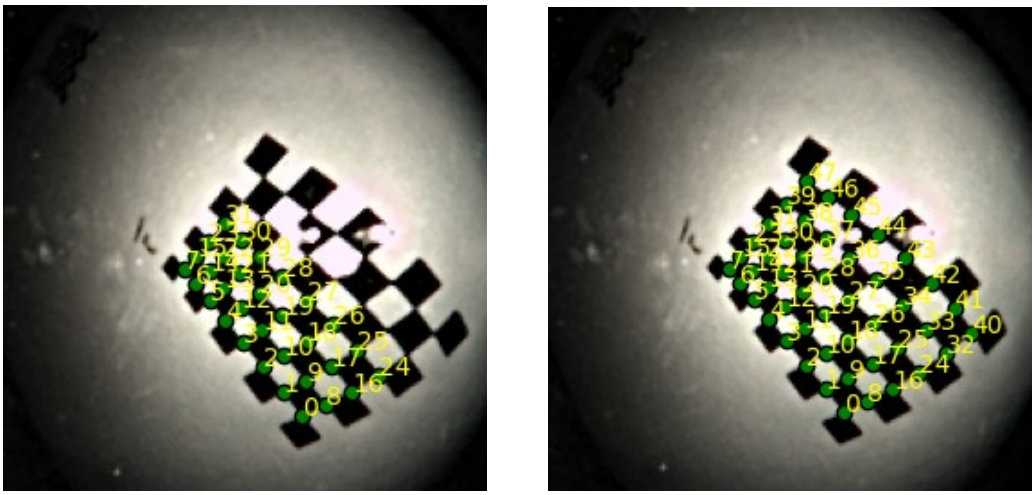
When looking at the results obtained from the real world data, both the thermal infrared and the multispectral images, we observe significant variations. However, the Geiger method with the proposed Gaussian process enhancement consistently demonstrates the best overall performance. The results are presented in Table 6.3.

The checkerboards in the IR dataset originally have white checkers and dark borders, in other words the checkerboards in the images look like the inverse of regular checkerboard targets. Therefore we performed two tests on the IR dataset: with the original images and with images whose grayscale was inverted so the checkerboards look like regular checkerboards. We notice a substantial disparity between the IR and IR inverted datasets, indicating that the performance of both OpenCV methods is adversely affected when images are not preprocessed beforehand. Upon inverting the infrared images, we observe improvements across all methods, with the most significant enhancement occurring in the case of the OpenCV method, resulting in an eightfold increase in the amount of detected corners.

Furthermore, a noteworthy observation can be made when analysing the results of the MSI on the large dataset. Surprisingly, the Geiger method outperforms the others, even though the simulated dataset results indicated that it struggled with increasing augmentations compared to the alternative methods. The application of Gaussian process enhancement further bolsters the detection of checkerboards, particularly in the case of large checkerboards (14 x 9). This substantial increase can be attributed to the higher chances of missing a corner in larger checkerboards. While a similar trend is observed with small checkerboards, the OpenCV method performs relatively better but still falls short of the performance achieved by the Geiger method with the proposed Gaussian process enhancement.

6.3.5 Use Case: Endoscopic Camera Images

As a final demonstration of our method, we describe the use case of working with images of calibration patterns taken to calibrate a Pillcam COLON2 double endoscope camera capsule. Several difficulties arise when calibrating this device. The resolution is low (320x320), the warping of the image is significant (172° field of view), and even the most minute artefacts in the glass casing are visible in the image. Detecting a checkerboard with the Geiger method fails, as not every corner is detected. This results in a significant portion of the corners not being used in the checkerboard.



(a) (b)

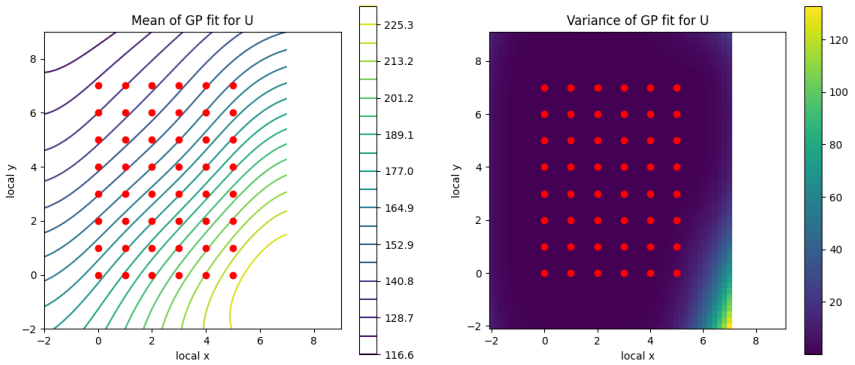
Figure 6.7: An example of a low quality image of a checkerboard detected by (a) Geiger and (b) Geiger plus the Gaussian process enhancement.

The current workaround is to add them manually, which is labour-intensive. Our method can infer the missing corners. We demonstrate this on images in the work of [111], taken from their accompanying online GitHub repository. An example can be seen in Figure 6.7. This image of a checkerboard is of very low quality, which resulted in several occlusions. The upper right part can no longer be fitted in a checkerboard by the Geiger method. Our Gaussian process method expands the initially found checkerboard to include other corners and predicts corner uv-coordinates for occluded corners as well. Moreover, the entire checkerboard is refined.

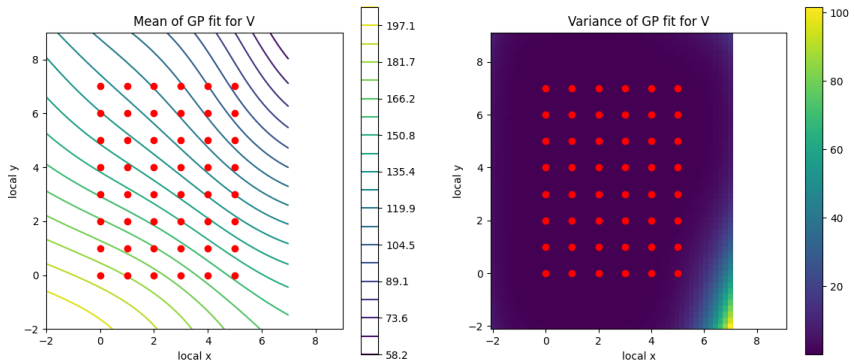
The Gaussian process prediction is accompanied by an uncertainty estimate under the form of a posterior variance. In other words, every predicted corner location consists of both a mean and a variance. This variance allows for an assessment of the quality of the prediction. The posterior distribution for this use case can be seen in Figure 6.8. Notice how the variance increases far away from the data.

6.4 Discussion

In this research, we are working with the mean of the posterior distribution of the Gaussian processes (the predicted uv-coordinates in an image for a given local xy-coordinate on a checkerboard). This tends to smooth out the results [130]. A squared exponential kernel is implemented, resulting in infinitely differentiable and thus smooth functions. The overall result is a checkerboard that has been smoothed out. This can be seen as removing noise or jitter on the found corners. We exploit this feature as an extra final step of corner refinement. However, caution is advised when working with heavily warped checkerboards. When working with GPs, one could overly smooth out the corners in cases of heavily warped checkerboards or images from fisheye lenses. This could be addressed with more complex kernels or even deep Gaussian processes that will be investigated in future work.



(a)



(b)

Figure 6.8: The posterior distribution for the prediction in Figure 6.7 for the (a) u-coordinate and (b) v-coordinate.

Another way to look at this, is the insight that a GP will adjust the coordinates for a single corner based on the coordinates of all other corners. In a GP, all data points, in our case corners, are assumed jointly Gaussian [130]. This means they are not independent of each other. They co-vary. Other refinement methods are only based the values of neighbouring pixels, not the whole (possibly warped) grid. The justification for this is the fact that the underlying truth is a regularly spaced rectangular pattern.

The Gaussian process prediction is accompanied by an uncertainty estimate under the form of a posterior variance. In other words, every predicted corner location consists of both a mean and a variance. This variance allows for an assessment of the quality of the prediction.

When making predictions using a Gaussian process, we not only obtain a mean estimate for the predicted corner location but also an associated posterior variance. This variance provides valuable information about the uncertainty in our prediction. Essentially, it allows us to assess the

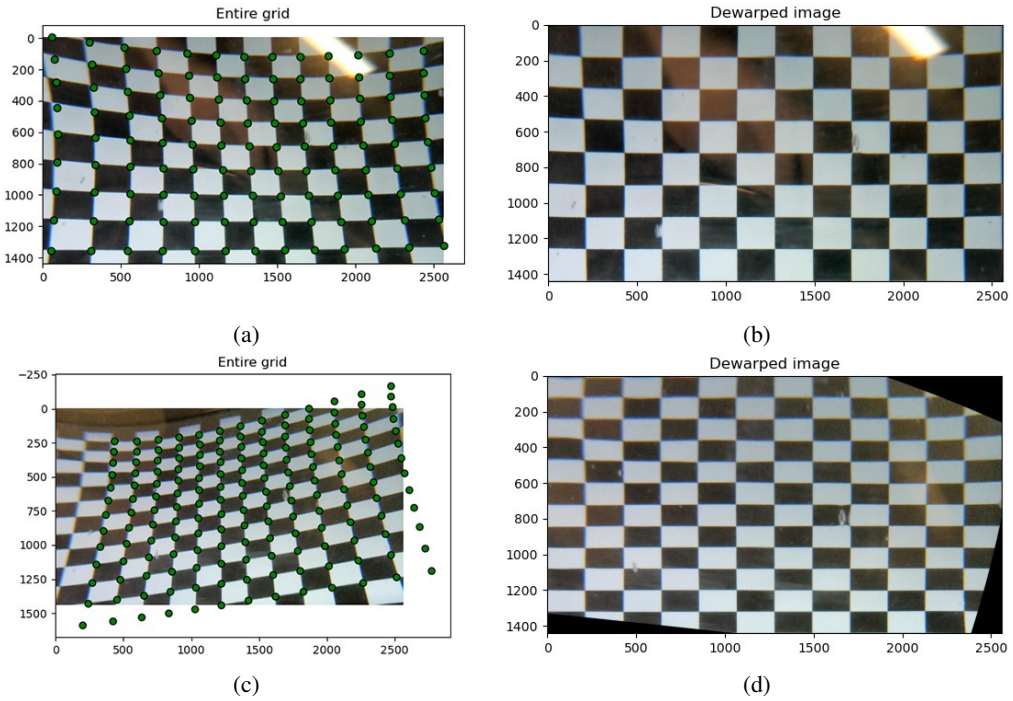


Figure 6.9: (a) A Gaussian process predicted the (green) corners of a checkerboard. Notice how the occlusions from the lighting are also filled in. (b) This model can be used to dewarp and frontalise an image. (c) A checkerboard with points beyond the image borders. (d) The dewarped and frontalised result, leaving pixels outside the original image black.

quality and reliability of the estimated corner position. Our software allows for the visualization of this under the form of contour plots. An example was given in Figure 6.8. These contour lines can also be used to estimate the over-smoothing of the Gaussian process.

The GP enhancement method can be used to interpolate and extrapolate point locations in between the corners and even outside the checkerboard assuming the curvature pattern can be described by the points in the board. This allows us to unwarp and frontalise the image. We simply predict the corresponding pixel uv-coordinate for a dense grid of xy-coordinates. A demonstration can be seen in Figure 6.9.

Even though Gaussian processes can perform well in low data regimes [130], some numerical instabilities and oddities might occur. For instance, when working with a slightly warped 3x3 checkerboard, there is not much information to build a solid statistical model on. The danger is that, from a Bayesian point of view, the corners could be explained by considering their coordinates as pure noise [88]. This results in patterns for the predicted checkerboard corners that simply do not make sense. In this case, our software allows for end-users to incorporate prior knowledge in the model by adjusting some of the options and hyperparameters in the software. In the code itself, we provide a lot of comments to guide the end-user. We suggest two things to try. First, put a lower limit on the length scales. This will automatically force the model to consider configurations in which the points are on a smooth grid, which in fact they are. Second, consider an upper level on the noise. This will force the model to stay close to the values of the detected corners. All of this is included in comments in the code.

6.5 Conclusions

In this chapter, we designed a new checkerboard detection pipeline, consisting of a checkerboard detector and a checkerboard enhancement with Gaussian processes. This research has shown that our Gaussian process enhancement is able to improve checkerboard detection results. By learning a mapping from local board coordinates to image pixel coordinates via a Gaussian process, we can fill in occluded corners, expand the board beyond the image borders, allocate detected corners that do not fit an initial grid and remove noise on the detected corners locations. We explained the role Gaussian processes play in enhancing the results. Lastly, we provide all our code open source in the form of a modular and easy-to-use Python checkerboard detection library called **PyCBD**. The findings will be of interest to people working on camera calibration, camera localisation, SLAM, stereoscopic vision, depth sensing and many others.

Chapter 7

How to Turn Your Camera into a Perfect Pinhole Model

As mentioned in the previous chapter, camera calibration is a first and fundamental step in many computer vision applications. Despite being an active field of research, Zhang's method remains widely used for camera calibration due to its implementation in popular toolboxes like MATLAB and OpenCV. However, this method initially assumes a pinhole model with sometimes oversimplified distortion models. In this chapter, we propose a novel camera calibration approach that involves a pre-processing step to remove distortions from images by means of Gaussian processes.

Our method does not need to assume any distortion model and can be applied to severely warped images, even in the case of multiple distortion sources, e.g., a fisheye image of a curved mirror reflection. The Gaussian processes capture all distortions and camera imperfections, resulting in virtual images as though taken by an ideal pinhole camera with square pixels. Furthermore, this ideal GP-camera needs only one image of a square grid calibration pattern to remove the distortion.

This model allows for a serious upgrade of many algorithms and applications that are designed in a pure projective geometry setting but with a performance that is very sensitive to non-linear lens distortions. We demonstrate the effectiveness of our method by simplifying Zhang's calibration method, reducing the number of parameters and getting rid of the distortion parameters and iterative optimisation. We validate by means of synthetic data and real world images. The contributions of this work include the construction of a virtual ideal pinhole camera using Gaussian processes, a simplified calibration method and lens distortion removal.

We presented this work at a poster presentation during the CIARP 2023 conference in Coimbra, Portugal. It is also published in [34].

7.1 Introduction

Camera calibration is a vital first step in numerous computer vision applications, ranging from photogrammetry [92] and depth estimation [102] to robotics [81] and SLAM [145, 39]. As such, it is still a very active field of research, resulting in a myriad of calibration techniques [10, 18, 131, 169, 126]. In this chapter, we use the term *camera* for systems such as multi-camera systems or catadioptric systems [55] which also include a mirror.

Several attempts have been made to unify the calibration procedures for different camera types and camera systems [127, 66, 93]. However, the most popular method in practice today is still Zhang's method [168], which is the basis for the camera calibration toolboxes of both MATLAB and OpenCV. This method assumes a pinhole model with additional lens distortions. The resulting calibration is a compromise between all intrinsic and extrinsic values, including the distortion model parameters. This approach has several drawbacks. First, the pinhole assumption is not applicable to non-central cameras. Second, the calibration is the result of a converging optimisation process in which one parameter is adjusted in favour of another one to obtain a better optimum, without actual justification. Third, the proposed distortion models for radial and tangential distortion are in some cases oversimplifications, for instance when the distortion is not perfectly radially symmetric or when the centre of distortion is not at the principle point of the camera.

In this work, we propose a new approach in which we first perform a pre-processing step on the images to remove all distortions. Next, the undistorted images serve as input for a simplified version of Zhang's method for perfect pinhole cameras. By distortions, we mean all deformations resulting from lenses, camera hardware imperfections, faults in the calibration board and even noise. To capture these, we rely on Gaussian processes [130]. They are a non-parametric Bayesian regression technique that are very well suited to handle sparse noisy datasets.

The proposed method applies to a variety of 2D-cameras. For any such camera, we train a Gaussian process on the relationship from pixel coordinates of the corners detected in an image of a square grid pattern (e.g., a checkerboard) to a perfectly spread square lattice of virtual 2D points. Only one image of the calibration board is needed for this training. This lattice can be seen as the non-linear projection of the checkerboard corners in the original camera image to a virtual image plane, consisting of virtual pixels. The Gaussian process captures all possible distortions.

All future images can now be mapped to the same virtual image plane. As all distortions are removed, we are left with virtual images as though they were taken by an ideal pinhole camera. This method does not need to assume any distortion model and can be applied to severely warped images, even in the case of multiple distortion sources, e.g., a fisheye image of a curved mirror reflection.

The process of first taking images by the given camera followed by the Gaussian process mapping to this virtual image plane can be considered as acquiring images by a virtual camera, called the *GP-camera*, replacing the original camera. We will validate that the imaging by this GP-camera corresponds to a perspectivity (central projective transformation) from the 3D scene to the virtual image plane. In other words, we prove that the GP camera is a pinhole camera.

The main benefit of obtaining an ideal pinhole camera is that a lot of well-studied algorithms and applications can be employed on its images. These include pose estimation, depth estimation, epipolar geometry, shape from motion, 3D scene reconstruction, optical flow, externally calibrating multiple cameras and other 3D sensors, etc. Many of these assume a central projection. For a treatise on these topics, we refer to the book [66]. Our model allows for a serious upgrade of these algorithms and applications that are designed in a pure projective geometry setting. Their performance is very sensitive to non-linear lens distortions. In particular, the quality of calibration techniques that lean on sphere images is drastically improved when rectified images with square pixels are available [147, 117, 116].

In [128], Gaussian processes are also used to model lens distortions. However, they serve as a surrogate model for the function that captures the lens distortion. As such, they are still part of the iterative camera calibration process. Lens distortion based on one checkerboard pattern is pro-

posed in [164]. However, they implement the Levenberg–Marquardt algorithm to find an optimal set of parameter values for their distortion models. Gaussian processes are non-parametric and as such do not depend on this. A deep learning variant of this can be found in [166].

The contributions of this works are:

- We explain how to construct a virtual ideal pinhole camera out of a variety of non-pinhole cameras using Gaussian processes.
- We show that our calibrated GP-camera using a simplified version of Zhang's method leads to more accurate measurements compared to the calibrated original camera using the general Zhang's method with iteration.
- We show how our method can be used to remove heavy distortions in images.

The rest of this chapter is structured as follows: Section 7.2 provides the construction and operation of a virtual GP-camera, and describes how we will validate this pinhole model. In Section 7.3 we show the results and compare our method to the MATLAB implementation of Zhang's method. We discuss these results in Section 7.4. Finally, we formulate our conclusions in Section 7.5.

7.2 Methods

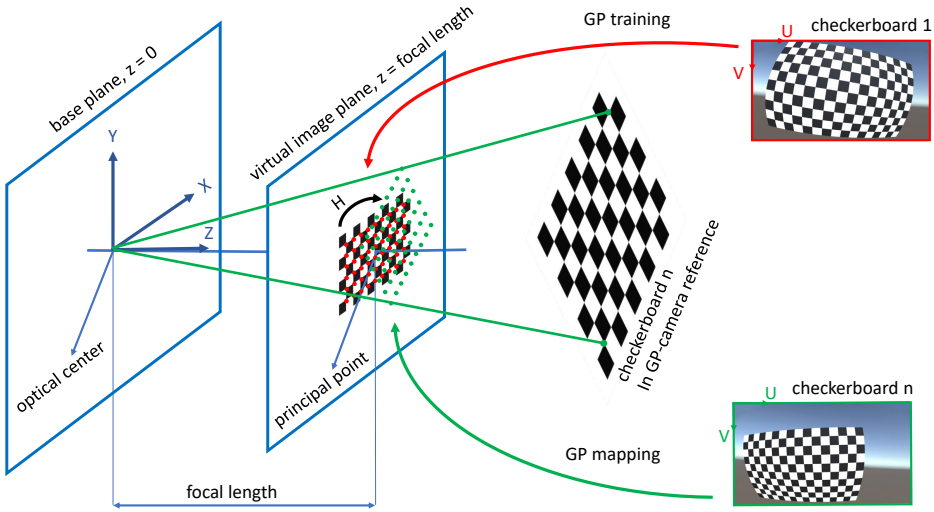


Figure 7.1: An overview of our method. A GP model maps the found corners to the image plane, capturing all lens distortions and camera and checkerboard imperfections.

7.2.1 Constructing an Ideal Pinhole Camera

Every pixel in an image taken by a camera corresponds to a ray of incoming light, which is a straight line. This means that all points on this straight line in 3D space are mapped to the same pixel. For a perfect *pinhole camera*, as described in [168], all these lines intersect in a central point called the optical centre. Consequently, for the pinhole model, taking images corresponds to a perspective projection, which is an important example of a projective map from $\mathbb{P}^3$ to $\mathbb{P}^2$. The image plane is perpendicular to the focal axis, which contains the optical centre and pointing in the direction in which the camera perceives the world. The distance between the optical centre and the image plane is called the focal length. In Figure 7.1 we present a pinhole camera, where every checkerboard is projected to the image plane.

A myriad of calibration techniques for (pinhole) cameras can be found in literature. Due to its importance in computer vision, this is still a very active field of research. Here, we restrict ourselves to the methods most commonly used in practice, such as the camera calibration toolboxes in MATLAB and OpenCV. Their implementation is based on the well-known Zhang's method [168], which is based on the Direct Linear Transform (DLT) method, with the calibration points located in planar objects such as flat checkerboards. See [66, 16] for a more in depth treatise. We provide an overview of Zhang's method in Appendix A.

To account for distortion, Zhang's method first assumes a perfect pinhole model with no distortion at all, and approximates the calibration matrix K by means of several homographies between checkerboard positions and the image plane. These homographies are used to determine the positions of these boards relative to the camera. These camera intrinsics and extrinsics serve as an initial guess for an iterative process in which the distortion model is integrated. A non-linear optimisation process is then implemented to find, after convergence, values for both the intrinsics, extrinsics and the distortion model parameters.

Herein lies the main pitfall of this method. By iterating towards a convergence in the parameters values, there is a compromise between them. This means unjustly altering the value of one parameter in favour of another one. Both the pinhole and the distortion model might be an oversimplification of the underlying reality.

Now we will explain how we can construct an ideal pinhole camera by using Gaussian processes that first remove all factors that make the pinhole assumption invalid. What is left, is a virtual (ideal) pinhole camera for which a simplified Zhang's method can be used.

Using a fixed but arbitrary 2D-camera, physical or simulated, we take the image of **only one** checkerboard, or any planar square grid pattern. We introduce a local xy-reference frame on the board plane, with coordinate axes parallel to the grid lines, the unit equal to the square edges and the origin typically coincident with some grid corner. This board can be placed anywhere in 3D space, but for our purposes, it is best to position it in such a way that it fills up the entire image. Once this is done, we define a virtual image plane where the grid squares are the virtual pixels. We detect the grid corners in the image using any corner detection system, e.g., the MATLAB Camera Calibration Toolbox. We assign to every detected corner its local xy-coordinates on a virtual image plane. As a consequence, the original board image is mapped to a perfect square lattice of points in the virtual image plane.

These corresponding sets of data are used to train a Gaussian process model for a map between the uv-pixels in the original image plane and the xy-coordinates of the virtual image plane, explained in Section 7.3. In practice, we implement two independent scalar output Gaussian processes: one

for the resulting x-coordinate and one for the y-coordinate.

In summary, for a given 2D-camera and the image of some spatially positioned square grid patterns, we have constructed a virtual GP-camera that obtains its images by mapping the real images to the virtual image plane by means of a Gaussian process.

Although the Gaussian map of this GP-camera was trained on a single checkerboard image, it apparently removes the distortion for any image of any spatial object. In Section 7.3 we investigate the images for many positions of the calibration board and observe the straightness of all the GP-images of the grid lines. In other words, the GP-camera maps every plane (board position) as a *collineation*, which must be a *homography* according to the fundamental theorem of projective geometry [138]. We conclude that the GP-camera images the world as a projective transformation.

Furthermore, this projective transformation is a *perspectivity* (central projection) since it can be described by the multiplication by a *projection matrix* $\mathbf{P}$ that can be decomposed as:

$$\mathbf{x} \sim \mathbf{PX} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{X}, \tag{7.1}$$

with $\mathbf{K}$ the upper triangular intrinsic calibration matrix, $\mathbf{R}$ a rotation matrix and $\mathbf{t}$ a translation vector. We work with homogeneous coordinates $\mathbf{x} = (x, y, 1)^T$ for points in the virtual image plane and $\mathbf{X} = (X, Y, Z, 1)^T$ for spatial points.

In Section 7.3 we determine the calibration matrix $\mathbf{K}$ by a simplified version of Zhang's method, described in Appendix A. It is important to note that we need only three parameters for this. Since we are working on square pixels, there is no longer any skewness and the scale is the same in the $x-$ and y-direction:

$$\mathbf{K} = \begin{pmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{pmatrix}. \tag{7.2}$$

The extrinsic component $[\mathbf{R} \mid \mathbf{t}]$ in Equation 7.1 transforms the coordinates of the world reference frame to those in the GP-camera reference frame. The GP-origin is determined by the principal point (u_c, v_c) and the focal length f, which is measured in virtual pixel units. The X- and Y-axes of the latter are parallel to the corresponding axes of the virtual image plane as they appear on the checkerboard in its first position. The Z-axis is perpendicular to this, yielding the equation $Z = f$ for the GP image plane.

We validate this pinhole model for the GP-camera by showing the almost perfect match of $\mathbf{K}$ in Equation 7.2 as determined by the equations of Zhang's method for many board positions or homographies. In addition, we observe very small reprojection errors in the pinhole model of the GP-camera, using the relative positions of these boards as provided by the computed homographies (Section 7.3). Everything that makes our real world camera deviate from an ideal pinhole camera is captured by the Gaussian process model. This entails lens distortions, imperfections in the lens, camera, checkerboard and even noise in the image.

Although the GP-camera is a virtual camera, we have a geometric interpretation of its pinhole model. The focal axis of the virtual camera is perpendicular to the square lattice board in the position of the first reference image. This lattice board is the virtual image plane, having world pixels equal to the square cells. The focal length is measured from the GP-centre to this virtual image plane in the same grid units. The images of the GP-camera are obtained by a central projection from this centre onto the virtual image plane, as shown in Figure 7.1. These dimensions

Dataset	Nr of checkerboards	Nr of corners	Image resolution
Unity pinhole	30	15x9	3840x2160
Unity barrel	30	15x9	3840x2160
Unity pincushion	30	15x9	3840x2160
Webcam	11	15x9	2560x1440
Webcam with telelens	17	14x9	2560x1440
RealSense with mirror	26	14x9	971x871

Table 7.1: The datasets

of the virtual camera are fixed once and for all from the moment the first picture is taken. Furthermore, they are linked to the physical camera, even if it is later on moved to a different position and rotation. As our virtual camera represents a pinhole model, there is a bijective relationship between the real world reference system and the reference system of the virtual camera. Moreover, this relationship is a *similarity*, which consists of a rotation, translation and a scaling. The latter depends on the unit chosen for the perfect virtual grid of corners. This similarity is easily found by correspondences between the two systems.

The calibration of the GP-camera by the simplified Zhang's method not only validates the pinhole model with square pixels, it also provides an interesting alternative to camera calibration, clearly outperforming state-of-the-art methods with respect to simplicity and accuracy (Section 7.3).

7.2.2 The Datasets

We validate our findings on six diverse datasets of images of checkerboards. An overview is given in Table 7.1. The first three datasets are generated in a scene made in the Unity game engine software version (2020.2.5f1). These sets are based on the same scene, so they depict identical positions and rotations for the boards. The barrel distortion and pincushion distortion effect is obtained by the built-in post-processing package. The barrel distortion centre was placed in the centre of the image. The centre of the pincushion distortion is shifted to the left and bottom of the image.

The last three datasets are made with real world cameras. The Webcam is an Avalue 2k Webcam and the telelens is an Apexel Telelens x2. The RealSense is of the type D415. This can be seen as catadioptric system, as the camera is pointed at a mirror with unknown curvature. The centre is relatively flat while the edges show more spherical bending. There is no mathematical model to calibrate this system. In Figure 7.2 we depict an example of one board out of every dataset in the first column.

7.3 Results

We validate our findings with three assessments. First, we investigate how well the Gaussian processes predict collineations, meaning predictions for corners of a checkerboard result in rows and columns that are straight lines. Second, we calculate the reprojection error for found 3D coordinates of corners of the used checkerboards. Finally, we demonstrate the removal of distortion.

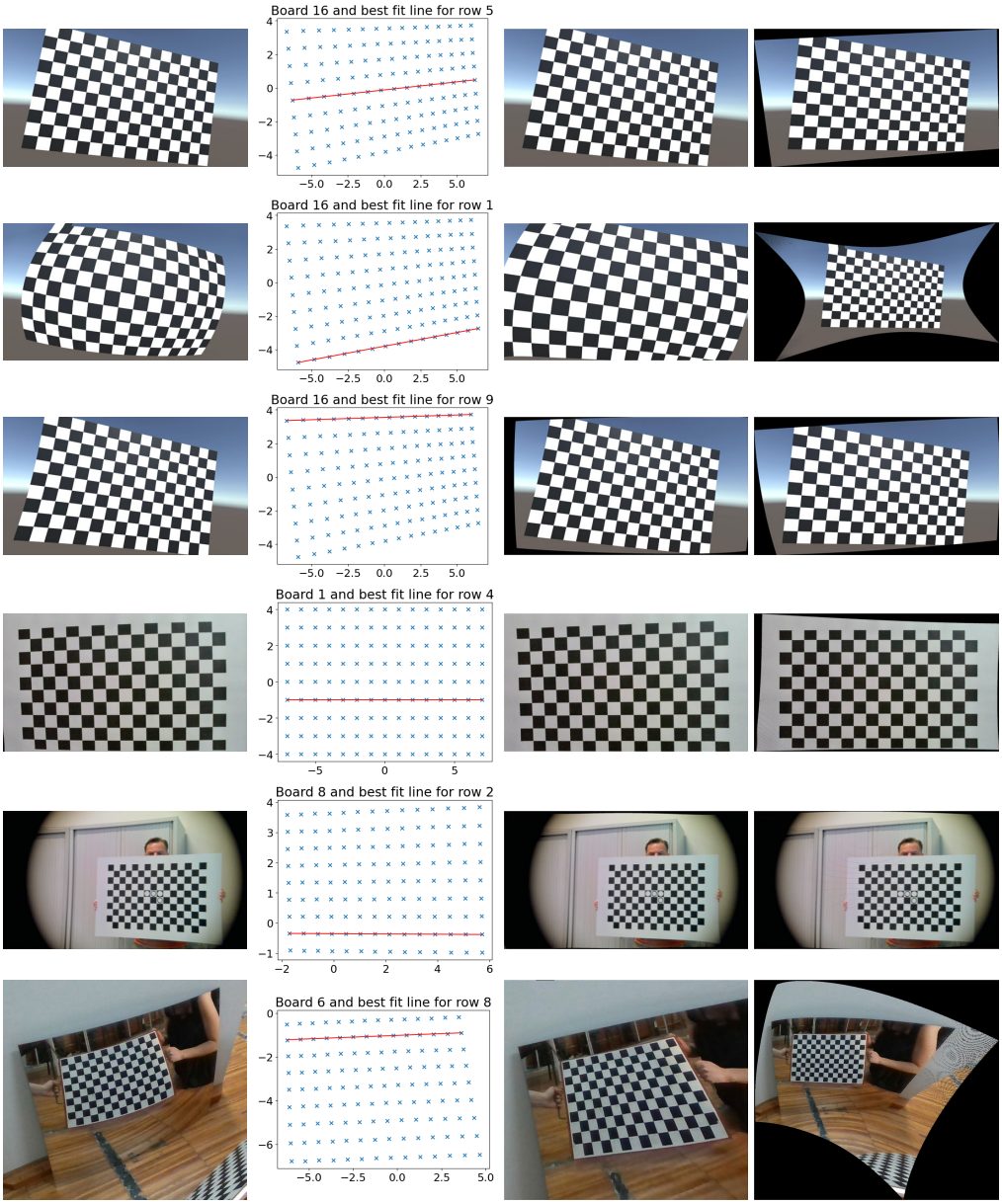


Figure 7.2: Column 1: an example of one board of the datasets of Table 7.2. Column 2: the collinearity check on the result of the GP for that board. Column 3: undistorted image by MATLAB. Column 4: undistorted image by Gaussian processes (our method).

7.3.1 Collineation Assumption

In this section, we validate our method by showing that the mapping done by our GP-based algorithm from uv- to xy-coordinates is a projective transformation where any straight line fed into the said mapping remains straight. In other words, collinearity of points is preserved. We prove the aforementioned statement both visually in Figure 7.2 and quantitatively in Table 7.2.

Let the grid formed by the corner points of a checkerboard have several rows and columns of multiple points each. We find the best fit lines through all of these individual rows and columns. Subsequently, we calculate the perpendicular distance of each point from the corresponding best fit line. The Root Mean Square (RMS) of these perpendicular distances for each row/column is divided by the distance between the end points of the respective row/column to obtain the unitless version of this RMS value. Finally, these scaled unitless RMS values are averaged across all the boards in each dataset to get the Average Root Mean Square Collinearity Error, which is abbreviated as GP CE in the third column of Table 7.2. For visual confirmation, we show one such best fit line for a given board in every dataset in the second column of Figure 7.2.

It is worth mentioning that the positions of the corners on the virtual image plane should be interpreted by the projection of the virtual Gaussian process camera, whose location and rotation differs from the real (or Unity) camera. This depends on the image of the first checkerboard (see also Section 7.2.1). In other words, they are not just unwarpings of the original image.

7.3.2 Reprojection Error

In our method, we calculate a camera matrix $\mathbf{K}$ for our GP-camera such that for each board, the corresponding homography $\mathbf{H}$ has the form $\mathbf{H} \sim \mathbf{K}[\mathbf{r_1} \mid \mathbf{r_2} \mid \mathbf{t}]$. Furthermore, $\mathbf{K}$ has the form

$$\mathbf{K} = \begin{pmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{pmatrix}. \tag{7.3}$$

When solving the overdetermined linear system of equations (see Appendix A), we observe a small value for the smallest singular values, providing an algebraic validation that our method results into a pinhole camera.

To interpret this from a geometrical point of view, we reproject the 3D coordinates of the corners to the image plane using the parameters found in the camera calibration process. Next, we compare those to the Gaussian process predictions and the found uv-coordinates of the corners for our method and the MATLAB Camera Calibration Toolbox (version R2023a) respectively. As done previously, we calculate the RMS error for these values and scale them by dividing by the distance between two corners, making them unitless. These errors are given in Table 7.2. From this table, we can see that our method outperforms the MATLAB Camera Calibration Toolbox, especially for datasets with severely warped images such as the Unity with severe barrel distortion, the Unity with eccentric pincushion distortion and the RealSense with curved mirror. This is due to the fact that the MATLAB Camera Calibration Toolbox is based on a model that oversimplifies these underlying realities.

Dataset	GP CE ($\times 10^{-4}$)	GP RE	MATLAB RE
Unity pinhole	0.954	0.1197	0.1229
Unity barrel	1.334	0.1410	1.0315
Unity pincushion	1.062	0.1243	0.3879
Webcam	3.077	0.2478	0.2666
Webcam with telelens	3.903	0.2576	0.2667
RealSense with mirror	9.455	0.4404	0.5376

Table 7.2: Collinearity Errors (CE) and Reprojection Errors (RE)

We demonstrate the *pinholeness* of our GP-camera further by visualising a grid of 10x10 lines that accompany a given set of pixels. For a pinhole camera, all lines intersect the optical centre, which is also the centre of the reference frame. This implies that all our pixels should correspond to straight lines going through the origin of the reference system. First, we define a 10x10 subset of pixels. We know $\mathbf{R_n}$ and $\mathbf{t_n}$ for each of the n checkerboards from the camera calibration. For each xy-coordinate of the 10×10 pixels, we calculate a corresponding 3D point on each checkerboard. Next, we group together coordinates of 3D points that belong to the same pixel. On these grouped points, we perform a least squares best fit line [90], including RANSAC. An alternative mechanical-inspired approach is given in [114]. We visualise these lines in Figure 7.4. Notice how, for instance, the barrel distortion manifests itself as a warping of the grid of lines, while retaining the pinhole model.

7.3.3 Distortion Removal

The trained GP predicts a new location in virtual xy-coordinate frame for every pixel uv-coordinate frame of the original distorted image. Based on those virtual pixel values, we distil a new image. The predicted coordinates are non-integer numbers, which we round to an integer value. This rounding could imply that some pixels are left empty (black). This means no original pixel is mapped to that specific virtual pixel. An example of this can be seen in Figure 7.3. This issue can be solved by implementing a median filter on the surrounding pixels. Alternatives to this exist, but are outside the scope of this chapter.

The resulting undistorted images are shown in the last column of Figure 7.2. Notice how our method is better equipped to handle distortions, as it is not limited to an underlying oversimplifying distortion model. The difference is most notable for severely warped checkerboards, such as the ones in the second and last row of Figure 7.2.

7.4 Discussion

The fact that collinearity is preserved when the GPs map uv-coordinates of found corners to the virtual image plane proves the projective transformation between the real world (including Unity) checkerboards and their virtual images. This means there is an almost perfect homography between two virtual images of two checkerboards, validating our approach. All non-linearities are captured by the GPs. The decomposition of the projective transformation for every board contains the same $\mathbf{K}$. We make use of this to calculate a line in 3D space for a given pixel. We observe that every line that corresponds to a pixel goes through the origin of the reference system of the GP-camera. This demonstrates the pinhole behaviour of our GP-camera.

An accurate corner detection algorithm is a crucial first step in our approach. Especially for the corners of the first checkerboard, on which the Gaussian processes are trained. Moreover, the checkerboard should consist of a sufficient amount of corners so that the distortion can be fully captured.

The reference system of the GP-camera is different from the real (or Unity) camera. This is the result of how the Gaussian process is trained. All uv-coordinates of corners are mapped to a regular square grid. This means the first board is perpendicular to the optical axis and all rows of points are horizontal. This can be seen in the last column of Figure 7.2.

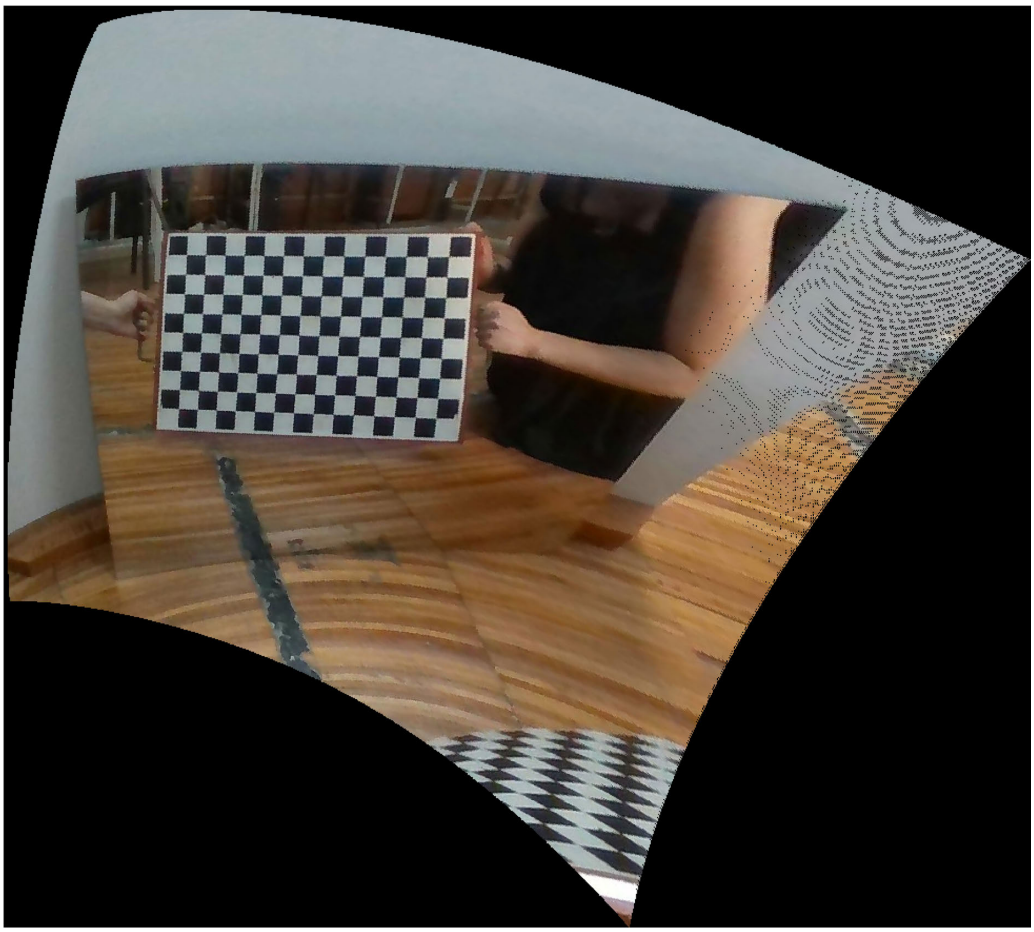


Figure 7.3: Example of missing (black) pixels after distortion removal. The effect is most notable in the top right corner. Not all pixels receive a value from the GP predictions.

From these images we can also observe that for the severe barrel distortion and the eccentric pincushion distortion (row two and three), the MATLAB method fails. Our method is more flexible and can capture these. Notice how the undistortion in the lower left region is better in the last image of row three.

Our Gaussian process predictions decrease in reliability as we move away from the region of the corners in the image of the first board. For those points, the Gaussian processes fall back on their prior, accompanied by large uncertainties. The latter can be taken into account. We retained the pixel predictions with a large uncertainty in the last image of the last row of Figure 7.2 for demonstration purposes. Notice that the results become meaningless far away from the first checkerboard where the GP was trained on.

A fallacy of working with Gaussian processes is that they sometimes smooth things out too much. Especially when working with a squared exponential kernel. This is an issue for example, when working with fisheye images, where a lot of corners are situated at the edge of the image and only a few are at the centre of the image. This data is in essence non-stationary. It varies more in some regions (the edges) than in others (the centre). A solution for this is to work with more corners, a

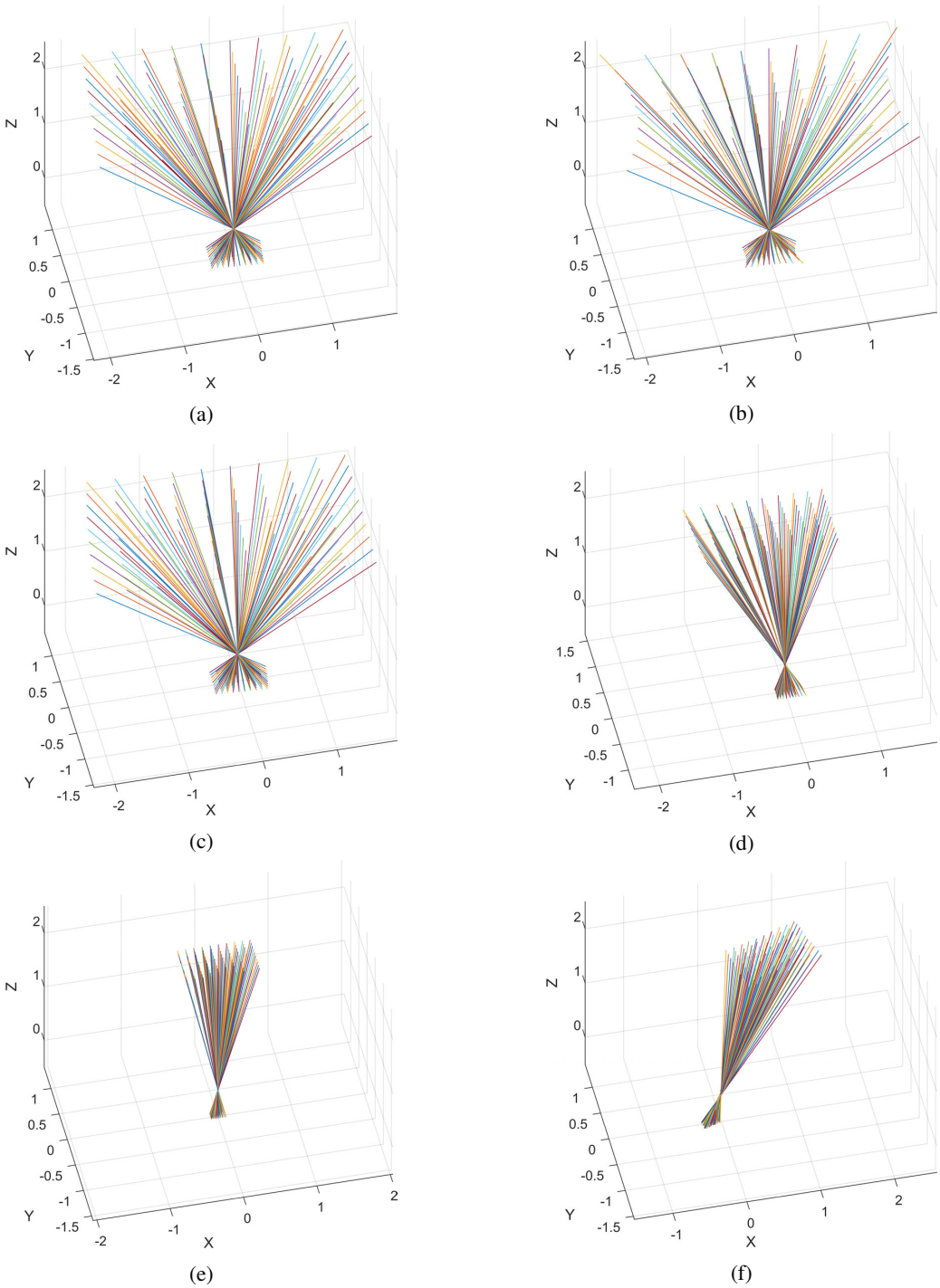


Figure 7.4: Visualisation of the pinhole results for each of the six datasets. Only (a) and (d) are actual pinhole cameras, in Unity and the real world respectively.

different kernel or even active targets with Gray code instead of corners [141].

An interesting question is how the Gaussian process prediction uncertainty for the corners propagates to the uncertainty of the homographies and the camera model parameters themselves. For a treatise on the subject of uncertain projective geometry, we refer the reader to the book [52]. We will address this in upcoming work.

7.5 Conclusion

The aim of the work in this chapter was to construct a virtual ideal pinhole camera out of a given camera (including catadioptric systems with mirrors). We showed how this is possible by means of Gaussian processes, which capture everything that makes the camera deviate from an ideal pinhole model. This includes lens distortions and imperfections. Experiments confirmed that our approach results in a pinhole camera.

Further work is required to establish the benefit of our approach in real world camera calibration and compare them to other state of the art methods. We will address this in upcoming publications, as there exists a myriad of camera systems and likewise calibration procedures.

Our model allows for a serious upgrade of many algorithms and applications that are designed in a pure projective geometry setting but with a performance that is very sensitive to non-linear lens distortions.

Part IV

Visuospatial Neglect

Chapter 8

Assessment and Treatment of Visuospatial Neglect

In this chapter, we focus on a 3D application that exploits the uncertainty that accompanies Gaussian process predictions. By doing so, we can learn a function or mapping over the input space in a very efficient way. The trick is to only perform measurements at the locations of the input space that provide us with the most information. These are the locations where the uncertainty is largest. This method is called *active learning*. Other deterministic regression techniques to learn this function could be used (e.g., neural networks), but they do not have the property of an uncertainty estimate over their predictions. As such they can not perform active learning.

We implement this method in a virtual reality application which can assess and treat a condition called visuospatial neglect. This disorder is characterised by impaired awareness for visual stimuli located in regions of space and frames of reference. It is often associated with stroke. Patients can struggle with all aspects of daily living and community participation. Assessment methods are limited and show several shortcomings, considering they are mainly performed on paper and do not implement the complexity of daily life. Similarly, treatment options are sparse and often show only small improvements.

The benefit of using active learning is the reduction of the effort it takes a patient to undergo an assessment. Less measurements are needed. Our application tracks a patient's eye movement and records the time it takes him or her to find a certain stimulus in a 3D virtual world. We describe how this model can be utilised in patient oriented treatment and how this opens the way to gamification, tele-rehabilitation and personalised healthcare, providing a promising avenue for improving patient engagement and rehabilitation outcomes.

To validate our assessment module, we conducted a clinical trial involving patients in a real-world setting. We compared the results obtained using our assessment with the widely used conventional visuospatial neglect tests currently employed in clinical practice. The validation process serves to establish the accuracy and reliability of our model, confirming its potential as a valuable tool for diagnosing and monitoring visuospatial neglect. Our VR application proves to be more sensitive, while intra-rater reliability remains high.

We published the findings of this chapter in [33].

8.1 Introduction

Visuospatial spatial neglect (VSN) is a cognitive disorder characterised by a lateralised attention deficit with reduced attention towards the contralesional hemispace and increased capture of information in the ipsilesional hemispace [165, 67, 49, 38]. VSN is a heterogeneous disorder that can vary both in regions of space and in frames of reference. It is often one of the deficits associated with stroke [105]. Approximately half of stroke patients worldwide experience VSN within the first two weeks of onset, and at least 40 percent of patients still experience symptoms a year post-stroke [106]. VSN has a significant impact on postural control, mobility, independence during daily-life and community participation after stroke [13, 46].

Accurately diagnosing VSN in clinical practice is thus highly relevant. However, current assessment methods for VSN are limited in this ability and show several shortcomings, including a lack of ecological validity, reliability, and distinctiveness between the different types (e.g., peripersonal, extra-personal) and severities of VSN [31, 40, 7]. Ecological validity pertains to the extent to which the evaluation setup faithfully replicates the real-world working conditions of the patient. It focuses on the accuracy with which the evaluation mirrors the aspects of the context in the real world.

Moreover, patients are often suffering from fatigue and a decreased attention span. Similarly, the current treatment options are sparse, while the effects are often small [9]. We elaborate on this in depth in Section 8.2.

As a result, patients remain highly dependent on the spontaneous recovery of the neural system because of the small treatment effects currently seen in clinical practice. However, a significant number of patients will have persistent VSN after rehabilitation, leading to substantial loss of community participation with a high dependency on (in)formal care.

Therefore, the overall primary objective of this work is to deliver a solution for some of the aforementioned problems by means of a Gaussian process (GP) based virtual reality (VR) application. The GP allows us to assess and treat VSN in a ”smart” way, while VR provides a three-dimensional simulation of a real-life environment, potentially increasing ecological validity. In short, we place stimuli in different virtual environments (both peri-personal and extra-personal) and measure the time it takes a patient to find them. From these measurements, we construct a heatmap of the patient’s VSN. The application tracks both eye- and head movements to gather data about visual scanning strategies and spatial attentional deficits.

Our assessment module is smart, in the sense that VSN will be mapped efficiently and accurately using AI. This effectively lowers the necessary amount of stimuli that has to be placed in order to acquire a viable assessment of a patient’s VSN, resulting in shorter assessment sessions. In practice, patients suffering from VSN and possibly other pathologies as well, are in no condition to spot dozens of stimuli in one time consuming session. Therefore, we rely on AI techniques that can handle small datasets. In this work, we implement active learning, as described by Pasolli in [113]. In the work of Holzinger [70], active learning is explained as a machine learning algorithm that can achieve greater accuracy with fewer training labels, if it is allowed to choose the data from which it learns. Starting with only a few measurements, the next location to place a stimulus in the virtual world is chosen on a criterion that maximises the gain in information. This results in reducing the overall number of measurements needed compared to measuring everywhere in a grid. We base our method on Gaussian processes [130]. This probabilistic machine learning technique makes predictions that are accompanied by an uncertainty. The latter can be exploited.

A more detailed explanation is given in Section 8.3.

Our application is human-centred as it continuously adapts itself to the input given by the patient itself. During assessment, the active learning algorithm chooses a new location in a patient's field of view to perform a measurement, i.e. choosing a location where to put a new stimulus. This location is based on how the patient performs. After every individual stimulus, the model is retrained with new information. After this training, a new location can be chosen. In this dynamic process, both the patient and the algorithm engage in a back-and-forth interaction, akin to a game of ping-pong.

Once the precise VSN characteristics of a certain patient is captured by our model, we can apply this in our treatment module. Exercises will be provided based on the results of the assessment module enhancing individualised specific treatment. More details on this treatment module are given in Section 8.4.2.

Besides the technical benefits of using both machine learning (GP) and VR described above, there is an added benefit. VR is a mature technology that requires only commercially available hardware that is reasonably priced. The latest generation of head mounted displays (HMD) are stand alone, meaning they do not require a powerful desktop PC or laptop and are cordless. This facilitates the adaptation by acute hospitals, rehabilitation centres, private practises, and even home users. The latter opens the possibility for tele-rehabilitation. Our application allows for independent training of the machine learning model by the patient at home under supervision of a remote therapist.

The rest of this chapter is structured as follows. In the next section we reference related work. In Section 8.3, we give some theoretical background on active learning. Section 8.4 describes how we implemented this in assessment and treatment. We present our results in Section 8.5 and we discuss these in 8.6. Finally, conclusions are provided.

8.2 Related work

8.2.1 Visuospatial neglect

In this section we describe the condition known as visuospatial neglect and reference to existing work on its assessment and treatment. We also explain the shortcomings of said methods that are still used in practice today.

The medical condition *stroke* is a condition defined by the World Health Organisation by rapidly developing symptoms and signs of a focal (or at times global) neurological impairment, lasting more than 24 hours or leading to death, with no apparent cause other than that of vascular origin. The number of stroke events in Europe is projected to rise from 1.1 million in 2000 to 1.5 million per year by 2025, largely due to the ageing population. The high burden on the community is the result of stroke being the leading cause of complex disability worldwide in adults (WHO). These people are often suffering from sensorimotor impairments in both the upper and lower limbs and the trunk. Furthermore, stroke survivors can have speech and comprehension problems, and cognitive impairments such as disorientation in time and space, decreased information processing time and volume, memory problems and attentional deficits. One of these post-stroke attentional deficits is visuospatial neglect (VSN) characterised by impaired awareness for visual stimuli lo-

cated on the contralesional side of space. In other words, patients do not pay attention to stimuli on the side opposite to the brain lesion (in hemispheric strokes). This results in problems with reporting, responding or orienting towards contralesional visual stimuli, which cannot be attributed to sensory or motor impairments alone. This means that VSN is located at the processing level of information rather than impairments at the input level (e.g., eyes, vestibular apparatus).

Spontaneous neurological recovery of VSN follows a natural logistic pattern of improvement in some patients, depending on severity and type of VSN, within the first 12 to 14 weeks post-stroke. Afterwards, the curve flattens and severity of VSN remains almost invariant, leaving at least 40 percent of patients with initial VSN still with symptoms a year post-stroke [106]. This is important since people with VSN experience significant postural impairments and a high fall risk. In addition, consequences can be more practical of nature as patients with attentional deficits are, for example, unaware of the traffic lights at street crossings or even traffic in general, but also lack the ability to find products at grocery stores. It is obvious that people dealing with cognitive impairments encounter difficulties in all aspects of ADL (Activities of Daily Living) and community participation, and can even lack the ability to live independently at home [107, 105, 13, 8]. With this in mind, research on VSN is crucial to improve assessment and explore new treatment options.

Although research in cognitive impairments is advancing, there are still some gaps in the literature considering assessment and treatment of attentional deficits such as VSN. Currently, VSN assessment usually consists of pen-and-paper tasks that are administered in a quiet room where distractions are minimal. Although these tests are easy to administer and assess underlying cognitive impairments such as VSN, they suffer from several shortcomings.

First, research has reported a lack of ecological validity [31]. Performances on pen-and-paper tests do not correlate well to ADL, which results in a poor understanding of the difficulties patients encounter in daily life [40]. These paper-and-pencil tasks do not fully capture the complex and dynamic demands of real-world ADL, where individuals need to navigate their environment and interact with objects. For example, finding stars between distractors on a piece of paper is not something we experience in our daily life, whereas having to search for objects in a 3D environment, such as in our VR application, is. Furthermore, these tests cannot differentiate well between different types of VSN (e.g., space within reach versus far space) and the exact severity of the deficit.

Second, standard assessment is sub-optimal since VSN is a clear three-dimensional problem and current assessment methods are two-dimensional, therefore lacking indeed ecological validity, reliability and distinctiveness between different types of VSN. Moreover, it has been stated that many patients with VSN are not even identified as a neglect patient and therefore not treated accordingly [7].

Third, in standard assessment, we do not have accurate information on eye and head movements which are important to assess the strategies or compensations the patient uses to complete the task. The information on both eye- and head movements will be imperative to have a good understanding of the (compensatory) strategies used by the patient. Furthermore, poor understanding of the severity and nature of the VSN derived from the standard assessment methods highly impacts treatment efficacy. Treatment options are sparse and effects are often limited. In literature, especially visual scanning training, active limb activation, prism adaptation training, and sustained attention training are the better options. However, these treatments are often delivered in clinical, controlled settings, which lack the complexity and stimuli of real-life environments. Bringing patients within such complex environments during rehabilitation (e.g., a leisure park or shopping

centre) is clinically not feasible due to associated health risks and costs for the health care system. The use of 3D VR may overcome these issues, and may deliver ecologically valid, patient-centred training whilst still being in a controlled and safe environment [135].

8.2.2 AI or VR Aided Assessment

In this section we zoom in on methods described in the literature that assess visuospatial neglect and are based on machine learning, or AI in general, VR or a combination of both.

In the paper by Dvorkin et al. [44], a subject was presented 3D spheres in a 3D virtual environment projected on a large screen. With the push of a response button, the subject could indicate the perceptance of the stimulus. Head movement was tracked with a special rolling shutter HMD. Linear regression models were implemented to map VSN. In comparison, our Gaussian process model allows for non-linear regression.

The work of Jang et al. [74] describes field of regard, field of view and attention bias measurement based on VR. Again, only head movement is measured while subjects search for visual stimuli in a virtual environment.

A fully developed VR game to treat stroke-induced attention deficits is described by Huygelier et al. in [71]. They implemented a dynamic difficulty adjustment (DDA) mechanic that tailors the experience to the needs of the patient. This serves two purposes. First, this allows the game to present high priority stimuli more frequently in the neglected field than in the good field. Second, this algorithm adjusts the difficulty of the game to an appropriate level for each player. Their DDA is based on a 2D Gaussian distribution, of which the mean is initialised at the centre of the visual field and is adjusted based on the median locations of missed targets at a fixed rate. This approach differs from our method, which is based in a Gaussian process, not a Gaussian distribution. The names for these are quite similar. In our work, the Gaussian process provides a Gaussian distribution for every possible stimulus location, not just for the overall field of view. A more thorough explanation is given in Section 8.3. Another example of an immersive virtual reality game to train spatial attention orientation after stroke, can be found in the pilot study by Huygelier [72].

In none of these works, the ecological validity was taken into account, except in the VR games. No eye-tracking solutions were developed. Moreover, no active learning strategies were implemented to reduce the number of stimuli placement iterations. We go in more detail on the topic of active learning in Section 8.3.2.

The paper by Saranti et al. [137] discusses the need for a *maybe* class in binary classification, where entities can exhibit a quantifiable tendency toward one of two opposites. The paper highlights the importance of human domain experts in marking entities and explaining the classification space. In our work, we leave the classification of sub-regions of the field of view of the patient into the labels neglect and no neglect to the clinicians themselves. They are only aided by the outcomes of our model.

8.3 Methods

8.3.1 Field of View

In this chapter, the inputs of the Gaussian process are points in the 2D space that represents a person's ideal field of view (without neglect). The outputs for this model are the recorded times it takes to find a particular stimulus. Neglected areas are given a maximum (truncated) value. We thus learn the mapping, or function, from a point in a person's field of view to search times.

We can visualise this 2D function as a heat map, as shown in Figure 8.1. A Gaussian process provides two values for each 2D coordinate in a patient's field of view: a mean (expected value) and an uncertainty (variance or two times sigma). We plot the mean on the left heat map and the uncertainty on the right. The axes represent the left-right and up-down direction in the field of view, expressed in degrees. The green colour in the mean heat map results from stimuli being found fast. The red and black are areas where stimuli are either not found or found only after a significant amount of time. As for the two times sigma plot, the green areas are locations where the uncertainty is low. This means stimuli have been placed there. Red areas are regions where no measurement has taken place and thus where uncertainty is high.

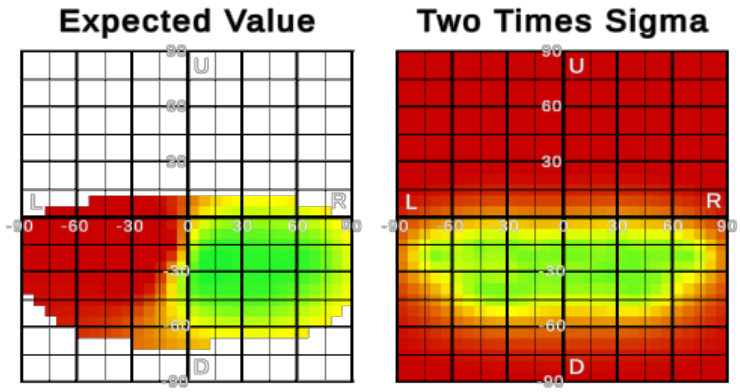
8.3.2 Active Learning

The process of mapping a patient's VSN accurately is time consuming. It depends on both the measurement of the search time for a given stimulus and the number of stimuli we want to present to a patient. The latter defines the resolution of the VSN situational map. The strategy to overcome this is to train a machine learning algorithm to predict these search time values for every possible point in a patient's field of view. The goal now is to train the model as accurately as possible given only a limited number of data points.

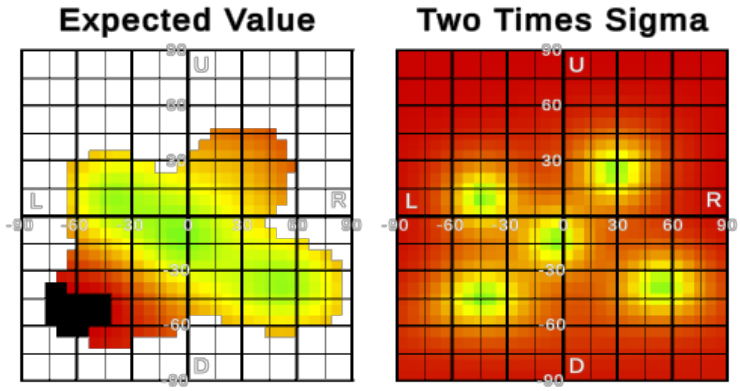
Our underlying model is a Gaussian process. It serves as a surrogate for the true VSN situation we wish to learn. By that, we mean the mapping from 2D points in a person's field of view to search times for those points. By placing stimuli at specific locations in a 3D virtual environment which correspond to 2D points in a person's field of view, we can make point measurements of this search time value for that specific point. In our application, we work with predefined locations, instead of a continuous range. We refer to these locations as spawn points.

Our active learning strategy consists of the following steps:

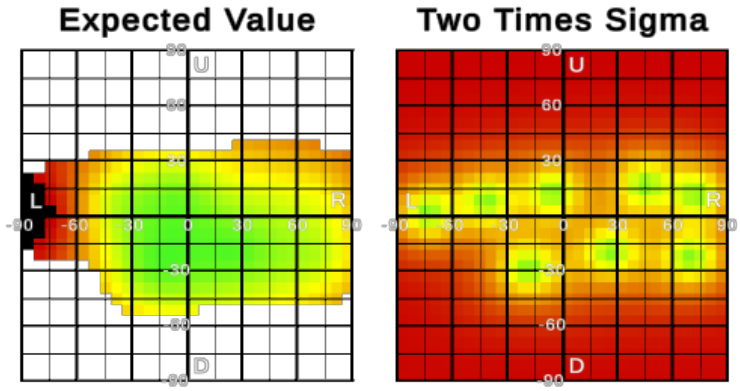
1. We start by presenting stimuli one by one at a small amount of spawn points. These are sampled at random from the list of all possible spawn points. Alternatively, those points could lay on a grid, be Latin hypercube sampled or chosen from a Sobol sequence [108, 48, 76].
2. A Gaussian process is trained on this initial small dataset.
3. The spawn point from the 2D field of view with the highest uncertainty (variance) in the GP's posterior distribution is chosen to place a stimulus next. This method is called Uncertainty Sampling (US). Alternatively, the spawn point which reduces the total variance of the posterior could be chosen. This method is called Integrated Variance Reduction (IVR).



(a)



(b)



(c)

Figure 8.1: Heatmaps depicting the VSN for a patient's field of view corresponding to the situations depicted in Figure 8.2 b, d and f respectively. These images are generated on the HMD itself. The resolution is kept low for performance reasons. Regions where the uncertainty is large (red in the right plot) are made white in the left plot.

(a) (b)

(c) (d)

(e) (f)

Figure 8.2: Screenshots capturing the view as seen through the HDM for the table (a), kitchen (c) and playground environment (e). Visualisations for the therapist showing a 3D view of a patient's neglect for those same environments (b, d, f). A green colour for the cubes represents short search times. A red colour indicates long search times. A black cube in the left part of (f) means a location where a stimulus was never found.

4. Step 3 is repeated until a certain criterion is met. When limited by the patient's condition, this could be a fixed number of iterations. Another criterion is convergence in the posterior distribution, which means that adding new data points no longer has a significant result on the predictions of the GP.

These steps are what is known as active learning by the machine learning community [130, 15, 113, 61, 110]. It is active in the sense that the learning is based iteratively on what is measured. This is in contrast to learning, or training, on an entire pre-measured dataset.

All computations have to be performed in real time. Longer computation times could result in lag between a patient's head movement and what can be seen through the HMD or a drop in the frame rate. This would significantly add to the risk of nausea for the patient. We implemented Uncertainty Sampling because it is faster to compute [134].

8.4 Our Application

Our application is made with the game engine Unity (version 2020.2.5f1). We build this for the HMD Pico Neo 2 Eye, which has six degrees of freedom, a 4k RGB display, a 101° field of view and built-in Tobii eye tracking hard- and software. We implemented a VSN assessment module based on active learning and a treatment module that provides the patient with a personalised experience. Stimuli are placed on spawn points picked by the model. A patient has to gaze at that stimulus for a certain amount of time in order to mark it as detected. During which, a yellow circle fills up around the stimulus, as shown in Figure 8.2. A video that showcases the application can be found on YouTube[1].

8.4.1 Assessment

In our application, we implemented three different scenes: a picnic table, a kitchen and a playground. This allows for the placement of stimuli in three different regions. These regions correspond to three clinically significant distances: near peripersonal (reaching) space, far peripersonal space and extrapersonal (far) space. The 3D effect of a patient's VSN can thus be studied more accurately. Screenshots are given in Figure 8.2.

A therapist has access to the following parameters to tailor the application to a patient's needs: game modus (assessment or treatment), scene (table, kitchen, playground), patient's number, number of stimuli to show, maximum allowed search time, minimum gaze distance (in degrees) and difficulty level. The latter controls the amount of distractors in the scene. These are static objects that resemble stimuli to be found. For example, a plate on the table. We conveniently grouped some of these parameters into editable presets.

The results of our assessment module are presented in heatmaps, as shown in Figure 8.1. In our case, these results are the predictions for search times predicted by the machine learning algorithm. We implemented a Gaussian process as a surrogate model. This allows us to perform active learning. Predictions of a Gaussian process for a search time at a specific point in a person's field of view are in fact Gaussian distributions. They consist of a mean or expected value and a

[1] `https://youtu.be/rflCHMf4ey4?si=OAlJy6W-jV19Aq_k`

variance, here reformulated as a two times sigma value. The left part of the heatmap shows the expected value for a point in the field of view of a person. A green colour on the heatmap indicates a small search time, red indicates a longer time and black means the stimulus was not found. The right part of the heatmap shows the two times sigma values that accompany the expected values. Both heatmaps provide information about the patient's field of view simultaneously: the prediction on the left and the (un)certainty of that prediction on the right.

The same colour scheme is kept, meaning that the green areas are regions where the visuospatial neglect is well mapped, while the red areas are regions where there is still some uncertainty. These red areas are candidates to put future stimuli in the active learning process. A red point in the expected value heatmap that corresponds to a green point in the two times sigma heatmap indicates that in this area we can be sure there is an attention deficit. On the other hand, a green point in the expected values heatmap that corresponds with a red point in the two times sigma heatmap, indicates we are not sure there is no neglect in this area.

One of the benefits of working with our statistical method is that we can exploit the uncertainty in the predictions. In the last step of our active learning strategy, explained in Section 8.3.2, we defined stopping criteria. One of those is the convergence of the posterior distribution, both for the mean and the variance (uncertainty) of the predicted values. If our model notices convergence, then there is little information to be gained from new measurements. This allows us to terminate the exercise for the patient prematurely. Alternatively, a predefined fixed number of stimuli to be found might result in burdening the patient with more stimuli than needed to come to the same conclusion about its neglect condition.

8.4.2 Treatment

Once a patient's VSN situation has been learned by the model, it can be exploited in therapy. Our application saves the trained Gaussian process models for each assessment session. During treatment, this model can be queried for spawn points in the vicinity of the border between the neglected areas and the non-neglected area. Placing stimuli in random places, which are either in the neglected area or not, results in exercises that are either too hard or too easy. Moreover, systematically placing stimuli on this border can train a patient to seek out this region in his field of view. The aim of this therapy, also known as cueing [59], is to gradually shift that border.

8.5 Results

To validate the methods used in our assessment, we conducted a case-control cross-sectional pilot study. It has been approved by the local Ethics Committee of the Antwerp University Hospital (Belgian registration number: B3002020000216). Data collection took place between October 2020 and September 2021 at the Rehabilitation Hospital RevArte in Antwerp, Belgium. All participants were asked to give signed informed consent.

We worked with three groups of participants: healthy controls, stroke participants without VSN and stroke participants with VSN. Due to the Covid-19 pandemic, only thirty-eight participants could be recruited. Participants were recruited from the stroke population of the rehabilitation hospital Revarte, Antwerp, Belgium, which is not an acute stroke unit. Participants between 16 and 99 years old with an ischemic or haemorrhagic stroke in the right hemisphere were eligible

for inclusion. Subjects were excluded if they fit one of the following criteria: unable to sit in a wheelchair or chair, unable to understand the procedure due to cognitive impairment, having impaired eyesight (i.e., visual field deficit), and refusal to participate. Cybersickness symptoms were assessed with the Simulator Sickness Questionnaire [80], as well as user experience.

Participants were seated in a wheelchair or on a straight back chair. Their trunk was restricted to the chair. Prior to the assessment, the screen of the head mounted display was projected onto a laptop and the eye tracking was calibrated to the participants' eye positions by the investigator. A tutorial of the far space version (the playground) was shown in which participants were instructed to look for the objects shown in the search task and hold their gaze until a yellow circle was completed. The investigator examined if participants had enough range of motion in the neck to view all objects. Participants were allowed to move their head and eyes in every direction without moving their trunk.

The levels for peripersonal (near) and for extrapersonal (far) neglect were completed in one session by all participants. This means the table and playground environment. All participants were tested again on the third difficulty level of both versions within one week to investigate the intra-rater reliability.

Cognitive function was tested with the mini mental state examination (MMSE). For testing neglect, the following three tests were administered: broken hearts test (BHT), line bisection test (LBT) and visual search time test (VSTT). These tests are commonly used in practice and serve as the standard to which we compare our VR application. All thirty-eight participants completed the VR programme. It was stated that a stroke patient had VSN if 2 out of 3 neglect test (BHT, LBT, VSTT) were positive. These results were compared with the findings of our application. This clinical statistical analysis is beyond the scope of this thesis. For more details, we refer to our publication [33]. The conclusion of that analysis, is that our application appears to be more sensitive compared to the standard test, with good intra-rater reliability.

The Simulator Sickness Questionnaire [80] showed that no participants experienced moderate to high levels of cybersickness symptoms after completing the VR application.

8.6 Discussion

Our VR application seems to be more sensitive than the pen-and-paper tests. By taking wider angles into account, the application can better imitate real life and reflect the impact of neglect on daily life more closely. According to the maximal search angles and the search area middle, there was only little difference between the initial testing and the re-testing, which indicates that our VR application is reliable. However, these results need to be interpreted with caution, as a larger study population is needed. Moreover, the treatment module was not included in the clinical investigation.

One limitation of our approach is the sensitivity to outliers. This is due to the fact that we strive to work with as few data points as possible. Typically, we work with ten to fifteen data points. If one of those is a false positive or negative, then the heatmaps are severely influenced in the wrong direction. This problem can be tackled by outlier detection. In our application, we added the additional check, that a stimulus that is successfully spotted can not be surrounded by stimuli that are not spotted in time. And vice versa, stimuli that are missed can not be surrounded by stimuli that are successfully spotted. In other words, the heatmaps should not show islands.

Moreover, a false positive or negative will also result in a large uncertainty around that measurement and a global increase of the total uncertainty. This is because the model now learns to deal with outcomes that vary more, which manifests itself as smaller length scales in the hyperparameters. This increase in uncertainty around the false positive or negative will influence how a next location for the placement of a new stimulus is chosen. The model will zoom in on these larger uncertainties to learn more. This effect, in combination with the outlier detection described above, makes the application less sensitive to outliers. However, care is still needed to be taken by clinicians when interpreting the results, as working with fewer measurements always means a higher dependency on an individual measurement.

The model constructs a personalised heatmap, which is a valuable tool for clinicians as it provides a clear view of a patient's personal VSN. This will make it easier to interpret the condition and explain it to not only the patient itself, but also friends, family and other caregivers. This can help to create awareness in the patient's surroundings.

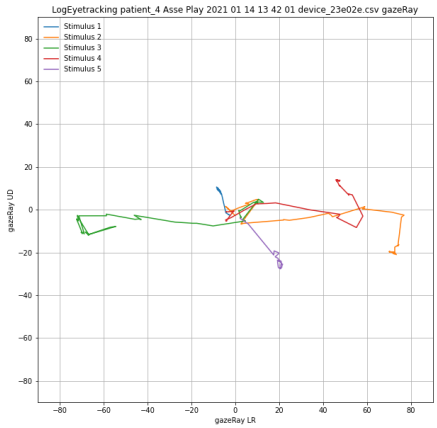
Our application also keeps detailed log files of the tracking of both eye and head movement. As shown in an example in Figure 8.3, the gaze of a patient in the 3D world can be decomposed in head and eye movement. We can see that for this individual searching for stimuli, the eyes remain scanning around the centre, while the head looks more left and right. This information can be used in research on search strategies, neck movement etc. This application can also be beneficial to assess and treat other attention-based conditions, search strategies and neck movements. This would be a fruitful area for further work. We plan to further expand the gamification aspect by implementing a game in the kitchen scene where the player has to follow a certain recipe.

8.7 Conclusion

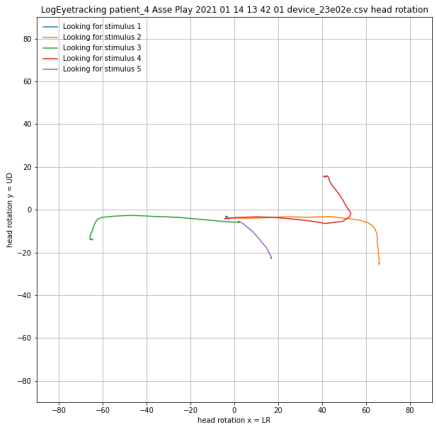
In this chapter, we described an artificial intelligence based virtual reality application that is able to assess and treat a patient's visuospatial neglect. Our application is human-centred, continually adjusting to the patient's input. During assessment, an active learning algorithm selects new measurement locations based on the patient's performance, incorporating new data after each stimulus to guide the choice of the next location. This iterative process fosters a dynamic interaction between the patient and the algorithm, much like a game of ping-pong.

By placing stimuli at certain spawn points in a patient's field of view and measuring the search times for those, we can effectively map his VSN situation using a surrogate model. In our work this is a Gaussian process. The term surrogate model is applied to Gaussian processes because they serve as proxies for the actual underlying reality. GPs approximate this by offering predictions. This makes GPs valuable in situations where we need to estimate or model complex relationships, especially in the absence of a clear, known model for the underlying data. Moreover, this surrogate model allows for active learning, which effectively lowers the number of stimuli to be placed. We perform measurements in the field of view where the uncertainty is largest. This results in a more feasible (shorter) experience for patients while still maintaining high accuracy.

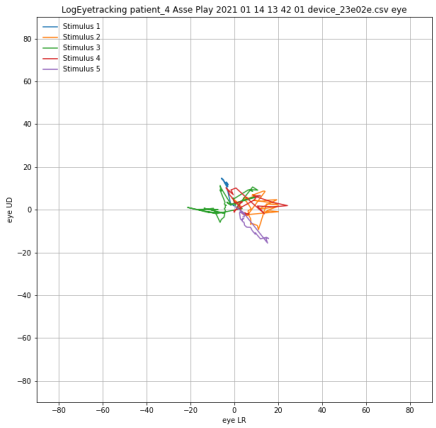
We developed our application using the Unity 3D game engine. It works on a Pico Neo 2 Eye, which has built-in eye tracking. As an added bonus, this provides therapists and clinical researchers with detailed data on eye and head movements that can be used in research on search strategies etc. We compared our work to standard tests used in practice today: broken hearts test, line bisection test and visual search time test. Our VR application proves to be more sensitive,



(a)



(b)



(c)

Figure 8.3: Movement of a person's gaze in the 3D world (a), decomposed in his head movement (b) and his eye movement (c).

while intra-rater reliability remains high.

Moreover, the trained models reveal the border between neglected areas and non-neglected areas in a patient's field of view. This can be exploited in therapy, where stimuli can be placed around this border in order to gradually move it. Our implementation tailors both the assessment and the treatment of visuospatial neglect to the patient itself. This is currently not possible using the existing practical testing methods on pen-and-paper.

Finally, this treatment could be performed without the intervention of a therapist, opening the way for tele-rehabilitation. Although in practice, for most patients, handling the headset and starting the application still requires assistance from a caregiver. More usability research is needed to

make this as practicable as possible.

The method in this work is protected under patent *Computer Implemented Method And System For Mapping Spatial Attention*.
Inventors: Wim Saeys, Steven Truijen and Ivan De Boi
Publication Number WO/2022/122834
Publication Date 16.06.2022
International Application No. PCT/EP2021/084817
International Filing Date 08.12.2021
Priority Data 20212871.6 09.12.2020 EP

Chapter 9

General Conclusions

Novel calibration procedures for devices that perform 3D measurements based on straight lines have been provided in this thesis. In each chapter, we have provided conclusions. In this section we will summarise the most important aspects and look at overall limitations and recommendations.

9.1 Conclusions

Earlier work showed the benefit of working with data-driven models to calibrate galvanometric setups. Our semi-data-driven approach, presented in Chapter 3 and 4, takes this reasoning one step further. It shows that we can achieve higher accuracy in predicting straight lines when incorporating one simple assumption in data-driven models, namely that lasers are straight lines in our 3D world. Moreover, we expanded the literature on constrained Gaussian processes by applying several techniques to enforce quadratic constraints to kernels in the context of Plücker coordinates of straight lines. It is of interest to the calibration of rotating Lidars, laser Doppler vibrometers and 3D scanners.

Via principle component analysis, we can compose a lower dimensional latent space representation of line elements of a point cloud. This was already well studied in [120, 119, 123, 89]. However, PCA can be seen as a linear map. Alternatively, the Gaussian process latent variable model allows for a non-linear mapping from points in a lower dimensional latent space to higher dimensional observations [86]. Building on this in Chapter 5, we demonstrated the benefits of this approach in surface approximation, surface segmentation and surface denoising. Before this study, only primitive shapes, such as the ones shown in Figure 5.1, could be handled. Our method allows for a much wider range of possible shapes. This new understanding should help to improve the quality of 3D laser scanners.

Exact checkerboard detection is of utmost importance in computer vision to calibrate a camera. Most approaches are based on Zhang's Method [168]. Our work in Chapter 6 allows for the enhancement of the coordinates of detected corners and even for the recovery of missing corners. This will be of great benefit to people relying on low quality images of checkerboards. We also presented a novel camera calibration procedure in Chapter 7, which maps every pixel in the image to a straight line. This approach is able to transform camera (systems) into virtual pinhole camera models. This study lays the groundwork for future research on the usage of cameras with

heavy distortion (such as fisheye lenses or catadioptric systems) into SLAM, shape-from-motion, photogrammetry, and many more fields that often rely on pinhole camera setups.

Reliable mapping of a patient's visuospatial neglect situation is studied in Chapter 8. We employ active learning techniques within a virtual reality setting to reduce the number of measurements, thereby easing the burden on patients affected by this disorder. Our approach shows higher sensitivity compared to pen-and-paper tests, which are still common practice today. Moreover, our method is able to measure visuospatial neglect in a 3D environment, instead of on a 2D flat piece of paper. Today, our software enables ongoing research on the treatment of visuospatial neglect. Moreover, it provides new data for analysing how patients use their eyes and head when searching for an object in their 3D world.

Gaussian processes can play an important role in the calibration of various 3D measurement devices. By changing the calibration process from a parameter finding optimisation problem to a data-driven regression challenge, we can overcome the errors introduced by the sometimes overly simplified physical and mathematical models that explain the inner workings of the measuring device under investigation. The main benefit of working with probabilistic methods, and more specifically Gaussian processes, is the fact that they handle low data regimes very well.

Other important benefits are the built-in mechanism against overfitting and the uncertainty estimate for the predictions. This can be further exploited by techniques such as Bayesian optimisation and active learning. All these benefits make Gaussian processes a prime choice when it comes to calibrating 3D measuring devices.

9.2 Recommendations, Limitations and Future Work

Neural networks are used throughout a myriad of machine learning models and are also capable of learning complex non-linear relationships. The works [97, 101] implement these in their purely data-driven calibration methods. In our context of working with straight lines, we recommend for this approach to enforce the Grassmann-Plücker relationship in Equation 2.13 via so called *physics-informed neural networks*. This will ensure the orthogonality of the direction and moment vector of the predicted straight lines.

New understandings in Chapter 3 and 4 allowed for the calibration on a set of straight lines in 3D themselves. In contrast, the work [162] predicts points on a flat validation plane, making this a 2D calibration challenge. Our models predict straight lines in 3D space, which effectively renders our approach a full 3D calibration.

As mentioned in Chapter 4, we can construct more elaborate kernels so that the predicted six-tuples obey the Grassmann-Plücker relationship in Equation 2.13. The biggest downside to these approaches was the increase in training time. The culprit was the construction of the kernel, which required the dataset to be reformulated into a much larger matrix to invert. With n the number of data points in a dataset, the standard complexity of the Gaussian process is $\mathcal{O}(n^3)$ for computation and $\mathcal{O}(n^2)$ for storage [161]. There is a gain in the quality of the predictions, but at a significant time penalty. We did not explore the many solutions available in the literature [161]. It would be interesting future work to see if the gain in accuracy of working with the more complicated kernels that predict zero pitch six-tuples can be upheld when also relying on approximations for speeding up the training times. Another strategy is to accept the non-zero pitches of the GP

predictions. Further research is needed to fully understand the implications working with screws instead of straight lines in calibration procedures.

Gaussian process latent variable models provide a non-linear alternative to the well known PCA methods to discover latent lower dimensional subspaces in point cloud line element data described in Chapter 5. This method could also be used to handle missing data, e.g. filling in holes in the point cloud. The main consideration would once again be the assurance of following the Grassmann-Plücker relation of Equation 2.13. Deviating too far from this will result in a screw that is not representative of the normal line to the surface. Moreover, this faulty line representation will also have a non negligible effect on the location of the point on that line, and thus a bad prediction for the missing point in the point cloud. The limits of this approach remain an open question.

In Chapter 6, we provide a novel detection method for checkerboards in camera calibration. However, Gaussian processes can be seen as universal smoothing machines and as such can sometimes yield functions that are overly smooth. This happens when the number of corners is low relative to the underlying distortion of the checkerboard. For instance, when working with fisheye lenses and a 6x8 checkerboard, as is the case in Figure 6.7. The Gaussian process model is not always able to capture the curvature of the board in regions where there are few corners. There simply is not enough data available. The result might be an over-smoothing of the predicted locations of those corners in the image. To overcome this issue, one could resort to working with larger checkerboards with enough corners or with a technique such as Gray code that provides per-pixel information and not just information about the pixels of detected corners [141].

Various works describe how we can combine views of multiple cameras into one stitched panorama or utilise them in a 3D depth estimation of the surrounding world. An excellent overview is given in the book [66]. However, most of these methods rely on geometry, which only holds when working with pinhole cameras. We showed in Chapter 7 how Gaussian processes can learn the relationship between image pixels and 3D real world straight lines, effectively calibrating the camera to a perfect pinhole camera. A promising area of research is to explore the usage of this approach in camera systems where multiple cameras are implemented, with or without overlapping fields of view.

Examples of how our camera calibration allows for the rectification of images are given in Figure 7.2. However, there are two main drawbacks. First, there is no guarantee that all pixels in the resulting image get filled in by the Gaussian process predictions. Some pixels get no value and remain black. These can be filled in with various infilling techniques, also sometimes called inpainting, such as diffusion models [25], Neural Diffusion Processes [42], GPLVMs [84] like the ones used in Chapter 5 or Variational Autoencoders [160]. At the moment of writing, this is a very active field in computer vision.

Yielding two predictions for each pixel (one for an x-value and one for a y-value), and working with a lot of pixels, the Gaussian process model has a significant drawback regarding the time it takes to construct the rectified image. Once the two GPs have been trained, the predictions for n pixels costs $\mathcal{O}(n)$ operations for the mean and $\mathcal{O}(n^2)$ operations for the variance [161]. In upcoming work, we will address this by working with multi-resolution Gaussian processes [65]. We also investigated the feasibility of Kd-trees in Gaussian process regression to partition high resolution input spaces [35]. We deemed this paper outside the scope of this work.

One benefit of working with probabilistic models is that they allow for an exploitation of the uncertainty in the predictions. Chapter 8 presented a measuring technique to assess visuospatial

neglect, a condition where patients ignore a part of their 3D surroundings. The main difference our approach introduces to the state-of-the-art, is *active learning*. We pick a new location in a patient's field of view to perform a measurement based on the Gaussian process posterior uncertainty. This is only possible when working with probabilistic machine learning methods. This distinguishes Gaussian processes from for instance neural network solutions.

Utilising the uncertainty in optimisation is the focus of another technique called *Bayesian optimisation*. The goal is to find an optimum in as few evaluations as possible in a typically expensive to evaluate objective function. The predicted uncertainty is used to trade-off exploitation and exploration when querying the input space. In our work on dynamic line scan thermography parameter design [157], we implemented a Gaussian process emulator in this way to find an optimal set of parameters (e.g., camera position, speed of the moving object, power consumption, etc.). We did not include this paper in this thesis to keep the focus on techniques related to calibration.

Underlying structure in the data can also be discovered via the work of [14]. It describes a method to determine a variety based on samples or measurements. A variety is defined as the set of solutions of a system of polynomial equations over real or complex numbers. Informally speaking, they describe a shape. In the context of this work, the measurements are straight lines. These are not just six-tuples without underlying structure. For example, a pinhole camera is determined by straight lines going through one central point of intersection. All of these lines, as points in $\mathbb{P}^5$, lie on a flat 2-plane, entirely contained in the Klein quadric. Another example are the lines obtained by a rotating mirror that reflects a fixed laser line. These lines form a ruled surface known as a hyperboloid. Their point representations in $\mathbb{P}^5$ show a conic on the Klein quadric. By reflecting a fixed laser with two rotating mirrors, we get a two-mirror-congruence [115]. All these line representations on the Klein quadric can be described by a variety. Calibrating a 3D measuring device can be reformulated as finding the correct variety and the correspondence between the inputs and the points on the variety. Moreover, measurements can be corrected to be solutions of the variety equations, which effectively is the same as removing measurement noise. In [14], several methods building on persistent homology are proposed to discover these underlying varieties. It would be a fruitful exercise to try these methods on the 3D measuring devices mentioned in this thesis, more specifically galvanometric laser scanners and cameras.

Probabilistic machine learning methods for calibrating 3D measuring devices based on straight lines have been a very interesting topic to study. This work has sparked several directions for our own scientific curiosity. We sincerely hope that this has been helpful and perhaps even inspiring for future researchers as well.

Bibliography

[1] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. *Electronics*, 9(8), 2020. 61

[2] Andrea Albarelli, Emanuele Rodolà, and Andrea Torsello. Robust Camera Calibration using Inaccurate Targets. In *Proceedings of the British Machine Vision Conference*, page 16.1–16.10. BMVA Press, 2010. 67

[3] A Alizadeh Naeini, A Ahmad, M M Sheikholeslami, P Claudio, and G Sohn. An Unsupervised Registration of 3D Point Clouds to 2D CAD Model: a Case Study of Floor Plan. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020:9–13, 2020. 45

[4] Abd Albasset Almamou, Thomas Gebhardt, Sebastian Bock, Jörg Hildebrand, and Willfried Schwarz. Quality control of constructed models using 3d point cloud, 2015. 45

[5] Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for Vector-Valued Functions: a Review, 2012. 22, 35

[6] E Barkan and J Swartz. System Design Considerations In Bar-Code Laser Scanning. *Optical Engineering*, 23:413–420, 1984. 17

[7] A M Barrett and K E Houston. Update on the Clinical Approach to Spatial Neglect. *Current Neurology and Neuroscience Reports*, 19(5), 4 2019. 96, 98

[8] A M Barrett and Tufail Muzaffar. Spatial cognitive rehabilitation and motor recovery after stroke. *Current Opinion in Neurology*, 27(6):653–658, 12 2014. 98

[9] Anna M Barrett, Laurel J Buxbaum, H Branch Coslett, Emmeline Edwards, Kenneth M Heilman, Argye E Hillis, William P Milberg, and Ian H Robertson. Cognitive Rehabilitation Interventions for Neglect and Related Disorders: Moving from Bench to Bedside in Stroke Patients. *Journal of Cognitive Neuroscience*, 18(7):1223–1236, 7 2006. 96

[10] Paul Beardsley, David Murray, and Andrew Zisserman. Camera calibration using multiple images. In *Computer Vision—ECCV'92: Second European Conference on Computer Vision Santa Margherita Ligure, Italy, May 19–22, 1992 Proceedings 2*, pages 312–320, 1992. 81

[11] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Jürgen Gall, and Cyrill Stachniss. Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset. *The International Journal of Robotics Research*, 40(8-9):959–967, 2021. 45

[12] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Joshua A Levine, Andrei Sharf, and Claudio T Silva. State of the Art in Surface Reconstruction from Point Clouds. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014. 47

[13] Martine S Bosma, Tanja C W Nijboer, Monique A A Caljouw, and Wilco P Achterberg. Impact of visuospatial neglect post-stroke on daily activities, participation and informal caregiver burden: A systematic review. *Annals of Physical and Rehabilitation Medicine*, 63(4):344–358, 7 2020. 96, 98

[14] Paul Breiding, Sara Kališnik, Bernd Sturmfels, and Madeleine Weinstein. Learning algebraic varieties from samples. *Revista Matemática Complutense*, 31(3):545–593, 2018. 112

[15] Mona Buisson-Fenet, Friedrich Solowjow, and Sebastian Trimpe. Actively learning gaussian process dynamics. In *Learn. Dyn. Control*, pages 5–15. PMLR, 2020. 103

[16] Wilhelm Burger. Zhang's camera calibration algorithm: in-depth tutorial and implementation. *Hgb16-05*, pages 1–6, 2016. 84

[17] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. Albumentations: Fast and Flexible Image Augmentations. *Information*, 11(2), 2020. 72

[18] Bruno Caprile and Vincent Torre. Using vanishing points for camera calibration. *International journal of computer vision*, 4(2):127–139, 1990. 81

[19] Paolo Castellini, Gian Marco Revel, and Enrico Primo Tomasini. Laser doppler vibrometry. In *An Introduction to Optoelectronic Sensors*, pages 216–229. World Scientific, 2009. 17

[20] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020. 61

[21] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, and others. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 47

[22] Ben Chen, Yuyao Liu, and Caihua Xiong. Automatic checkerboard detection for robust camera calibration. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2021. 67, 68

[23] Ben Chen, Caihua Xiong, and Qi Zhang. CCDN: Checkerboard Corner Detection Network for Robust Camera Calibration. In *Intelligent Robotics and Applications*, pages 324–334, Cham, 2018. Springer International Publishing. 68, 72

[24] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. *IEEE Signal Processing Magazine*, 38(1):68–86, 2020. 45

[25] Ciprian Corneanu, Raghudeep Gadde, and Aleix M Martinez. LatentPaint: Image Inpainting in Latent Space With Diffusion Models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4334–4343, 1 2024. 111

[26] Henry Crapo and Walter Whiteley. Statics of frameworks and motions of panel structures: a projective geometric introduction. *Structural Topology, 1982, núm. 6*, 1982. 13

[27] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 47

[28] Andreas C Damianou, Carl Henrik Ek, Michalis K Titsias, and Neil D Lawrence. Manifold Relevance Determination. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 531–538, Madison, WI, USA, 2012. Omnipress. 53, 63

[29] Andreas C. Damianou and Neil D. Lawrence. Deep Gaussian processes. In *Journal of Machine Learning Research*, volume 31, 2013. 11

[30] Andreas C Damianou, Neil D Lawrence, and Carl Henrik Ek. Multi-view Learning as a Nonparametric Nonlinear Inter-Battery Factor Analysis. *J. Mach. Learn. Res.*, 22:86:1–86:51, 2021. 53, 62, 63

[31] Deirdre R Dawson and Thomas D Marcotte. Special issue on ecological validity and cognitive assessment. *Neuropsychological Rehabilitation*, 27(5):599–602, 5 2017. 96, 98

[32] Ivan De Boi, Carl Henrik Ek, and Rudi Penne. Surface Approximation by Means of Gaussian Process Latent Variable Models and Line Element Geometry. *Mathematics*, 11(2), 2023. 45

[33] Ivan De Boi, Elissa Embrechts, Quirine Schatteman, Rudi Penne, Steven Truijen, and Wim Saeys. Assessment and treatment of visuospatial neglect using active learning with Gaussian processes regression. *Artificial Intelligence in Medicine*, 149:102770, 2024. 95, 105

[34] Ivan De Boi, Stuti Pathak, Marina Oliveira, and Rudi Penne. How to Turn Your Camera into a Perfect Pinhole Model. In Verónica Vasconcelos, Inêsand Domingues, and Simão Paredes, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 90–107, Cham, 2024. Springer Nature Switzerland. 81

[35] Ivan De Boi, Bart Ribbens, Pieter Jorissen, and Rudi Penne. Feasibility of kd-trees in gaussian process regression to partition test points in high resolution input space. *Algorithms*, 13(12), 2020. 111

[36] Ivan De Boi, Seppe Sels, Olivier De Moor, Steve Vanlanduit, and Rudi Penne. Input and Output Manifold Constrained Gaussian Process Regression for Galvanometric Setup Calibration. *IEEE Transactions on Instrumentation and Measurement*, 71:1–8, 4 2022. 31

[37] Ivan De Boi, Seppe Sels, and Rudi Penne. Semidata-Driven Calibration of Galvanometric Setups Using Gaussian Processes. *IEEE Transactions on Instrumentation and Measurement*, 71:1–8, 11 2021. 1, 17, 32, 40

[38] Stefan Van der Stigchel and Tanja C W Nijboer. The imbalance of oculomotor capture in unilateral visual neglect. *Consciousness and Cognition*, 19(1):186–197, 3 2010. 96

[39] Frederic Devernay and Olivier Faugeras. Straight lines have to be straight. *Machine vision and applications*, 13:14–24, 2001. 81

[40] Neila J Donovan, Diane L Kendall, Shelley C Heaton, Sooyeon Kwon, Craig A Velozo, and Pamela W Duncan. Conceptualizing Functional Cognition in Stroke. *Neurorehabilitation and Neural Repair*, 22(2):122–135, 9 2007. 96, 98

[41] Alexander Duda and Udo Frese. Accurate Detection and Localization of Checkerboard Corners for Calibration. In *British Machine Vision Conference*, 2018. 68, 73

[42] Vincent Dutordoir, Alan Saul, Zoubin Ghahramani, and Fergus Simpson. Neural Diffusion Processes. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8990–9012. PMLR, 10 2023. 111

[43] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, Cambridge, 11 2014. 8, 9

[44] Assaf Y Dvorkin, Ross A Bogey, Richard L Harvey, and James L Patton. Mapping the neglected space: gradients of detection revealed by virtual reality. *Neurorehabilitation and neural repair*, 26(2):120–131, 2012. 99

[45] Hossam El-Din Fawzy. 3D laser scanning and close-range photogrammetry for buildings documentation: A hybrid technique towards a better accuracy. *Alexandria Engineering Journal*, 58(4):1191–1204, 2019. 17

[46] Elissa Embrechts, Tamaya Van Criekinge, Jonas Schröder, Tanja Nijboer, Christophe Lafosse, Steven Truijen, and Wim Saeys. The association between visuospatial neglect and balance and mobility post-stroke onset: A systematic review. *Annals of Physical and Rehabilitation Medicine*, 64(4):101449, 7 2021. 96

[47] Ikenna Enebuse, Mathias Foo, Babul Salam Ksm Kader Ibrahim, Hafiz Ahmed, Fhon Supmak, and Odongo Steven Eyobu. A Comparative Review of Hand-Eye Calibration Techniques for Vision Guided Robots. *IEEE Access*, 9:113143–113155, 2021. 68

[48] Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and modeling for computer experiments*. Chapman and Hall/CRC, 2005. 100

[49] Julia Fellrath, Vanessa Blanche-Durbec, Armin Schnider, Anne-Sophie Jacquemoud, and Radek Ptak. Visual search in spatial neglect studied with a preview paradigm. *Frontiers in Human Neuroscience*, 6, 2012. 96

[50] Thomas Fersch, Robert Weigel, and Alexander Koelpin. Challenges in miniaturized automotive long-range lidar system design. In Bahram Javidi, Jung-Young Son, and Osamu Matoba, editors, *Three-Dimensional Imaging, Visualization, and Display 2017*, volume 10219, page 160 – 171. SPIE, 2017. 17

[51] Martin A Fischler and Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, 6 1981. 20, 32

[52] Wolfgang Förstner and Bernhard P Wrobel. *Photogrammetric Computer Vision*, volume 64. Springer Cham, 1 edition, 2016. 92

[53] Barbara Frank, Cyrill Stachniss, Nichola Abdo, and Wolfram Burgard. Using Gaussian Process Regression for Efficient Motion Planning in Environments with Deformable Objects. In *Automated Action Planning for Autonomous Mobile Robots*, 2011. 30

[54] Peter Fuersattel, Sergiu Dotenco, Simon Placht, Michael Balda, Andreas Maier, and Christian Riess. OCPAD - Occluded checkerboard pattern detector. In *2016 IEEE Winter Conf. Appl. Comput. Vis.* IEEE, 2016. 68

[55] M Galan, M Strojnik, and Y Wang. Design method for compact, achromatic, high-performance, solid catadioptric system (SoCatS), from visible to IR. *Optics express*, 27(1):142–149, 2019. 81

[56] Andreas Geiger, P Lenz, Christoph Stiller, and Raquel Urtasun. The KITTI vision benchmark suite. *URL http://www. cvlibs. net/datasets/kitti*, 2, 2015. 47

[57] Andreas Geiger, Frank Moosmann, Omer Car, and Bernhard Schuster. Automatic camera and range sensor calibration using a single shot. In *2012 IEEE International Conference on Robotics and Automation*, pages 3936–3943. IEEE, 5 2012. 68

[58] Paul J Gemperline, James R Long, and Vasilis G Gregoriou. Nonlinear multivariate calibration using principal components regression and artificial neural networks. *Analytical Chemistry*, 63(20):2313–2323, 1991. 1, 18

[59] Lisa Kunkel genannt Bode, Andreas Sprenger, Christoph Helmchen, Björn Hauptmann, Thomas F Münte, and Björn Machner. Combined optokinetic stimulation and cueing-assisted reading therapy to treat hemispatial neglect: A randomized controlled crossover trial. *Annals of Physical and Rehabilitation Medicine*, 66(5):101713, 2023. 104

[60] Marc G Genton. Classes of Kernels for Machine Learning: A Statistics Perspective. *J. Mach. Learn. Res.*, 2:299–312, 3 2002. 9

[61] Alexandra Gessner, Javier Gonzalez, and Maren Mahsereci. Active multi-information source Bayesian quadrature. In *Uncertain. Artif. Intell.*, pages 712–721. PMLR, 2020. 103

[62] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *International Conference on Machine Learning*, pages 3809–3820, 2021. 47

[63] Yongjie Gui, Yanyan Wu, Yajie Wang, and Chunpeng Yao. Visual Image Processing of Humanoid Go Game Robot Based on OPENCV. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 3713–3716, 2020. 68

[64] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, 2021. 45, 47

[65] Oliver Hamelijnck, Theodoros Damoulas, Kangrui Wang, and Mark A. Girolami. Multi-resolution multi-task Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 111

[66] R. I. Hartley and A Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 82, 84, 111

[67] Kenneth M Heilman and Edward Valenstein. Mechanisms underlying hemispatial neglect. *Annals of Neurology*, 5(2):166–170, 2 1979. 96

[68] Michaël Hillen, Ivan De Boi, Thomas De Kerf, Seppe Sels, Edgar Cardenas De La Hoz, Jona Gladines, Gunther Steenackers, Rudi Penne, and Steve Vanlanduit. Enhanced Checkerboard Detection Using Gaussian Processes. *Mathematics*, 11(22), 2023. 67

[69] M Hofer, B Odehnal, H Pottmann, T Steiner, and J Wallner. 3D shape recognition and reconstruction based on line element geometry. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1532–1538, 2005. 46, 48, 49, 50, 59

[70] Andreas Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016. 96

[71] Hanne Huygelier, Céline R Gillebert, Raymond van Ee, and Vero Vanden Abeele. The Design of a Virtual Reality Game for Stroke-Induced Attention Deficits. In *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play*. ACM, 10 2017. 99

[72] Hanne Huygelier, Brenda Schraepen, Christophe Lafosse, Nathalie Vaes, Fabienne Schillebeeckx, Karla Michiels, Eline Note, Vero Vanden Abeele, Raymond van Ee, and Céline R Gillebert. An immersive virtual reality game to train spatial attention orientation after stroke: A feasibility study. *Applied Neuropsychology: Adult*, 29(5):915–935, 9 2020. 99

[73] Akitoshi Ito, Jinghui Li, and Yusuke Maeda. SLAM-Integrated Kinematic Calibration Using Checkerboard Patterns. In *2020 IEEE/SICE International Symposium on System Integration (SII)*, pages 551–556, 2020. 68

[74] Woncheol Jang, Joon-Ho Shin, Mingyu Kim, and Kwanguk (Kenny) Kim. Human field of regard, field of view, and attention bias. *Computer Methods and Programs in Biomedicine*, 135:115–123, 10 2016. 99

[75] Carl Jidling, Niklas Wahlström, Adrian Wills, and Thomas B Schön. Linearly constrained Gaussian processes. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 31, 35, 36

[76] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998. 100

[77] Rigoberto Juarez-Salazar and Victor H Diaz-Ramirez. Flexible camera-projector calibration using superposed color checkerboards. *Optics and Lasers in Engineering*, 120:59–65, 7 2019. 67

[78] Adrian Kaehler. *Learning OpenCV 3 : computer vision in C++ with the OpenCV library*. O'Reilly Media, 2016. 68

[79] Jiwoo Kang, Hyunse Yoon, Seongmin Lee, and Sanghoon Lee. Sparse Checkerboard Corner Detection from Global Perspective. In *2021 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 12–17, 2021. 67, 68

[80] Robert S Kennedy, Norman E Lane, Kevin S Berbaum, and Michael G Lilienthal. Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *The International Journal of Aviation Psychology*, 3(3):203–220, 7 1993. 105

[81] Asif Khan, Jian-Ping Li, Asad Malik, and M Yusuf Khan. Vision-based inceptive integration for robotic control. In *Soft Computing and Signal Processing: Proceedings of ICSCSP 2018, Volume 2*, pages 95–105, 2019. 81

[82] Inki Kim, Renato Juliano Martins, Jaehyuck Jang, Trevon Badloe, Samira Khadir, Ho-Youl Jung, Hyeongdo Kim, Jongun Kim, Patrice Genevet, and Junsuk Rho. Nanophotonics for light detection and ranging technology. *Nature Nanotechnology*, pages 1–17, 2021. 18

[83] J Lagarias, J Reeds, M Wright, and P E Wright. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM J. Optim.*, 9:112–147, 1998. 28

[84] Vidhi Lalchand, Aditya Ravuri, and Neil D Lawrence. Generalised GPLVM with Stochastic Variational Inference. In Gustau Camps-Valls, Francisco J R Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 7841–7864. PMLR, 10 2022. 53, 56, 60, 62, 63, 111

[85] Neil Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, 6(60):1783–1816, 2005. 53

[86] Neil D Lawrence. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 329–336, Cambridge, MA, USA, 2003. MIT Press. 46, 53, 109

[87] Neil D Lawrence and Joaquin Quiñonero-Candela. Local Distance Preservation in the GP-LVM through Back Constraints. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 513–520, New York, NY, USA, 2006. Association for Computing Machinery. 53, 60

[88] Miguel Lázaro-Gredilla. Bayesian Warped Gaussian Processes. In F Pereira, C J Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 79

[89] In-Kwon Lee, Johannes Wallner, and Helmut Pottmann. Scattered data approximation with kinematic surfaces. *SAMPTA*, 99:72–77, 1999. 46, 49, 50, 59, 109

[90] Lesueur Vincent and Vincent Nozick. Least Square for Grassmann-Cayley Agelbra in Homogeneous Coordinates. In Akihiro Huang FaySugimoto, editor, *Image and Video Technology – PSIVT 2013 Workshops*, pages 133–144, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. 20, 27, 32, 89

[91] Ping Li and Songcan Chen. A review on Gaussian Process Latent Variable Models. *CAAI Transactions on Intelligence Technology*, 1(4), 2016. 53

[92] ZHANG Li, L I U Yuxuan, S U N Yangjie, L A N Chaozhen, A I Haibin, and F A N Zhongli. A review of developments in the theory and technology of three-dimensional reconstruction in digital aerial photogrammetry. *Acta Geodaetica et Cartographica Sinica*, 51(7):1437, 2022. 81

[93] Kang Liao, Lang Nie, Shujuan Huang, Chunyu Lin, Jing Zhang, Yao Zhao, Moncef Gabbouj, and Dacheng Tao. Deep Learning for Camera Calibration and Beyond: A Survey. *arXiv preprint arXiv:2303.10559*, 2023. 82

[94] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. 8

[95] Haitao Liu, Jianfei Cai, and Yew-Soon Ong. Remarks on multi-output Gaussian process regression. *Knowledge-Based Systems*, 144:102–121, 2018. 22, 35

[96] Yaroslava Lochman, Kostiantyn Liepieshov, Jianhui Chen, Michal Perdoch, Christopher Zach, and James Pritts. BabelCalib: A Universal Approach to Calibrating Central Cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15253–15262, 2021. 67

[97] James R Long, Vasilis G Gregoriou, and Paul J Gemperline. Spectroscopic calibration and quantitation using artificial neural networks. *Analytical Chemistry*, 62(17):1791–1797, 1990. 1, 18, 110

[98] Yifan Lu, Jiayi Ma, Leyuan Fang, Xin Tian, and Junjun Jiang. Robust and Scalable Gaussian Process Regression and Its Applications. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2023-June, 2023. 11

[99] David J C MacKay. Introduction to Gaussian processes. *NATO ASI series F computer and systems sciences*, 168:133–166, 1998. 34

[100] A Manakov, H Seidel, and Ivo Ihrke. A Mathematical Model and Calibration Procedure for Galvanometric Laser Scanning Systems. In *Vision, Modeling, and Visualization*, pages 207–2014. The Eurographics Association, 2011. 1, 18, 32, 34

[101] Xianyu Meng, Guohua Cao, Xue Li, and Qiongying Lv. 2-D Scanning Galvanometer Error Analysis and Its Correction. *Journal of Physics: Conference Series*, 1345:22068, 11 2019. 18, 110

[102] Alican Mertan, Damien Jade Duff, and Gozde Unal. Single image depth estimation: An overview. *Digital Signal Processing*, page 103441, 2022. 81

[103] Niloy J Mitra and An Nguyen. Estimating Surface Normals in Noisy Point Cloud Data. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, pages 322–328, New York, NY, USA, 2003. Association for Computing Machinery. 50

[104] Tormod Næs, Knut Kvaal, Tomas Isaksson, and Charles Miller. Artificial Neural Networks in Multivariate Calibration. *Journal of Near Infrared Spectroscopy*, 1(1):1–11, 1993. 1, 18

[105] Tanja Nijboer, Ingrid van de Port, Vera Schepers, Marcel Post, and Anne Visser-Meily. Predicting Functional Outcome after Stroke: The Influence of Neglect on Basic Activities in Daily Living. *Frontiers in Human Neuroscience*, 7, 2013. 96, 98

[106] Tanja C W Nijboer, Boudewijn J Kollen, and Gert Kwakkel. Time course of visuospatial neglect early after stroke: A longitudinal cohort study. *Cortex*, 49(8):2021–2027, 9 2013. 96, 98

[107] Tanja C W Nijboer, Boudewijn J Kollen, and Gert Kwakkel. The Impact of Recovery of Visuo-Spatial Neglect on Motor Recovery of the Upper Paretic Limb after Stroke. *PLoS ONE*, 9(6):e100584, 6 2014. 98

[108] Umberto Noè, Alan Lazarus, Hao Gao, Vinny Davies, Benn MacDonald, Kenneth Mangion, Colin Berry, Xiaoyu Luo, and Dirk Husmeier. Gaussian process emulation to accelerate parameter estimation in a mechanical model of the left ventricle: a critical step towards clinical end-user relevance. *J. R. Soc. Interface*, 16(156), 2019. 100

[109] B Odehnal, H Pottmann, and J Wallner. Equiform kinematics and the geometry of line elements. *Beitr. Algebra Geom.*, 47(2):567–582, 2006. 49, 50, 59

[110] Sergey Oladyshkin, Farid Mohammadi, Ilja Kroeker, and Wolfgang Nowak. Bayesian3 Active Learning for the Gaussian Process Emulator Using Information Theory. *Entropy 2020, Vol. 22, Page 890*, 22(8):890, 8 2020. 103

[111] Kutsev Bengisu Ozyoruk, Guliz Irem Gokceler, Taylor L Bobrow, Gulfize Coskun, Kagan Incetan, Yasin Almalioglu, Faisal Mahmood, Eva Curto, Luis Perdigoto, Marina Oliveira, Hasan Sahin, Helder Araujo, Henrique Alexandrino, Nicholas J Durr, Hunter B Gilbert, and Mehmet Turan. EndoSLAM dataset and an unsupervised monocular visual odometry and depth estimation approach for endoscopic videos. *Medical Image Analysis*, 71:102058, 2021. 77

[112] Hee-Kyung Park, Jin-Woo Chung, and Hong-Seop Kho. Use of hand-held laser scanning in the assessment of craniometry. *Forensic Science International*, 160(2):200–206, 2006. 17

[113] E Pasolli and F Melgani. Gaussian process regression within an active learning scheme. In *2011 IEEE International Geoscience and Remote Sensing Symposium*, pages 3574–3577, 2011. 96, 103

[114] Rudi Penne. A mechanical interpretation of least squares fitting in 3D. *Bulletin of the Belgian Mathematical Society-Simon Stevin*, 15(1):127–134, 2008. 89

[115] Rudi Penne, Ivan De Boi, and Steve Vanlanduit. On the Angular Control of Rotating Lasers by Means of Line Calculus on Hyperboloids. *Sensors*, 23(13), 2023. 41, 112

[116] Rudi Penne, Bart Ribbens, and Steven Puttemans. A New Method for Computing the Principal Point of an Optical Sensor by Means of Sphere Images. In *Computer Vision ACCV 2018*, pages 676–690. Springer International Publishing, 2019. 82

[117] Rudi Penne, Bart Ribbens, and Pedro Roios. An Exact Robust Method to Localize a Known Sphere by Means of One Image. *International Journal of Computer Vision*, 127(8):1012–1024, 12 2018. 82

[118] Simon Placht, Peter Fürsattel, Etienne Assoumou Mengue, Hannes Hofmann, Christian Schaller, Michael Balda, and Elli Angelopoulou. ROCHADE: Robust Checkerboard Advanced Detection for Camera Calibration. In *Computer Vision – ECCV 2014*, pages 766–779, Cham, 2014. Springer International Publishing. 68

[119] H Pottmann, I K Lee, and T Randrup. Reconstruction of kinematic surfaces from scattered data. In *Proceedings of symposium on geodesy for geotechnical and structural engineering*, pages 483–488, 1998. 45, 46, 49, 50, 59, 109

[120] H Pottmann, M Peternell, and B Ravani. Approximation in Line Space - Applications in Robot Kinematics and Surface Reconstruction. *Advances in Robot Kinematics: Analysis and Control*, pages 403–412, 1998. 46, 109

[121] Helmut Pottmann, Michael Hofer, Boris Odehnal, and Johannes Wallner. Line Geometry for 3D Shape Understanding and Reconstruction. In Pajdla Tomás and Jiří Matas, editors, *Computer Vision - ECCV 2004*, pages 297–309, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. 46, 49, 50, 59

[122] Helmut Pottmann, Martin Peternell, and Bahram Ravani. An introduction to line geometry with applications. *Comput. Aided Des.*, 31:3–16, 1999. 12

[123] Helmut Pottmann and Thomas Randrup. Rotational and helical surface approximation for reverse engineering. *Computing*, 60(4):307–322, 1998. 49, 50, 59, 109

[124] Helmut Pottmann and Johannes Wallner. *Computational line geometry*. Springer Science and Business Media, 2009. 12, 26, 27, 39

[125] Helmut Pottmann, Johannes Wallner, and Stefan Leopoldseder. Kinematical methods for the classification, reconstruction, and inspection of surfaces. *Publications de l'Équipe de Mathématiques Appliquées (Numéro special)*, pages 51–60, 2001. 46, 49, 50, 59

[126] Luis Puig, Jesús Bermúdez, Peter Sturm, and José Jesús Guerrero. Calibration of omnidirectional cameras in practice: A comparison of methods. *Computer Vision and Image Understanding*, 116(1):120–137, 2012. 81

[127] Srikumar Ramalingam and Peter Sturm. A Unifying Model for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1309–1319, 2017. 82

[128] Pradeep Ranganathan and Edwin Olson. Gaussian Process for lens distortion modeling. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3620–3625, 2012. 82

[129] Carl Edward Rasmussen and Zoubin Ghahramani. Occam s Razor. In T Leen, T Dietterich, and V Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. 9

[130] Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian processes for machine learning, 2006. 2, 7, 8, 10, 20, 29, 40, 46, 62, 68, 72, 77, 78, 79, 82, 96, 103

[131] Syed Navid Raza, Hafiz Raza ur Rehman, Suk Gyu Lee, and Gyu Sang Choi. Artificial intelligence based camera calibration. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 1564–1569, 2019. 81

[132] Chiara Romanengo, Andrea Raffo, Yifan Qie, Nabil Anwer, and Bianca Falcidieno. Fit4CAD: A point cloud benchmark for fitting simple geometric primitives in CAD objects. *Computers and Graphics*, 102:133–143, 2022. 47

[133] Martin Rufli, Davide Scaramuzza, and Roland Siegwart. Automatic detection of checkerboards on blurred and distorted images. *2008 IEEE/RSJ Int. Conf. Intell. Robot. Syst. IROS*, pages 3121–3126, 2008. 68

[134] Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. Design and Analysis of Computer Experiments. *https://doi.org/10.1214/ss/1177012413*, 4(4):409–423, 11 1989. 103

[135] Adriana Salatino, Claudio Zavattaro, Roberto Gammeri, Emanuele Cirillo, Maria Luisa Piatti, Maria Pyasik, Hilary Serra, Lorenzo Pia, Giuliano Geminiani, and Raffaella Ricci. Virtual reality rehabilitation for unilateral spatial neglect: A systematic review of immersive, semi-immersive and non-immersive techniques. *Neuroscience I& Biobehavioral Reviews*, 152:105248, 9 2023. 99

[136] Mathieu Salzmann and Raquel Urtasun. Implicitly constrained Gaussian process regression for monocular non-rigid pose estimation. *Advances in Neural Information Processing Systems*, 23:2065–2073, 2010. 31, 37, 38

[137] Anna Saranti, Miroslav Hudec, Erika Mináriková, Zdenko Takáč, Udo Großschedl, Christoph Koch, Bastian Pfeifer, Alessa Angerschmid, and Andreas Holzinger. Actionable Explainable AI (AxAI): A Practical Example with Aggregation Functions for Adaptive Classification and Textual Explanations for Interpretable Machine Learning. *Machine Learning and Knowledge Extraction*, 4(4):924–953, 10 2022. 99

[138] B Sarath and K Varadarajan. Fundamental theorem of projective geometry. *Communications in Algebra*, 12(8):937–952, 1 1984. 85

[139] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A Toolbox for Easily Calibrating Omnidirectional Cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701, 2006. 68

[140] S Sels, B Bogaerts, S Vanlanduit, and R Penne. Extrinsic Calibration of a Laser Galvanometric Setup and a Range Camera. *Sensors (Basel, Switzerland)*, 18, 2018. 18

[141] Seppe Sels, Bart Ribbens, Steve Vanlanduit, and Rudi Penne. Camera Calibration Using Gray Code. *Sensors*, 19(2):246, 7 2019. 67, 92, 111

[142] Seppe Sels, Steve Vanlanduit, Boris Bogaerts, and Rudi Penne. Three-dimensional full-field vibration measurements using a handheld single-point laser Doppler vibrometer. *Mechanical Systems and Signal Processing*, 126:427–438, 2019. 20

[143] Amar Shah, Andrew Gordon Wilson, and Zoubin Ghahramani. Student-t processes as alternatives to Gaussian processes. In *Journal of Machine Learning Research*, volume 33, 2014. 10

[144] Jieying Shi, Ziheng Zhu, Jianhua Zhang, Ruyu Liu, Zhenhua Wang, Shengyong Chen, and Honghai Liu. CalibRCNN: Calibrating Camera and LiDAR by Recurrent Convolutional Neural Network and Geometric Constraints. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10197–10202, 2020. 1, 18

[145] Paul Smith, Ian D Reid, and Andrew J Davison. Real-Time Monocular SLAM with Straight Lines. In *British Machine Vision Conference*, 2006. 81

[146] Łukasz Sobczak, Katarzyna Filus, Adam Domański, and Joanna Domańska. LiDAR point cloud generation for SLAM algorithm evaluation. *Sensors*, 21(10):3313, 2021. 45

[147] J Sun, X Chen, Z Gong, Z Liu, and Y Zhao. Accurate camera calibration with distortion models using sphere images. *Optics Laser Technology*, 65:83–87, 2015. 82

[148] Jiachen Sun, Qingzhao Zhang, Bhavya Kailkhura, Zhiding Yu, Chaowei Xiao, and Z Morley Mao. Benchmarking robustness of 3d point cloud recognition against common corruptions. *arXiv preprint arXiv:2201.12296*, 2022. 47

[149] Weibin Sun, Xubo Yang, Shuangjiu Xiao, and Wencong Hu. Robust Checkerboard Recognition for Efficient Nonplanar Geometry Registration in Projector-Camera Systems. In *Proceedings of the 5th ACM/IEEE International Workshop on Projector Camera Systems*, PROCAMS '08, New York, NY, USA, 2008. Association for Computing Machinery. 67

[150] Xiaoling Sun, Bin Zhou, Weihao Xie, and Yuangeng Zhang. The design of laser scanning galvanometer system. In Ailing Tian, Anand Asundi, Weiguo Liu, and Chunmin Zhang, editors, *The International Conference on Photonics and Optical Engineering (icPOE 2014)*, volume 9449, page 676 – 684. SPIE, 2015. 17

[151] Laura P Swiler, Mamikon Gulian, Ari L Frankel, Cosmin Safta, and John D Jakeman. A survey of constrained Gaussian process regression: Approaches and implementation challenges. *J. Mach. Learn. Model. Comput.*, 1(2):119–156, 2020. 31

[152] Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 61(3), 1999. 52

[153] Michalis Titsias and Neil D Lawrence. Bayesian Gaussian Process Latent Variable Model. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 844–851, Chia Laguna Resort, Sardinia, Italy, 10 2010. PMLR. 53

[154] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019. 47

[155] Ondřej Vaniček, Michal Chaluš, and Jindřich Liška. 3D Vision Based Calibration Approach for Robotic Laser Surfacing Applications. In *2022 20th International Conference on Mechatronics-Mechatronika (ME)*, pages 1–6, 2022. 68

[156] Tamás Várady, Ralph R Martin, and Jordan Cox. Reverse engineering of geometric models—an introduction. *Computer-Aided Design*, 29(4):255–268, 1997. 45

[157] Simon Verspeek, Ivan De Boi, Xavier Maldague, Rudi Penne, and Gunther Steenackers. Dynamic Line Scan Thermography Parameter Design via Gaussian Process Emulation. *Algorithms*, 15(4), 2022. 112

[158] Vladimir Vezhnevets. OpenCV calibration object detection. 68

[159] Guohui Wang, Hao Zheng, and Xuan Zhang. A Robust Checkerboard Corner Detection Method for Camera Calibration Based on Improved YOLOX. *Front. Phys.*, 9(February):1–9, 2022. 68

[160] Zhizun Wang. Using Gaussian Process in Clockwork Variational Autoencoder for Video Prediction. In *2022 International Conference on Information Technology Research and Innovation, ICITRI 2022*, 2022. 111

[161] Andrew Gordon Wilson, Christoph Dann, and Hannes Nickisch. Thoughts on massively scalable Gaussian processes. *arXiv preprint arXiv:1511.01870*, 2015. 41, 110, 111

[162] Tobias Wissel, Benjamin Wagner, Patrick Stüber, Achim Schweikard, and Floris Ernst. Data-driven learning for calibrating galvanometric laser scanners. *IEEE Sensors Journal*, 15(10):5709–5717, 2015. 18, 26, 27, 29, 32, 34, 39, 110

[163] Hao Wu and Yi Wan. A highly accurate and robust deep checkerboard corner detector. *Electron. Lett.*, 57(8):317–320, 2021. 68

[164] Yubin Wu, Shixiong Jiang, Zhenkun Xu, Song Zhu, and Danhua Cao. Lens distortion correction based on one chessboard pattern image. *Frontiers of Optoelectronics*, 8:319–328, 2015. 83

[165] Paul Theo Zebhauser, Marine Vernet, Evelyn Unterburger, and Anna-Katharine Brem. Visuospatial Neglect - a Theory-Informed Overview of Current and Emerging Strategies and a Systematic Review on the Therapeutic Use of Non-invasive Brain Stimulation. *Neuropsychology Review*, 29(4):397–420, 11 2019. 96

[166] Yesheng Zhang, Xu Zhao, and Dahong Qian. Learning-Based Distortion Correction and Feature Detection for High Precision and Robust Camera Calibration. *IEEE Robotics and Automation Letters*, 7(4):10470–10477, 2022. 67, 68, 83

[167] Yihuan Zhang, Liang Wang, Xuhui Jiang, Yong Zeng, and Yifan Dai. An efficient LiDAR-based localization method for self-driving cars in dynamic environments. *Robotica*, pages 1–18, 2021. 17

[168] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000. 82, 84, 109

[169] Zhao Zheng, Xiaohua Xie, and Yang Yu. Image Undistortion and Stereo Rectification Based on Central Ray-Pixel Models. In *Artificial Intelligence and Robotics: 7th International Symposium, ISAIR 2022, Shanghai, China, October 21-23, 2022, Proceedings, Part II*, pages 40–55, 2022. 81

Appendix

Appendix A

Simplified Zhang's Method

The intrinsic camera matrix for a pinhole camera $\mathbf{K}$ can be written as

$$\mathbf{K} = \begin{pmatrix} fs_x & s_\theta & u_c \\ 0 & fs_y & v_c \\ 0 & 0 & 1 \end{pmatrix}, \tag{A.1}$$

in which f is the focal length, s_x and s_y are sensor scale factors, s_θ is a skew factor and (u_c, v_c) is the coordinate of the image centre with respect to the image coordinate system. However, real world cameras and their lenses suffer from imperfections. This introduces all sorts of distortions, of which radial distortion is the most commonly implemented. Calibrating this non-ideal pinhole camera, means finding values for both $\mathbf{K}$ and $[\mathbf{R} \mid \mathbf{t}]$, and whichever distortion model is implemented.

Zhang's method is based on the images of checkerboards with known size and structure. For each position of the board, we construct a coordinate system where the $X-$ and Y-axis are on the board and the Z-axis is perpendicular to it. We assign all checkerboard corners a 3D coordinate in this system with a Z-component zero. This allows us to rewrite Equation 7.1

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} \sim \mathbf{K}[\mathbf{r_1} \mid \mathbf{r_2} \mid \mathbf{t}] \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}, \tag{A.2}$$

in which $\mathbf{r_1}$ and $\mathbf{r_2}$ are the first two columns of $\mathbf{R}$. This equation shows a 2D to 2D correspondence known as a homography. This means we can write

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \mathbf{H} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}, \tag{A.3}$$

with $\mathbf{H}$ the 3x3 matrix that describes the homography. This matrix is only determined up to a scalar factor, so it has eight degrees of freedom. Each point correspondence yields two equations. Therefore, four point correspondences are needed to solve for $\mathbf{H}$. In practice, we work with several more points in an overdetermined system to compensate for noise in the measurements.

From these homographies, one for every position of the checkerboard, we estimate the camera intrinsics and extrinsic parameters. From Equation A.2 and A.3, we can write a decomposition

for $\mathbf{H}$, up to a multiple, as

$$\lambda \mathbf{H} = \lambda [\mathbf{h_1} \mid \mathbf{h_2} \mid \mathbf{h_3}] = \mathbf{K}[\mathbf{r_1} \mid \mathbf{r_2} \mid \mathbf{t}], \tag{A.4}$$

where λ is a scaling factor and $\mathbf{h_1}$, $\mathbf{h_2}$ and $\mathbf{h_3}$ are the columns of $\mathbf{H}$. We observe the following relationships:

$$\lambda \mathbf{K^{-1}} \mathbf{h_1} = \mathbf{r_1}, \tag{A.5}$$

$$\lambda \mathbf{K^{-1}} \mathbf{h_2} = \mathbf{r_2}. \tag{A.6}$$

Moreover, since $\mathbf{R}$ is a rotation matrix, it is orthonormal. This means $\mathbf{r_1}^T \mathbf{r_2} = 0$ and $\|\mathbf{r_1}\| = \|\mathbf{r_2}\|$. Combining these equations yields

$$\mathbf{h_1}^T \mathbf{K^{-T}} \mathbf{K^{-1}} \mathbf{h_2} = 0, \tag{A.7}$$

$$\mathbf{h_1}^T \mathbf{K^{-T}} \mathbf{K^{-1}} \mathbf{h_1} = \mathbf{h_2}^T \mathbf{K^{-T}} \mathbf{K^{-1}} \mathbf{h_2}. \tag{A.8}$$

These are now independent of the camera extrinsics.

We can write $\mathbf{K^{-T}} \mathbf{K^{-1}}$ as a new symmetric 3x3 matrix $\mathbf{B}$, alternatively by a 6-tuple $\mathbf{b}$. From Equations A.7 and A.8 we can write $\mathbf{A}\mathbf{b} = 0$, in which $\mathbf{A}$ is composed out of all known homography values of the previous step and $\mathbf{b}$ is the vector of six unknowns to solve for. For n checkerboards, and thus n homographies, we now have $2n$ equations. This means we need at least three checkerboard positions. Once $\mathbf{b}$ and thus $\mathbf{B}$ is found, we can calculate $\mathbf{K}$ via a Cholesky decomposition on $\mathbf{B}$. From $\mathbf{K}$, we know all camera intrinsics such as skewness, scale factor, focal length and principal point.

From Equations A.5 and A.6 we can determine $\mathbf{r_1}$ and $\mathbf{r_2}$. The scaling factor λ can be found by normalising $\mathbf{r_1}$ and $\mathbf{r_2}$ to unit length. Building on the orthogonality of the rotation matrix $\mathbf{R}$, we can write

$$\mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2}. \tag{A.9}$$

Lastly, we find

$$\mathbf{t} = \lambda \mathbf{K}^{-1} \mathbf{h_3}. \tag{A.10}$$

Up until this point, we have assumed an ideal pinhole camera model. MATLAB and OpenCV use this as a first step in an iterative process in which they introduce extra intrinsic camera parameters to account for image distortion. After convergence, a compromise is found for all camera parameters.

In Chapter 7, we construct an ideal virtual GP-camera. The Gaussian processes capture all distortions and other imperfections in a pre-processing step. This means that the images of the checkerboards on the virtual image plane are projections of a perfect checkerboard, up to noise. This allows us to simplify Zhang's method as follows.

First, since there is no skewness in the virtual image plane and all virtual pixels are squares, we can rewrite the intrinsic camera matrix $\mathbf{K}$ as

$$\mathbf{K} = \begin{pmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{pmatrix}. \tag{A.11}$$

This results in

$$\mathbf{B} = \mathbf{K^{-T}} \mathbf{K^{-1}} = \begin{pmatrix} \frac{1}{f^2} & 0 & \frac{-u_c}{f^2} \\ 0 & \frac{1}{f^2} & \frac{-v_c}{f^2} \\ 0 & 0 & \frac{u_c}{f^2} + \frac{v_c}{f^2} + 1 \end{pmatrix}. \tag{A.12}$$

The rest of the procedure is similar to Zhang's method. We combine Equations A.7, A.8 and A.12 into the system $\mathbf{Ab} = 0$. The vector $\mathbf{b}$ is now the vector of four unknowns to solve for, instead of six. For n checkerboards, and thus n homographies, we still have $2n$ equations. This means we need at least two checkerboard positions to be able to solve this, instead of three. As before, more positions provide more equations, which are solved via Singular Value Decomposition (SVD). Notice that the form of Equation A.12 is such that we do not have to perform the Cholesky decomposition anymore.

Second, there is no need for a distortion model, nor for a converging iterative process. The camera calibration is reduced to a one-step analytical calculation.