Precision of Pose Estimation using Corner Detection

Matthew J. Edwards

A thesis presented for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

1 March 2022

ABSTRACT

The aim of this research was to develop a method for recording ground truth with performance comparable to motion capture, in order to produce high-quality outdoor visual odometry datasets. A novel fiducial marker system was developed, featuring a smooth pattern which is used in an optimisation process to produce refined estimates. On average, precision was increased by 27 % compared to traditional fiducial markers. To investigate the limit of the increase in pose estimation precision possible with this method, the marker was modelled as a dense grid of checkerboard corners and the Cramér-Rao lower bound of the corresponding estimator was derived symbolically. This gave a lower bound for the variance of a pose estimated from a given image. The model was validated in simulation and using real images.

The distribution of the error for a common checkerboard corner detector was evaluated to determine whether modelling it using independent and identically distributed Gaussian random variables was valid. In a series of experiments where images were collected from a tripod, a robot arm, and a slider-type electric actuator, it was determined that the error is usually normally distributed but its variance depends on the amount of lens blur in the image, and that any amount of motion blur can produce correlated results. Furthermore, in images with little blur (less than approximately one pixel) the estimates are biased, and both the bias and the variance are dependent on the location of the corner within a pixel. In real images, the standard deviation of the noise was around 80 % larger at the pixel edges than at the centre. The intensity noise from the image sensor was also found not to be identically distributed: in one camera, the standard deviation of the intensity noise varied by a factor of approximately four within the region around a checkerboard corner.

This research suggests that it is possible to significantly increase fiducial marker pose estimation precision, presents a novel approach for predicting and evaluating pose estimation precision, and highlights sources of error not considered in prior work.

Deputy Vice-Chancellor's Office
Postgraduate Research Office

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Publication 1: Edwards, M.J., Hayes, M.P. and Green, R.D. (2016), 'High-accuracy fiducial markers for ground truth', In *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Palmerston North, New Zealand, pp. 1–6.

Parts of Chapters 1 and 3 present material from this publication.

Publication 2: Edwards, M.J., Hayes, M.P. and Green, R.D. (2017), 'Statistical Lower Bound for Variance of Checkerboard Pose Estimate', In *2017 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Christchurch, New Zealand, pp. 1–6.

Parts of Chapters 2 and 4 present material from this publication.

Publication 3: Edwards, M.J., Hayes, M.P. and Green, R.D (2018), 'Error Distribution of Estimated Checkerboard Corner Location', In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Auckland, New Zealand, pp. 1–6.

Parts of Chapter 5 present material from this publication.

Publication 4: Edwards, M.J., Hayes, M.P. and Green, R.D. (2020), 'Experimental Validation of Bias in Checkerboard Corner Detection', In *2020 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Wellington, New Zealand, pp. 1–6.

Parts of Chapter 6 present material from this publication.

Please detail the nature and extent (%) of contribution by the candidate:

Publication 1: The methodology was developed through discussions between Edwards, Hayes, and Green. The data were collected by Edwards; Hayes, Green, Dave Healy, and Zane Barker assisted with the initial setup. The derivation was produced by Hayes. The software, results, figures, and writing were produced by Edwards. Hayes and Green provided comments and review. 80 % of the research and 95 % of the writing were contributed by Edwards.

Publication 2: The methodology was proposed by Hayes. The data were collected by Edwards; Green assisted with the initial setup. The software, results, figures, and writing were produced by Edwards. Hayes and Green provided comments and review. 50 % of the research and 75 % of the writing were contributed by Edwards.

Publication 3: The methodology was developed through discussions between Edwards, Hayes, and Green. The data were collected by Edwards. The mathematical model for corner refinement bias was produced by Hayes. The software, results, figures, and writing were produced by Edwards. Hayes and Green provided comments and review. 90 % of the research and 95 % of the writing were contributed by Edwards.

Publication 4: The methodology was developed through discussions between Edwards and Hayes. The data were collected by Edwards; Hayes, Morgan King, and Sam Schofield assisted with the initial setup. The software, results, figures, and writing were produced by Edwards. Hayes and Green provided comments and review. 90 % of the research and 95 % of the writing were contributed by Edwards.

Certification by co-authors

If there is more than one co-author then a single co-author can sign on behalf of all. The undersigned certifies that:

- The above statement correctly reflects the nature and extent of the Doctoral candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work, he or she wrote the text

Name: *Dr. Michael P. Hayes* Signature: MPH Date: 29 July 2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

CRLB	Cramér-Rao lower bound
FFT	fast Fourier transform
FIM	Fisher information matrix
GPS	Global Positioning System
i.i.d.	independent and identically distributed
IMU	inertial measurement unit
INS	inertial navigation system
P3P	perspective-three-point
PDF	probability distribution function
PnP	perspective-n-point
RMS	root-mean-square
ROS	Robot Operating System
RTK	real-time kinematic
SLAM	simultaneous localisation and mapping
SNR	signal-to-noise ratio
UAV	unmanned aerial vehicle
VIO	visual-inertial odometry
VO	visual odometry

PREFACE

This research was aligned with the 2016–2018 MBIE Smart Ideas project, UOCX1601 "Autonomous forest pruning and data collection of tree metrics". When I helped put that bid together, we aimed to address the issue of clearwood supply in the New Zealand forestry industry. In short, for a variety of complicated reasons, the forestry sector has been steadily moving toward growing pine trees as quickly and cheaply as possible in order to export bulk unprocessed logs. This is impacting the timber processing industry, which relies on a steady supply of *clearwood* (wood with few knots or defects) in order to

Figure 1 A late prototype of the tree pruning robot, shown in the indoor aerial pruning test environment we set up in the motion capture lab. Visible on the prototype from left to right are: the sixth iteration of the pruning tool, the camera stabilisation rig for visual odometry cameras (with the front camera missing) and its integrated pruner gimbal, the pruner motion controller I designed and programmed, the second generation of the custom airframe, and the IMU stabilisation rig (below the onboard computer). Out of view is the custom interface board I designed for the IMU and the 120 W power supply I designed to power the onboard computer from the flight battery.

produce appearance-grade wood products. Producing clearwood has several downsides. The trees must be pruned several times throughout their lifetime: an operation that is physically difficult, time-consuming, and somewhat hazardous. The trees must be planted further apart in order to maximise clearwood production per tree, which reduces the amount of wood overall, and even then only the wood that grows over the pruned area on each tree is clearwood. Finally, the value of a plantation must be estimated several decades in advance to determine profitability, and that value depends on political and geopolitical factors which are hard to predict.

Our idea was to develop an aerial robot that could prune trees. This would largely eliminate a job that the forestry industry does not particularly want, increase clearwood production per tree by making it practical to prune further up the trunk than usual, and encourage better land utilisation by enabling pruning on land too steep for human pruners to practically operate on. The idea garnered a lot of interest from Scion, who want to restructure the forestry industry around high-value products; the New Zealand Forest Owners Association and various sawmill owners, who want more pruned logs;

Figure 2 A photo from a field trip to a commercial forest plantation near Hatepe, Waikato, in the North Island of New Zealand. These trees are around the age at which they would have their second pruning, which is the first we envisioned performing in our project (since the first pruning begins at ground level and it would be difficult for a UAV to access the trunk). Note the heavy undergrowth, and imagine trying to navigate from tree to tree with a medium-sized UAV without getting stuck in the brush. Thanks to Richard Parker from Scion for organizing the trip and for this photo, and to Lake Taupō Forest Trust for welcoming us onto their land.

and several large forest managers, who want methods for pruning their trees which will stay economically feasible despite uncertain clearwood prices. As a side benefit, the robots could collect extremely detailed data as they navigate through the forest, which forest managers were equally interested in.

The concept for the robot was a multirotor UAV that would localise itself, map, and navigate inside a forest, using 3D SLAM and simple semantic models of the trees. A swarm of these robots bearing lightweight pruning tools would collaborate to cover a specified block of forest, pruning all the branches one by one until whatever criteria the forest manager chooses are satisfied. At the time, we thought the hardest part would be developing and testing the SLAM algorithm. My personal concern was that we would lack a good evaluation method for it, given that none of the usual techniques for recording ground truth camera poses work well in that environment, which lead to the research in this thesis. As it turned out, the hard part was all the practical things, and we spent year after year iterating on UAV airframes and pruning tool designs and so on and ran out of time to do much relevant research. We did end up with an impressive-looking prototype (as shown in Figure 1), and performed various convincing-looking demonstrations of fully autonomous tree pruning for a variety of audiences, both in the lab and in pine forests.

Field trips to our industry collaborators' various plantations (see Figure 2) impressed upon us the sheer scale of the challenge we had undertaken. For example, some areas are so heavily vegetated that forestry workers habitually use their ladders to clamber over the undergrowth. Since the project officially started in late 2016, the focus in visual odometry research has gone from mostly increasing performance in indoor and urban environments to a broad push for robustness in all environments (fuelled in part by the explosion in UAV research at around the same time), even including groups demonstrating UAVs flying along tramping tracks through the woods. Even so, the state of the art in 2021 still falls well short of what is needed for an application like this. The other area we wildly underestimated was the actual interaction with the pruner. We assumed that once we had an appropriately equipped UAV and a system for precisely estimating the positions of the UAV and a branch, we could just make one intersect with the other and move on. Once we developed a reliable gimbal system so we could move the UAV without the tilt throwing out our pruner positioning, went to great lengths to ensure the physical gimbal matched our digital models so we could position the tool exactly, and carefully tuned the gimbal controller to achieve low latency without jerking the UAV around through reaction torque, we found that our UAV's position control capability was insufficient for performing cuts without exceeding the industry 5 mm branch stub limit. We were right up against the physical limits imposed by the propulsion system itself, and even after a redesign that sacrificed battery life for agility we failed to achieve better than a few centimetres of hover precision, which was (and still is) in line with the state of the art for small multirotor UAVs.

As it turned out, we had stumbled into the field of aerial manipulation, where an increasingly large community of researchers have spent over a decade chipping away at problems similar to ours. When we visited several UAV research centres and attended an aerial manipulation workshop at a large robotics conference in Europe, we discovered a rich history of workarounds (but not really solutions) for the same core problems we were struggling with. After the Smart Ideas project wound down we revised our goal, and are now collaborating with engineers and scientists across the country to develop a simplified, robust aerial manipulation platform that would help NZ's robotics industry bring automation to new niche applications.

Meanwhile, my research progressed in a different direction. I intended to whip up a method for collecting accurate ground truth for visual odometry in a forest, then go on to develop a visual odometry algorithm that would let us jump straight from pruning tree stand-ins in the lab to pruning real trees in the field. After some initial exploration in which I found very little existing work, I developed a new kind of high-precision fiducial marker in the hope that it would be precise enough to solve the ground truth problem. It was not, so I then examined the factors that determine the precision of pose estimates (using checkerboards as a generic stand-in), and then went one level deeper and examined the noise distribution of corner estimates. The four papers I published on those topics form the main four chapters of this thesis:

Edwards, M.J., Hayes, M.P. and Green, R.D. (2016), 'High-accuracy fiducial markers for ground truth', In *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Palmerston North, New Zealand, pp. 1–6.

Edwards, M.J., Hayes, M.P. and Green, R.D. (2017), 'Statistical Lower Bound for Variance of Checkerboard Pose Estimate', In *2017 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Christchurch, New Zealand, pp. 1–6.

Edwards, M.J., Hayes, M.P. and Green, R.D (2018), 'Error Distribution of Estimated Checkerboard Corner Location', In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Auckland, New Zealand, pp. 1–6.

Edwards, M.J., Hayes, M.P. and Green, R.D. (2020), 'Experimental Validation of Bias in Checkerboard Corner Detection', In *2020 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Wellington, New Zealand, pp. 1–6.

Further to my work on the tree pruning project and my research, I built the Computer Science department's UAV lab from a storage room and a cupboard of old hand tools

into a fully equipped mechatronics workshop with a collection of commercial and custom-made UAVs, a motion capture system, a test flight cage, a high-end 3D printer, a virtual reality setup, and a dedicated research engineer/drone operator; spent a semester as a lecturer for the final-year embedded systems paper while a colleague was on sabbatical; gave a few lectures in the final-year robotics and computer vision papers; did extensive course development for the final-year embedded systems and robotics papers; supervised six final-year engineering group projects, five summer students, and one Master's student, and guided a few more Master's and PhD students less officially; and helped write six significant MBIE grant applications. I helped my colleagues and students with another three papers:

Schofield, S.D., Edwards, M.J. and Green, R.D. (2017), 'Sensitivity analysis of multirotor position control', In *2017 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Christchurch, New Zealand, pp. 1–6.

Schofield, S.D., Edwards, M.J. and Green, R.D. (2018), 'Calibration for Camera–Motion Capture Extrinsics', In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Auckland, New Zealand, pp. 1–6.

Lee, D., Muir, W., Beeston, S., Bates, S., Schofield, S.D., Edwards, M.J. and Green, R.D. (2018), 'Analysing Forests Using Dense Point Clouds', In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Auckland, New Zealand, pp. 1–6.

Overall, this thesis quantifies the performance of checkerboard pose estimators and examines the limitations of pose estimation in computer vision with regard to precision, combining original research with concepts from other fields.

Chapter 1

INTRODUCTION

Visual odometry (VO) is the process of estimating the motion of an object relative to its environment (its *egomotion*) using one or more cameras or other visual sensors (LiDAR, time-of-flight, etc.) attached to it [Scaramuzza and Fraundorfer 2011]. The change in pose is calculated for each pair of successive camera frames, producing a locally consistent trajectory. The frame-to-frame errors accumulate over time, causing the estimated trajectory to drift away from the real path. In visual-inertial odometry (VIO), inertial measurement units (IMUs) are used as well, which is particularly useful in monocular VO where scale is otherwise unobservable, and when visual approaches struggle, for example, over illumination changes, through textureless areas, and when there is motion

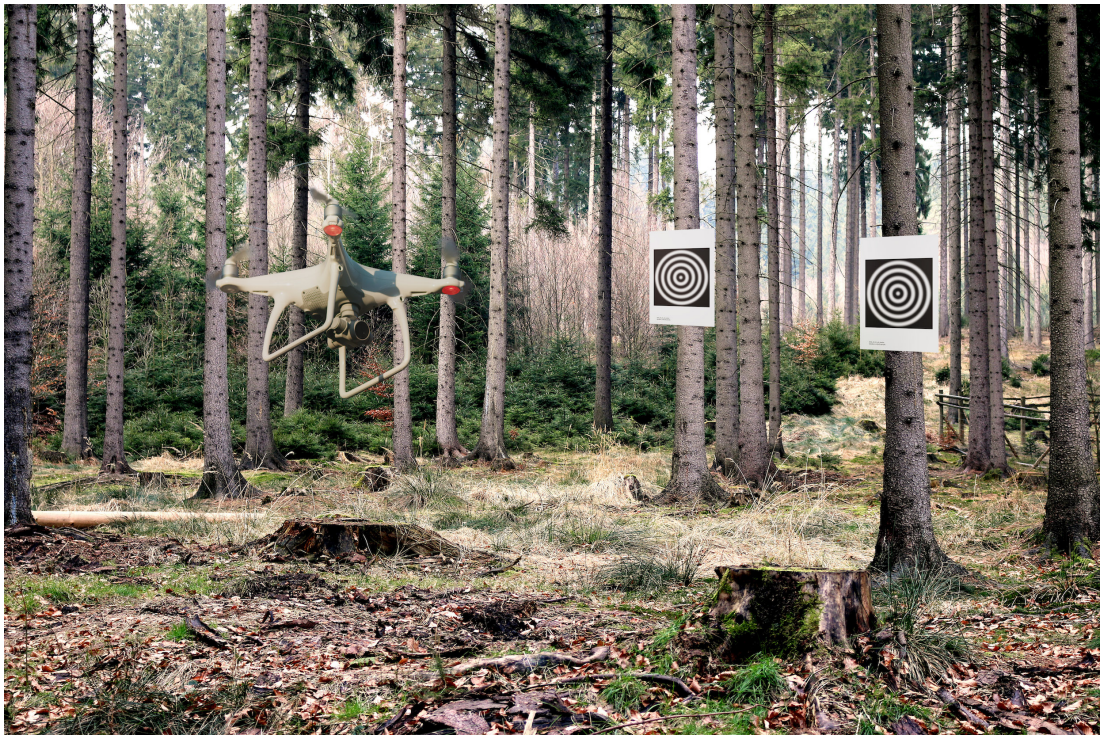


Figure 1.1 An artist's impression of a UAV flying through a forest, with the high-precision fiducial markers from Chapter 3 used to capture ground truth for evaluating its visual positioning system.

blur [Qin et al. 2018]. Visual simultaneous localisation and mapping (SLAM) extends visual odometry by also building up a map of the environment. Detecting when a previously visited area is re-visited (*loop closure*) allows the map and trajectory to be jointly optimised, eventually producing a geometrically consistent map and a trajectory with little long-term drift [Campos et al. 2020]. In practice, VO algorithms often maintain a map of recently observed features and perform real-time local optimisation to reduce short-term drift—blurring the line between VO and visual SLAM—and visual SLAM algorithms are often implemented as an addition to a VO algorithm that performs loop closure and global optimisation. The distinction is that the goal of VO is egomotion estimation, while the goal of SLAM is to build a map and localise within it.

This work is motivated by the lack of VO datasets with accurate ground truth for a niche application: flying small multi-rotor unmanned aerial vehicles (UAVs) through commercial forest plantations. VO and visual SLAM algorithms are commonly evaluated indoors using motion capture systems [Fraundorfer et al. 2012, Geneva et al. 2019, Kneip et al. 2011a, Qin et al. 2018, Rosinol et al. 2020, Shen et al. 2013, 2015], which typically measure pose with precision on the order of 0.5 mm and 0.05 degrees. However, motion capture systems require a controlled environment and do not have large enough tracking volumes for outdoor use. The EuRoC MAV dataset [Burri et al. 2016], a widely used stereo visual-inertial dataset collected using a small UAV, comprises sequences from two environments: a motion capture lab which is $8\,\mathrm{m} \times 8.48\,\mathrm{m} \times 4\,\mathrm{m}$ and a machine hall which is approximately $15\,\mathrm{m} \times 20\,\mathrm{m}$. In the machine hall, the only pose ground truth is low-rate 3D position measured using a robot total station,[1] the accuracy of which was described by the authors as "deteriorating under very dynamic motions" such as those exhibited in some of the sequences.

The KITTI outdoor VO dataset [Geiger et al. 2012b] uses a high-end inertial navigation system (INS) that fuses real-time kinematic (RTK) Global Positioning System (GPS) solutions with a 6-axis IMU for its ground truth. The manufacturer specifies an open-sky position error of 0.1 m and an orientation error of 0.03, 0.03 and 0.1 degrees for pitch, roll and heading, with a total weight of 3.4 kg and power consumption of 20 W. This represents the best-case scenario for GPS ground truth: open sky, clear line of sight to an RTK base station, and no limit on size, weight, power, or cost. Even so, the authors had to modify their benchmarking procedure to work around bias in the ground truth positions.[2] In a review of GPS accuracy under a forest canopy, Kaartinen et al. [2015] found that even their reference receiver (a survey-grade INS much like the one used for the KITTI dataset) had 0.7 m RMS error, which gives some indication of

[1] A robot total station uses a laser to measure the angle and range of a retroreflector with millimetre accuracy, but cannot measure orientation.

[2] The VO page on the website for the KITTI dataset says "On 03.10.2013 we have changed the evaluated sequence lengths from $(5, 10, 50, 100, \ldots, 400)$ to $(100, 200, \ldots, 800)$ due to the fact that the GPS/OXTS ground truth error for very small sub-sequences was large and hence biased the evaluation results" [Geiger et al. 2012a].

the performance of high-end GPS in less ideal conditions.

A meaningful evaluation requires ground truth that is more accurate than the VO algorithm. Currently, the best-performing VO algorithm on the popular KITTI autonomous driving VO benchmark (SOFT [Cvišić et al. 2018]) has a translational error of 0.38 %–0.85 % (average 0.57 %) and rotational error of 0.0005 °/m–0.0021 °/m (average 0.0010 °/m).[3] That means that on average, the accumulated error after travelling 100 m would be 0.57 m and 0.1 degrees. This is approaching the nominal orientation accuracy for the KITTI dataset's ground truth, which suggests that the KITTI dataset is no longer sufficient for evaluating state-of-the-art VO algorithms. A similar survey-grade INS operating in a forest environment would perform dramatically worse, not to mention that it would be too large and heavy to carry on a UAV small enough to fly between trees, so this is not useful for the forestry application.

The ETH3D multi-view stereo dataset [Schöps et al. 2017] captured a high-resolution colour point cloud of the environment using a laser scanner, and two sets of images: high-resolution images taken with a DSLR and stereo images taken with a set of four synchronised global-shutter machine vision cameras. The images were then aligned with the point cloud using offline structure-from-motion methods. There is no ground truth for the resulting camera poses, but the authors described the disparity images as "generally pixel-accurate" compared to the structure ground truth. For the PennCOSYVIO visual-inertial dataset [Pfrommer et al. 2017], the environment was prepared with 170 AprilTag fiducial markers, with the marker poses recorded using manual measurement and corrected using recent architectural models of the building and surrounds. Four image sequences were captured using a seven-camera rig, with the marker placement and trajectory carefully planned such that wide-baseline triangles of markers were visible throughout to improve pose estimation performance. Three of the cameras were used to estimate ground truth camera poses using a pose graph optimisation method. The resulting trajectory was described by the authors as "accurate to better than 10 cm along most of the path."

Other possible methods include simply taking off and landing at the same place (as precisely as possible) then comparing the estimated start and end locations (as demonstrated by Engel et al. [2012]), starting and ending the flight in a motion capture volume (as demonstrated by Schubert et al. [2018]), and starting and ending with a camera calibration sequence (as Schöps et al. [2019] did for their outdoor sequences). These do at least give a good measure of total accumulated drift. Total stations can be used for position ground truth, but only within line of sight, which is an issue in a forest environment. LiDAR odometry algorithms are sometimes used as ground

[3] SOFT is the best-performing algorithm on the KITTI benchmark, excluding the algorithms that use LiDAR. The method is named "SOFT2" in the KITTI table, which is an improved version of the original SOFT [Cvisic and Petrovic 2015], as mentioned in the SOFT-SLAM publication [Cvišić et al. 2018].

truth, but perform only slightly better than visual odometry in recent work (see e.g., V-LOAM [Zhang and Singh 2015] vs. SOFT [Cvišić et al. 2018]). Manually drawing a trajectory on an aerial map is arguably useful for ground vehicles on roads but much less so for UAVs under tree cover. Thus, in order to rigorously evaluate the performance of visual odometry algorithms in forest environments, a new ground truth method is required.

This thesis is an investigation into high-precision pose estimation using corner detection.

1.1 MAJOR CONTRIBUTIONS

There are three major contributions.

First, a new approach to pose estimation using fiducial markers is proposed that is more precise than existing algorithms. Rather than using only corners or edges for pose estimation, the marker has a radial sinusoid pattern that has a predictable appearance under perspective projection. The marker pose is initially estimated using traditional methods, then refined using a novel optimisation method in which a rendering of the marker is compared with the image. On average, pose estimation precision was increased by 27 % compared to traditional fiducial markers. This approach is a promising avenue for future research.

Second, an analytic model is derived for the Cramér-Rao lower bound (CRLB) of pose estimation using a checkerboard (or fiducial marker) pose estimator. The model gives a lower bound for the variance (and covariance) of the estimate, which effectively predicts the precision of the most accurate pose estimate from a given set of data. Both a Monte Carlo simulation and real data validate this model. The model can both predict estimator precision before data is available and evaluate the performance of a real estimator on real data compared to the theoretical precision limit. One significant finding is that generalising measurements of pose estimation precision is difficult: results from different cameras are not directly comparable, and the performance for a marker in one pose does not trivially predict the performance for another pose. This method for predicting and evaluating pose estimation precision has not been considered in prior research.

Third, OpenCV's sub-pixel corner refinement algorithm is found to introduce significant bias and noise which is dependent on the sub-pixel corner location. In real images, the standard deviation of the noise ranged from around 0.013 px at the pixel centre to 0.0072 px at the edges, a difference of around 1.8×. The bias could not be determined from the real images due to residual lens distortion, so the images were reproduced in simulation, where the bias had a maximum magnitude of 0.043 px. The overall noise variance depends on the amount of lens blur in the image, so images that are not equally

sharp throughout (for example, images with checkerboards at different depths or images which have been rectified) can have different noise distributions in different image regions. In images with significant (more than approximately one pixel) lens blur, the sub-pixel-dependent component is swamped by independent noise. Furthermore, corner refinement in images with any amount of motion blur can produce correlated results. These effects make the OpenCV sub-pixel corner refinement algorithm unsuitable for some precision applications.

1.2 THESIS STRUCTURE

The rest of this thesis is structured as follows: Chapter 2 outlines the notation and background theory for the subsequent chapters; Chapter 3 presents the fiducial marker system; in Chapter 4, a theoretical lower bound is derived for checkerboard pose estimation variance and used to investigate its dependence on various parameters; in Chapter 5, simulations are used to show that the estimation noise and bias of OpenCV's sub-pixel corner refinement algorithm measurably depends on the sub-pixel location of the corner; in Chapter 6, that effect is demonstrated in realistic conditions using a real camera and examined in the context of research from other areas; and Chapter 7 summarises the findings and gives recommendations for future work.

Chapter 2

BACKGROUND

This chapter introduces the notation used in the subsequent chapters and outlines background theory.

Aside: Asides
Details in these sections are not strictly necessary for the following chapters, but provide a deeper understanding and may help when reading related work.

2.1 BASIC NOTATION

Scalars are notated as lower-case letters, e.g., s. Vectors are notated as lower-case bold letters, e.g., $\mathbf{v}$. Matrices are notated as upper-case bold letters, e.g., $\mathbf{M}$. Random variables and processes are notated as upper-case letters, e.g., R. Some scalar constants are also notated as upper-case letters, e.g., N and M. Random vectors and matrices (and multivariate random processes) are notated as upper-case bold letters and are explicitly distinguished from matrices where necessary.

2.2 ROTATIONS

The coordinate systems used are all right-handed. Points are represented as column vectors from the origin to a position. Pre-multiplying a point by a *rotation matrix*,

$$\mathbf{R}(\theta_z, \theta_y, \theta_x) = \mathbf{R}_x(\theta_x)\mathbf{R}_y(\theta_y)\mathbf{R}_z(\theta_z), \tag{2.1}$$

where $\mathbf{R}_x$, $\mathbf{R}_y$, and $\mathbf{R}_z$ are the *elemental rotations*

$$\mathbf{R}_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix}, \tag{2.2}$$

$$\mathbf{R}_y(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}, \tag{2.3}$$

and

$$\mathbf{R}_z(\theta_z) = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.4}$$

rotates the point around the origin by θ_z counterclockwise about the z-axis, then θ_y counterclockwise about the y-axis, then θ_x counterclockwise about the x-axis. For example,

$$\mathbf{R}(0°, 0°, 90°) \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Rotation matrices can also represent an orientation (as a rotation from some reference frame) or change the reference frame in which a point is represented. Rotation matrices are generally referred to without their arguments (i.e., $\mathbf{R}_x$ instead of $\mathbf{R}_x(\theta_x)$). A rotation can also be represented using just the arguments $(\theta_z, \theta_y, \theta_x)$; these are often referred to as *Euler angles*, although not entirely correctly. In general, rotations can also be represented in various other useful ways: Davenport angles, axis-angle, unit quaternions, Rodrigues vectors, and so on [Terzakis et al. 2018].

Aside: More about rotation matrices

A real, square matrix $\mathbf{R}$ is a rotation matrix if and only if it is orthogonal ($\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}$) and has determinant 1 ($\det \mathbf{R} = 1$). The inverse of an orthogonal matrix is its transpose ($\mathbf{R}^{-1} = \mathbf{R}^{\mathrm{T}}$). Rotations are not commutative, apart from 2D rotations about the same point. The special orthogonal group $SO(3)$ is the set of all 3D rotation matrices.

A square, real, orthogonal matrix with determinant -1 is an *improper rotation matrix*, which performs a rotation and a reflection. These are sometimes referred to as rotation matrices (but not here).

A rotation matrix can be considered as representing an *active transformation* that changes the position of a point or a *passive transformation* that changes the coordinate system in which a point is described.[a] Both senses are used here. A rotation matrix can either pre-multiply a column vector or post-multiply a row vector, producing rotations in opposite directions.[b] Only column vectors and

pre-multiplication are used here.

A rotation matrix as defined in (2.1) performs a counterclockwise rotation in a right-handed coordinate system (as used here) or a clockwise rotation in a left-handed coordinate system.

A 3D rotation matrix performs a change of basis from the standard basis,

$$\left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\},$$

to a new set of basis vectors (the columns of the matrix) which are related to the standard basis by a rotation.[c] For example, the basis for an x-axis rotation is

$$\left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \cos\theta_x \\ \sin\theta_x \end{pmatrix}, \begin{pmatrix} 0 \\ -\sin\theta_x \\ \cos\theta_x \end{pmatrix} \right\},$$

which is each element of the standard basis rotated by θ_x counterclockwise about the x-axis.

Decomposing a general rotation into elemental rotations is ambiguous; there are many sequences of three elemental rotations that can represent any 3D rotation. These sequences are called *Davenport rotations*. Davenport rotations are either made up of three rotations about different axes (e.g., x, then y, then z) or a rotation about one axis, then another, then the first again (e.g., x, then y, then x). Rotations about three different axes are called *Tait-Bryan rotations*, and rotations about two different axes are called *Euler rotations*. The angles of each rotation from a sequence are called *Davenport angles*, *Tait-Bryan angles*, and *Euler angles*. Note that the term "Euler angles" is commonly used to specifically refer to Tait-Bryan angles rather than strict Euler angles.

A sequence of rotations can be extrinsic or intrinsic. *Extrinsic rotations* are performed using a fixed coordinate system. *Intrinsic rotations* are performed using a rotating coordinate system, which is initially aligned with the fixed coordinate system and moves with each rotation. An intrinsic rotation about the x-axis, then the y-axis, then the z-axis is produced by the rotation matrix

$$\mathbf{R} = \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z.$$

This rotation sequence is referred to as x-y'-z''. Each pre-multiplication rotates its operand with respect to the fixed axes, so $\mathbf{R}_x$ performs a rotation about the

fixed x-axis, $\mathbf{R}_y$ performs a rotation about the now-rotated y-axis (y'), and $\mathbf{R}_z$ performs a rotation about the now-twice-rotated z-axis (z''). Note that this is the same rotation as the extrinsic sequence z-y-x. The roll-pitch-yaw angles used in aviation are a commonly used intrinsic rotation sequence. Sometimes, the term "Euler angles" is used to mean the angles of an intrinsic (rather than extrinsic) rotation sequence.

[a] Active transformations are sometimes called alibi transformations, and passive transformations are sometimes called alias transformations.
[b] That is, $\mathbf{R}\mathbf{v}$ and $\mathbf{v}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}$ represent the same rotation.
[c] In the passive-rotation sense, the new basis vectors point in the directions of the axes of a new reference frame.

2.3 TRANSFORMATIONS

A *reference frame* consists of a set of coordinate axes and an origin point. Reference frames are notated by a lowercase letter in braces. The world (static/fixed) frame is $\{w\}$; all other frames are defined by their orientation and origin with respect to $\{w\}$. A frame $\{b\}$ is represented as the transformation from another frame $\{a\}$ to $\{b\}$: a rotation $\mathbf{R}_{ab} \in SO(3)$, the orientation of $\{b\}$ in $\{a\}$, followed by a translation $\mathbf{t}_{ab} \in \mathbb{R}^3$, the origin of $\{b\}$ in $\{a\}$.[1] The transformation from the world frame to a frame is notated with a single subscript (e.g., $\mathbf{T}_b = \mathbf{T}_{wb}$).

A point $\mathbf{p}$ in a frame $\{b\}$ is notated as

$$\mathbf{p}_b = \begin{pmatrix} p_{b,x} \\ p_{b,y} \\ p_{b,z} \end{pmatrix}. \tag{2.5}$$

Lists of points are notated as

$$\mathbf{p}_b^{(i)} = \begin{pmatrix} p_{b,x}^{(i)} \\ p_{b,y}^{(i)} \\ p_{b,z}^{(i)} \end{pmatrix}, \tag{2.6}$$

where i is an index. The transformation $(\mathbf{R}_{ab}, \mathbf{t}_{ab})$ can be used to change the frame in which a point is represented:

$$\mathbf{p}_a = \mathbf{R}_{ab}\mathbf{p}_b + \mathbf{t}_{ab}. \tag{2.7}$$

It can be notated more conveniently using homogeneous coordinates as the 4×4 *trans-*

[1] The orientation of $\{b\}$ in $\{a\}$ is the rotation from $\mathbf{R}_{wa}$, the orientation of $\{a\}$ in the world frame, $\{w\}$, to $\mathbf{R}_{wb}$, the orientation of $\{b\}$ in $\{w\}$. The origin of $\{b\}$ in $\{a\}$ is the translation from the origin of $\{a\}$ in $\{w\}$ to the origin of $\{b\}$ in $\{w\}$.

formation matrix

$$\mathbf{T}_{ab} = \begin{bmatrix} \mathbf{R}_{ab} & \mathbf{t}_{ab} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.8}$$

Then,

$$\begin{pmatrix} \mathbf{p}_a \\ 1 \end{pmatrix} = \mathbf{T}_{ab} \begin{pmatrix} \mathbf{p}_b \\ 1 \end{pmatrix}. \tag{2.9}$$

For convenience, homogeneous coordinates and Cartesian coordinates are used interchangeably.[2] Thus, (2.9) can be expressed as

$$\mathbf{p}_a = \mathbf{T}_{ab}\mathbf{p}_b. \tag{2.10}$$

Note that $\mathbf{T}_{ab}$ represents the transformation from $\{a\}$ to $\{b\}$, but pre-multiplication by it transforms a point from $\{b\}$ into $\{a\}$.

Aside: More about transformation matrices
The inverse of a transformation matrix $\mathbf{T}_{ab}$ is

$$\begin{aligned} \mathbf{T}_{ab}^{-1} &= \mathbf{T}_{ba} \\ &= \begin{bmatrix} \mathbf{R}_{ab}^{-1} & -\mathbf{R}_{ab}^{-1}\mathbf{t}_{ab} \\ \mathbf{0} & 1 \end{bmatrix}. \end{aligned}$$

Although a transformation matrix can perform arbitrary linear, affine, and projective transformations, only rigid-body transformations (rotation and translation) are used here. The special Euclidean group $SE(3)$ is the set of all such transformation matrices.

The action of a transformation matrix that represents a frame (e.g., $\mathbf{T}_{wa}$) is to convert a point from one frame to another (e.g., from $\{a\}$ to $\{w\}$). Consider a point $\mathbf{p}_a$. It is defined relative to the orientation and origin of $\{a\}$, so the point it represents in world coordinates is a weighted sum of the basis vectors of $\{a\}$ plus the origin vector of $\{a\}$:

$$\begin{aligned} \mathbf{p}_w &= p_{a,x}\mathbf{R}_{wa}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + p_{a,y}\mathbf{R}_{wa}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + p_{a,z}\mathbf{R}_{wa}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \mathbf{t}_{wa} \\ &= \mathbf{R}_{wa}\mathbf{p}_a + \mathbf{t}_{wa}. \end{aligned}$$

The *pose* (position and orientation) of an object is represented by a frame attached

[2] That is, $\mathbf{p}_b$ could refer to $\left(p_{b,x}, p_{b,y}, p_{b,z}\right)^{\mathrm{T}}$ or $\left(p_{b,x}, p_{b,y}, p_{b,z}, 1\right)^{\mathrm{T}}$ depending on the context.

to it, so the pose of an object $\{b\}$ in a frame $\{a\}$ is $\mathbf{T}_{ab}$. Transformation matrices can also change the frame in which a frame is represented, e.g.,

$$\mathbf{T}_{ac} = \mathbf{T}_{ab}\mathbf{T}_{bc}, \tag{2.11}$$

or represent the motion of a point or frame. For more background on rigid-body motions, see Lynch and Park [2017].

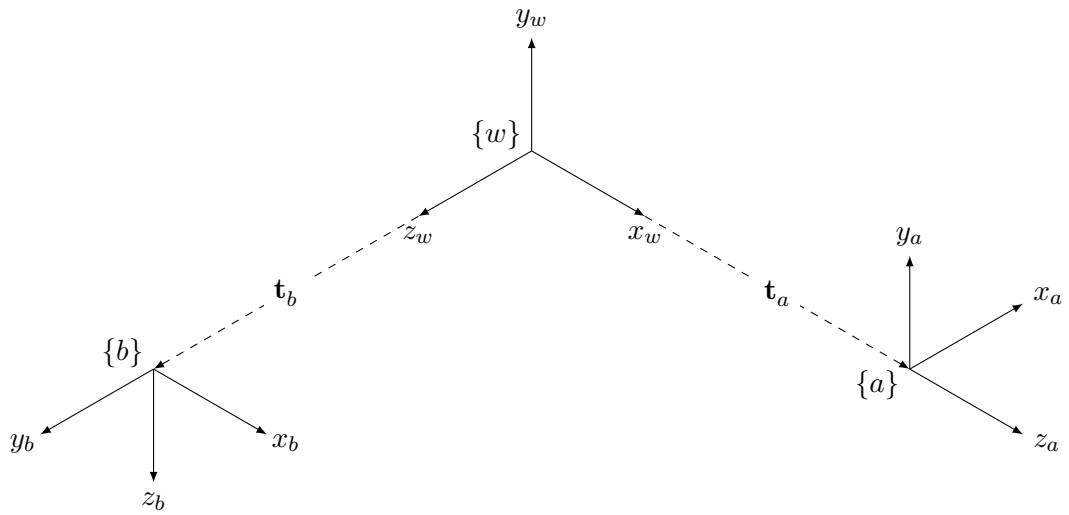


Figure 2.1 A diagram showing the relationship between the world frame $\{w\}$ and the two frames $\{a\}$ and $\{b\}$ used in Section 2.3.1, including their coordinate axes (x_w, y_w, z_w) and so on. The translation vectors are $\mathbf{t}_a = (5, 0, 0)^{\mathrm{T}}$ and $\mathbf{t}_b = (0, 0, 5)^{\mathrm{T}}$.

2.3.1 Example

Let $\{a\}$ have its origin at $(5, 0, 0)^{\mathrm{T}}$ and its orientation rotated 90° about the y-axis with respect to $\{w\}$. Let $\{b\}$ have its origin at $(0, 0, 5)^{\mathrm{T}}$ and its orientation rotated 90° about the x-axis with respect to $\{w\}$. These coordinate frames are shown in Figure 2.1. The transformations representing $\{a\}$ and $\{b\}$ are

$$\mathbf{T}_a = \mathbf{T}_{wa} = \begin{bmatrix} \mathbf{R}_y(90°) & (5, 0, 0)^{\mathrm{T}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.12}$$

and

$$\mathbf{T}_b = \mathbf{T}_{wb} = \begin{bmatrix} \mathbf{R}_x(90°) & (0, 0, 5)^{\mathrm{T}} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.13}$$

$\mathbf{T}_{ab}$ can be calculated from (2.12) and (2.13):

$$\begin{aligned}\mathbf{T}_{ab} &= \mathbf{T}_a^{-1}\mathbf{T}_b \\ &= \mathbf{T}_{wa}^{-1}\mathbf{T}_{wb} \\ &= \mathbf{T}_{aw}\mathbf{T}_{wb} \\ &= \begin{bmatrix} \mathbf{R}_y(-90°)\,\mathbf{R}_x(90°) & (-5,0,-5)^{\mathrm{T}} \\ \mathbf{0} & 1 \end{bmatrix}.\end{aligned} \tag{2.14}$$

Consider a point $\mathbf{p} = (2, 0, 2)$, as shown in Figure 2.2. Pre-multiplication by $\mathbf{T}_a^{-1}$ and $\mathbf{T}_b^{-1}$ gives its coordinates in $\{a\}$ and $\{b\}$,

$$\begin{aligned}\mathbf{p}_a &= \mathbf{T}_{aw}\mathbf{p}_w \\ &= \mathbf{T}_{wa}^{-1}\mathbf{p} \\ &= \mathbf{T}_a^{-1}\mathbf{p} \\ &= (-2, 0, -3)^{\mathrm{T}}\end{aligned} \tag{2.15}$$

and

$$\begin{aligned}\mathbf{p}_b &= \mathbf{T}_b^{-1}\mathbf{p} \\ &= (2, -3, 0)^{\mathrm{T}}.\end{aligned} \tag{2.16}$$

This result is shown in Figure 2.2.

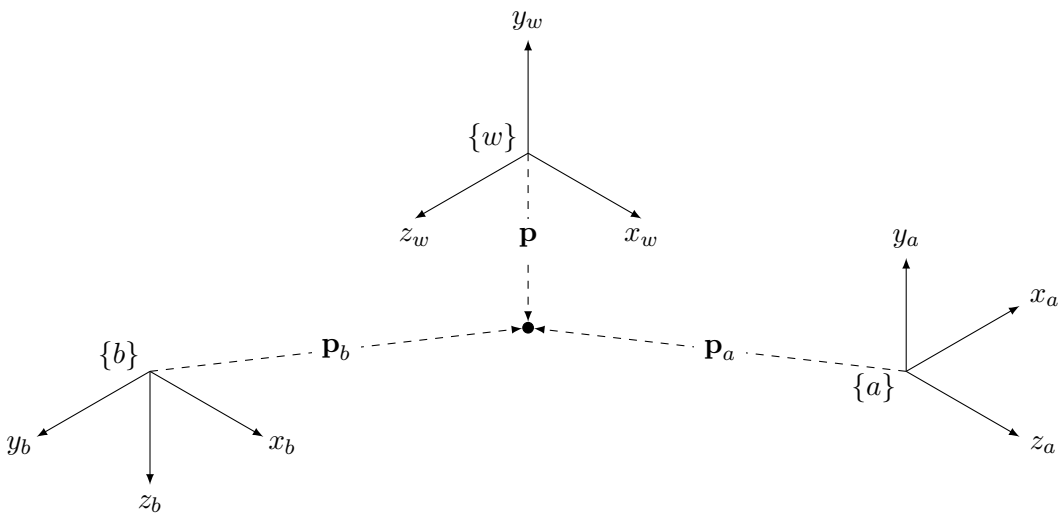


Figure 2.2 A diagram showing the point $\mathbf{p} = (2, 0, 2)^{\mathrm{T}}$ used in Section 2.3.1 and its representations in $\{a\}$ and $\{b\}$, $\mathbf{p}_a = (-2, 0, -3)^{\mathrm{T}}$ and $\mathbf{p}_b = (2, -3, 0)^{\mathrm{T}}$.

2.4 PROJECTIVE GEOMETRY

Historically, the *pinhole camera* (or *camera obscura*) is the predecessor to the modern camera: a box or darkened room with a small hole in an outside wall such that an inverted image is projected onto the opposite wall or a screen [Klette 2014], as shown in Figure 2.3. In projective geometry, an ideal pinhole camera is a point-sized aperture (the *focal point*) through which a scene is projected onto a plane (the *image plane*). Here, the camera frame $\{c\}$ has its origin at the focal point, its z-axis pointing out toward the scene (along the camera's *optical axis* or *principal axis*), its x-axis pointing right, and its y-axis pointing down, as shown in Figure 2.4. The image plane is parallel to the xy-plane at a distance f^* (the camera's *focal length*) back along the optical axis from the focal point. The image frame $\{i\}$ is aligned with $\{c\}$ but has its origin at the point where the optical axis intersects the image plane (the *principal point*).

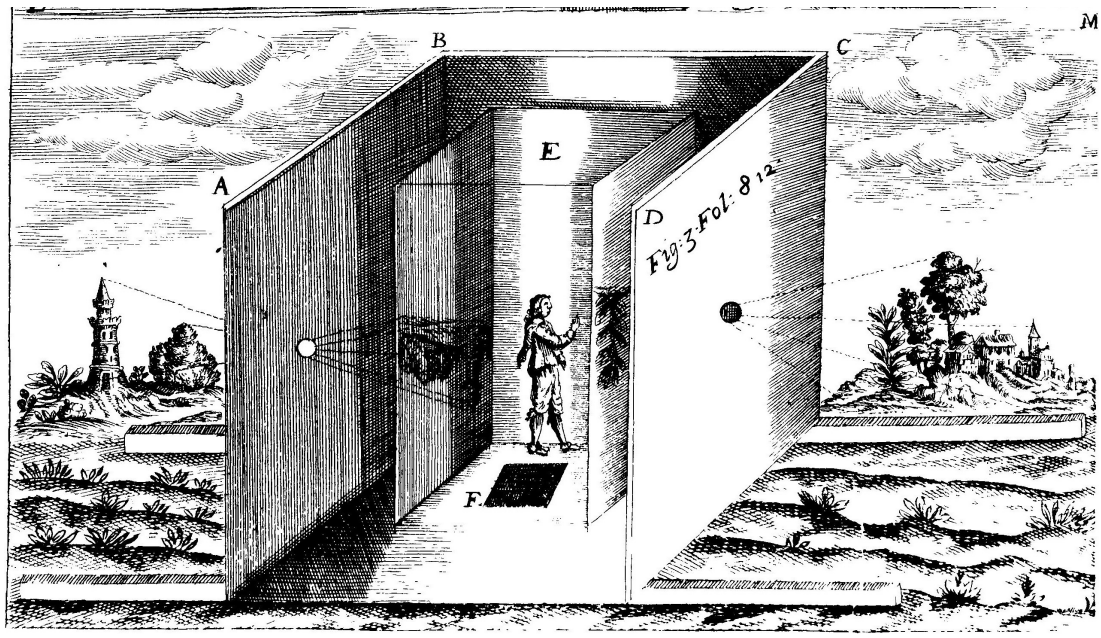


Figure 2.3 A seventeenth-century drawing of a camera obscura, showing outside objects projecting through small holes in the outside wall onto thin screens [Kircher 1646].

When viewed through the camera, a point $\mathbf{p}_c$ in the camera frame projects to the point where a line from $\mathbf{p}_c$ through the focal point intersects the image plane,

$$\mathbf{p}_i = \frac{-f^*}{p_{c,z}} \begin{pmatrix} p_{c,x} \\ p_{c,y} \end{pmatrix}. \tag{2.17}$$

This represents the mirrored image produced by a real pinhole camera, as shown in Figure 2.5. For mathematical convenience, the image plane is often moved in front of

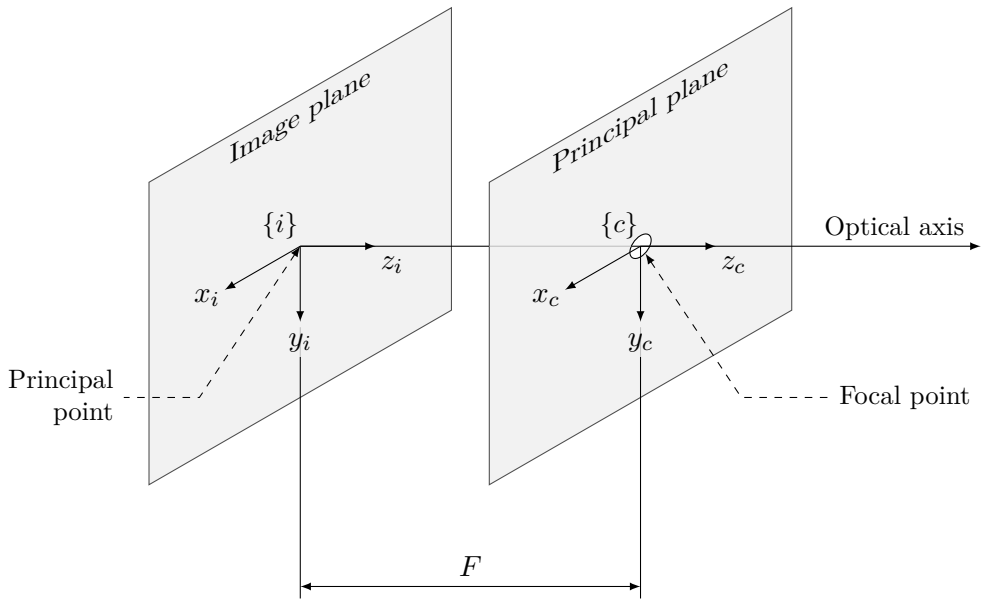


Figure 2.4 A diagram of a pinhole camera showing the image plane, principal point, image frame $\{i\}$, principal plane, focal point (pinhole), camera frame $\{c\}$, optical axis (principal axis), and focal length f^*.

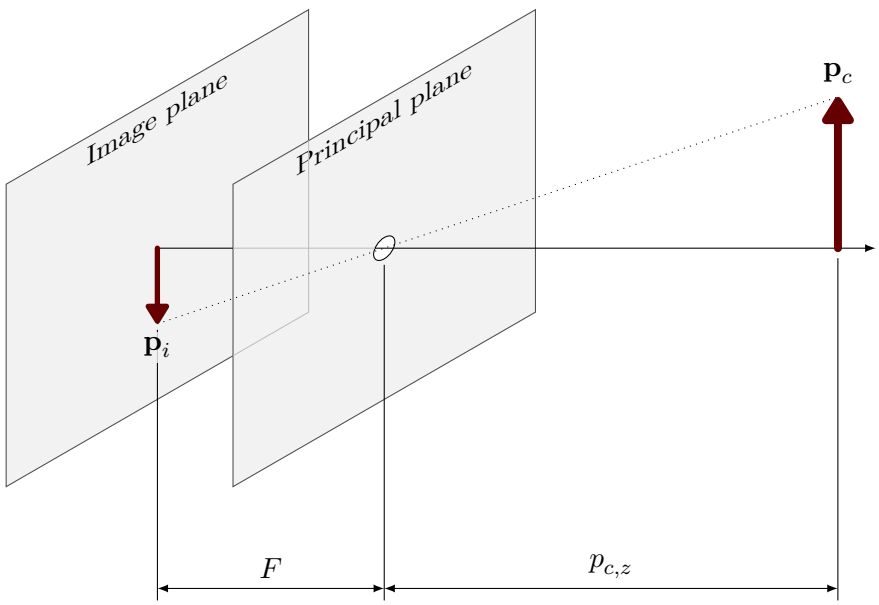


Figure 2.5 A diagram of a pinhole camera showing how a ray passing through the focal point and the point $\mathbf{p}_c$ intersects the image plane at $\mathbf{p}_i$. In this case, $\mathbf{p}_i = -f^*/p_{c,z}\left(p_{c,x}, p_{c,y}\right)^{\mathrm{T}}$.

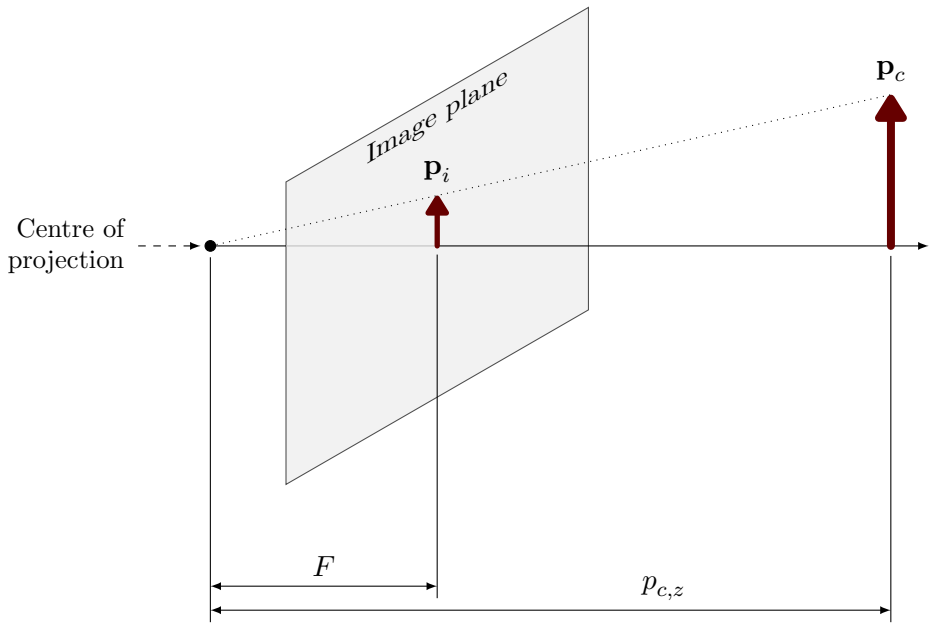


Figure 2.6 A diagram of a pinhole camera showing how a ray passing through the centre of projection and the point $\mathbf{p}_c$ intersects the frontal image plane at $\mathbf{p}_i$. In this case, $\mathbf{p}_i = f^*/p_{c,z}\left(p_{c,x}, p_{c,y}\right)^{\mathrm{T}}$.

the focal point[3] as shown in Figure 2.6, producing an unmirrored image where points project as

$$\mathbf{p}_i = \frac{f^*}{p_{c,z}} \begin{pmatrix} p_{c,x} \\ p_{c,y} \end{pmatrix}. \tag{2.18}$$

Analogously to (2.9), the projection operation can be expressed in homogeneous coordinates as

$$\begin{pmatrix} \mathbf{p}_i \\ 1 \end{pmatrix} p_{c,z} = \begin{bmatrix} f^* & 0 & 0 & 0 \\ 0 & f^* & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} \mathbf{p}_c \\ 1 \end{pmatrix}. \tag{2.19}$$

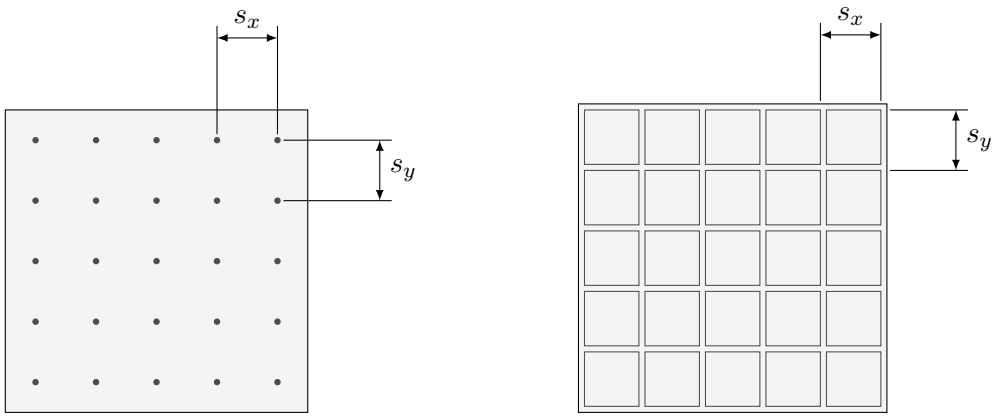
Converting the result of (2.19) back into Cartesian coordinates (i.e., dividing by $p_{c,z}$ and dropping the third component) gives $\mathbf{p}_i$.

An ideal digital camera samples the image on a pinhole camera's image plane at a grid of points (as shown in Figure 2.7a), resulting in a digital image which is a matrix of intensity values (as shown in Figure 2.8). A pixel coordinate $\mathbf{u} = (u, v)$ in the resulting image is related to a point $\mathbf{p}_i$ on the image plane by

$$\mathbf{u} = \begin{pmatrix} p_{i,x}/s_x \\ p_{i,y}/s_y \end{pmatrix} + \mathbf{u}_0, \tag{2.20}$$

where $\mathbf{s} = (s_x, s_y)$ is the distance between sampling points and $\mathbf{u}_0 = (u_0, v_0)$ is the

[3] The focal point in this system is referred to as the *centre of projection* or *optical centre*. The image plane is sometimes referred to as the *frontal image plane* or *virtual image plane*.



(a) A diagram of an ideal image sensor showing the sampling points and sampling point spacing.

(b) A diagram of a more realistic image sensor, showing the sensor elements (as squares) and the sampling grid size.

Figure 2.7 Diagrams of ideal and more realistic 5×5 image sensors. Commercially available cameras have much higher resolution. Note that (b) has sensor elements slightly smaller than the sampling grid size, so there are gaps between the sensor elements.

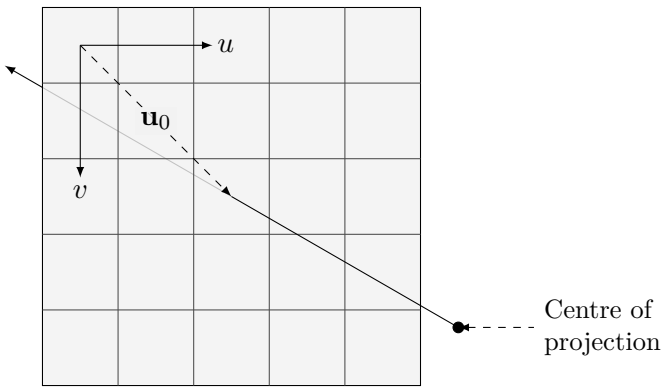


Figure 2.8 A diagram of a 5×5 image showing the pixel coordinate axes (u, v) and the principal point $\mathbf{u}_0$. The origin $(0, 0)$ is at the centre of the top left pixel. In the diagram, the optical axis passes through the centre of the middle pixel, so $c = (2, 2)$.

principal point in pixel coordinates.[4] A real digital camera uses an image sensor consisting of a matrix of tiny light sensors, so the value of each pixel represents the average intensity in a region on the image plane (as shown in Figure 2.7b) rather than the intensity at a point on the image plane, and $\mathbf{s}$ is the size of the sensor elements.[5]

In practice, the physical focal length and sensor element size are generally lumped together as

$$\mathbf{f} = f^* \mathbf{s} \tag{2.21}$$

and referred to as the focal length (in pixels).[6] Combining (2.19) to (2.21) gives

$$\begin{aligned} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} p_{c,z} &= \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} p_{c,x} \\ p_{c,y} \\ p_{c,z} \\ 1 \end{pmatrix}, \\ \begin{pmatrix} \mathbf{u} \\ 1 \end{pmatrix} p_{c,z} &= \mathbf{K} \left[\mathbf{I} \mid \mathbf{0} \right] \begin{pmatrix} \mathbf{p}_c \\ 1 \end{pmatrix}, \end{aligned} \tag{2.22}$$

where $\mathbf{u} = (u, v)^{\mathrm{T}}$ is the location of $\mathbf{p}_c$ in pixel coordinates and $\mathbf{K}$ is called the *intrinsic matrix* or *camera calibration matrix*.[7] This operation (projecting a point in the camera frame into pixel coordinates) is notated

$$\mathbf{u} = \mathrm{proj}\left(\mathbf{p}_c\right). \tag{2.23}$$

In the rest of this thesis, cameras are assumed to have square pixels and zero principal point, so $f_x = f_y = f$ and $\mathbf{u} = f\mathbf{p}_i$.

Combining (2.9) and (2.23) gives an expression for projecting a point in an arbitrary coordinate frame $\{a\}$ into pixel coordinates,

$$\begin{pmatrix} \mathbf{u} \\ 1 \end{pmatrix} p_{c,z} = \mathbf{K} \left[\mathbf{I} \mid \mathbf{0} \right] \mathbf{T}_{ca} \begin{pmatrix} \mathbf{p}_a \\ 1 \end{pmatrix}, \tag{2.24}$$

or simply

$$\mathbf{u} = \mathrm{proj}\left(\mathbf{T}_{ca}\mathbf{p}_a\right). \tag{2.25}$$

[4] Note that some texts incorrectly give (2.20) as $\mathbf{u} = \mathbf{s} \cdot \mathbf{p}_i + \mathbf{u}_0$.

[5] A real camera may have sensor elements whose active areas do not reach all the way to the edge (so $\mathbf{s}$ is the sample grid size rather than the sensor element size), but this is generally ignored.

[6] The physical focal length and sensor element size are difficult to measure, while the focal length in pixels can be readily determined through intrinsic calibration. In some situations, calibration results can be approximately verified using manufacturer specifications (for example when using a fixed-focal-length lens with a machine vision camera).

[7] Many authors include a skew term as K_{12}; it is omitted here as it is irrelevant for most cameras.

The matrix $\mathbf{P}_c = \mathbf{K}\,[\mathbf{I} \mid \mathbf{0}]\,\mathbf{T}_{cw}$ is sometimes called the *projection matrix*.[8,9] For more background on cameras and projective geometry, see chapter 6 of Klette [2014] or chapter 2 of Szeliski [2011].

2.5 POSE ESTIMATION USING 3D-TO-2D CORRESPONDENCES

An image of a known object can be used to estimate its pose (in the camera frame) by matching each of a set of N known 3D points on the object,

$$\{\mathbf{p}_o\} = \left\{\mathbf{p}_o^{(1)}, \mathbf{p}_o^{(2)}, \ldots, \mathbf{p}_o^{(N)}\right\},$$

to estimates of their locations in the image,

$$\{\mathbf{p}_i\} = \left\{\mathbf{p}_i^{(1)}, \mathbf{p}_i^{(2)}, \ldots, \mathbf{p}_i^{(N)}\right\}.$$

For example, when the object is a checkerboard or fiducial marker (see Chapters 3 and 4), the object points are the 3D positions of the corners (in the object frame) and the image points are their locations in image coordinates as estimated by a corner detection algorithm [e.g., Duda and Frese 2018, Harris and Stephens 1988]. The same formulation is used in VO, with the object points being the estimated 3D position of each feature in the previous image, the image points being the estimated locations of the corresponding features in the current image, and the object frame being the pose of the camera when the previous image was taken [Scaramuzza and Fraundorfer 2011].

The objective is to find a transformation from the camera frame to the object frame, $\mathbf{T}_{co}$, which minimises the image reprojection error:

$$\hat{\mathbf{T}}_{co} = \underset{\mathbf{T}_{co} \in SE(3)}{\arg\min} \sum_k \left\| \mathbf{p}_i^{(k)} - \operatorname{proj}\left(\mathbf{T}_{co}\,\mathbf{p}_o^{(k)}\right) \right\|. \tag{2.28}$$

This is known as the *perspective-n-point problem* (PnP), and it has been studied extensively [Fischler and Bolles 1981, Lepetit et al. 2009, Lu et al. 2000]; Terzakis and Lourakis [2020] present a good review. Estimating pose using three point correspondences (the minimum required, using a fourth point to disambiguate multiple solutions)

[8] $\mathbf{P}$ and $\mathbf{K}$ are both sometimes called the camera matrix.

[9] Defining the projection matrix as

$$\mathbf{P}_c = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{T}_{cw} \tag{2.26}$$

instead gives

$$\begin{pmatrix} \mathbf{u} \\ 1 \\ d \end{pmatrix} p_{c,z} = \mathbf{P}_c \mathbf{p}, \tag{2.27}$$

where $d = 1/p_{c,z}$ is the *disparity* (inverse depth).

is the *perspective-three-point problem* (P3P) [Ke and Roumeliotis 2017, Kneip et al. 2011b]. Pose can also be estimated using 2D-to-2D or 3D-to-3D correspondences. For more detail on any of these pose estimation methods, see Huang and Netravali's [1994] review, Scaramuzza and Fraundorfer's [2011] tutorial, or chapters 6–7 of Szeliski [2011].

Chapter 3

HIGH-PRECISION FIDUCIAL MARKER SYSTEM

In this chapter, which is based on published work [Edwards et al. 2016], a fiducial marker system that uses a smooth pattern to enable high-precision pose estimation is presented. The performance of this marker is evaluated in the context of existing work, and its strengths and weaknesses are examined.

3.1 INTRODUCTION

VO algorithms are typically evaluated indoors using ground truth from motion capture systems [Fraundorfer et al. 2012, Geneva et al. 2019, Kneip et al. 2011a, Qin et al. 2018, Rosinol et al. 2020, Shen et al. 2013, 2015]. However, this requires a controlled environment and cannot extend to large-scale outdoor operation. GPS/INS ground truth can be used in some outdoor environments. This is not always practical due to the high size, weight and power requirements of high-end equipment, and requires open sky to perform well. An alternative solution is to prepare the environment with *fiducial markers*: specially designed objects which, when visible to a camera, can be used to estimate the camera's pose relative to the object. The fiducial markers commonly used in the field of augmented reality incorporate unique identifiers, enabling localisation within an environment when visible. However, they only produce a precise pose estimate when close to the camera. In this chapter, a new kind of fiducial marker with increased pose estimation precision is proposed. A system that performs outdoor localisation using a combination of traditional markers (which provide a low-precision global pose estimate, since they are distinguishable) and the proposed markers (which would provide a high-precision relative pose estimate) would be a convenient solution to the problem of recording ground truth for evaluating VO algorithms in forests. An implementation with real-time performance would also have other applications, such as general robotic localisation and low-cost motion capture.

The rest of this chapter is organised as follows. Related work is described in Section 3.2. Section 3.3 presents preliminary work done while exploring the ideas of this chapter, and Section 3.4 introduces relevant background information. Section 3.5 out-

lines the fiducial marker system and experiments performed in this chapter. Section 3.6 presents the experimental results and discusses the limitations of the algorithm, and conclusions are drawn in Section 3.7.

3.2 RELATED WORK

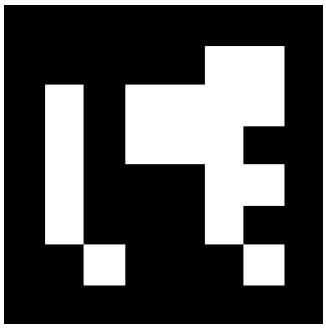
Square planar fiducial markers, like the ones shown in Figures 3.1a to 3.1c, have been used in the field of augmented reality for many years. ARToolkit [Kato and Billinghurst 1999] is perhaps the most well-known, as it was one of the first to be released as open-source software. The basic steps of the algorithm are image thresholding, finding quadrilateral-shaped regions, using the corners to calculate a homography, and performing correlation-based template matching on the marker image to identify the marker within a pre-configured library of markers. The template matching system has many downsides (complicated training procedure, high false positive/negative and inter-marker confusion rates which get worse as the library size increases, computation time increases with library size, etc), so various approaches were proposed which replaced it with barcode-like patterns incorporating error correction codes, notably ARTag [Fiala 2005] and ARToolkitPlus [Wagner and Schmalstieg 2007].

Recent research has focused on robustness. For ArUco, Garrido-Jurado et al. [2014] proposed an algorithm for generating marker dictionaries that maximise inter-marker distance (i.e., the number of bit flips between any two marker patterns in a dictionary). An ArUco marker is shown in Figure 3.1b. The basic algorithm steps are similar but incrementally improved, and the authors also refined the common practice of using multiple markers together by using detected markers in a scene to help identify partially occluded markers. Similarly, Olson [2011] proposed a detection method for AprilTag that is robust to occlusion and a coding system that is robust to rotation (shown in Figure 3.1c).

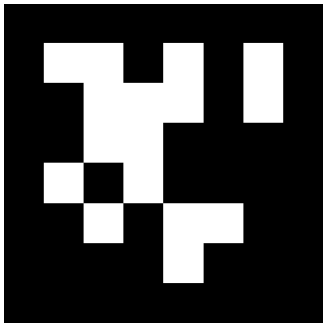
RUNE-Tag [Bergamasco et al. 2011] is a high-precision fiducial marker using concentric rings of dots, as shown in Figure 3.1d. It has excellent occlusion resistance and is resilient to noise, blur and uneven illumination. The authors claim its pose estimation precision in simulation is an order of magnitude better than ARToolkit, but they do not give much detail on their methodology and only appear to consider the angles, ignoring the translation. The section of their paper titled “Behaviour with Real Images” is rather short and largely lists its shortcomings. An open-source implementation of the algorithm was not released until 2017, which may explain why it received little attention.

Fourier Tag [Xu and Dudek 2011] features a radial sinusoid pattern as shown in Figure 3.1e, in which it stores information using phase-shift keying. It uses the circular shape of the marker to calculate its pose, and the authors do not focus on (or even evaluate) precision.

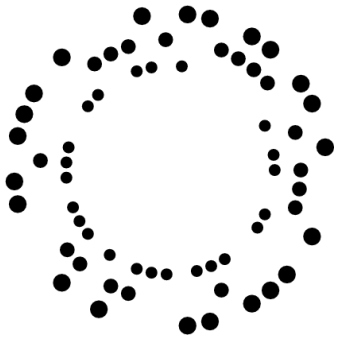
(a) An Artoolkit marker [Kato and Billinghurst 1999].



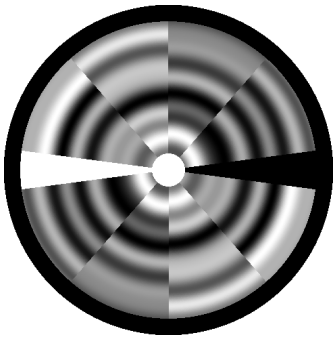
(b) An ArUco marker [Garrido-Jurado et al. 2014].



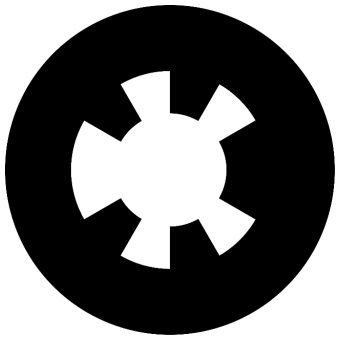
(c) An AprilTag marker [Olson 2011].



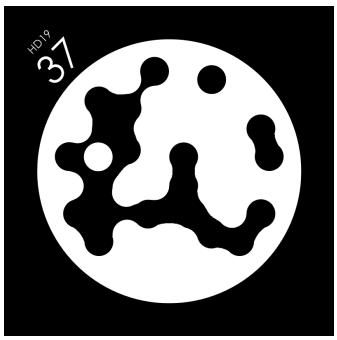
(d) A RUNE-Tag marker [Bergamasco et al. 2011].



(e) A Fourier Tag [Xu and Dudek 2011].



(f) A WhyCode marker [Lightbody et al. 2017].



(g) An STag marker [Benligiray et al. 2019].

Figure 3.1 A selection of other fiducial markers.

WhyCode [Lightbody et al. 2017] uses a circular pattern for fast and precise localisation, with an inner Binary Necklace pattern for identification and rotation estimation, as shown in Figure 3.1f. It was found to be much faster, more robust, and more precise than AprilTag and ARTag, but unfortunately, it was not tested against the much-newer ArUco, although the authors did acknowledge it as being state-of-the-art at the time.

STag markers [Benligiray et al. 2019] have a square outline that is used for marker detection in a similar way to previous square markers, and an inner circular border that is used in a pose refinement step for better estimation stability, as shown in Figure 3.1g. The authors claim it is more precise and robust than ArUco (but a bit slower), and much more robust and almost as precise as RUNE-Tag (and much faster).

After the research presented in this chapter was performed, Hannemose et al. [2019] proposed a camera calibration technique which uses a similar rendering and optimisation process to the one presented here. In their method, a standard checkerboard-based camera calibration technique is used for initial calibration, then the rendering and optimisation process is used to refine the calibration parameters. They convolve the calibration pattern with a Gaussian kernel in order to obtain a smooth, differentiable pattern, and make use of this property by computing the derivatives for the numerical optimisation algorithm.

3.2.1 Sinusoid parameter estimation

Estimating the parameters of a sinusoid signal is a problem that comes up in many fields. Rife and Boorstyn [1974] considered this problem from the then-novel perspective of "data set testing, telephone transmission system testing, radar, and other measurement situations", in which there is a real-valued continuous signal consisting of a single tone,

$$s(t) = b_0 \cos\left(\omega_0 t + \theta_0\right), \tag{3.1}$$

where b_0 is the amplitude, ω_0 is the angular frequency, and θ_0 is the phase offset. The signal is sampled at a constant rate of $1/T$ (where T is the sample period) with the first sample taken at $t = t_0$, and the samples include independent Gaussian noise with zero mean and variance σ^2. They found the CRLB[1] for estimating the amplitude, frequency, and phase of the signal from the samples under various conditions (estimating each with and without knowledge of the others). The simplest form for each is the bound for estimating ω with θ unknown,

$$\mathrm{Var}\left(\hat{\omega}\right) \geq \frac{12\sigma^2}{b_0^2 T^2 N(N^2-1)}, \tag{3.2}$$

[1] The CRLB gives the variance of the lowest-variance unbiased estimator. For more, see Section 4.2.1.

and the bound for estimating θ with ω known,

$$\mathrm{Var}\left(\hat{\theta}\right) \geq \frac{\sigma^2}{b_0^2 N}, \tag{3.3}$$

where N is the number of samples. The bound for estimating b is the same whether or not the other parameters are known,

$$\mathrm{Var}\left(\hat{b}\right) \geq \frac{\sigma^2}{N}. \tag{3.4}$$

3.2.2 High-precision one-dimensional markers

Roberts et al. [2015] proposed the concept of a "structured marker" which uses a one-dimensional marker with a sinusoid pattern for distance estimation. They required the marker to be parallel to the camera and in the centre of its frame. Their algorithm is relatively simple: they sampled a row of pixels along the centre of the image, applied a Gaussian window, took the Fourier transform, used the peak as an estimate of the spatial frequency of the marker, then used the ratio of that frequency to the (known) spatial frequency of the marker in world units (along with the camera intrinsics) to calculate the distance. To evaluate the performance of their method, they compared their results (from simulation) to the CRLB, as shown in Figure 3.2.

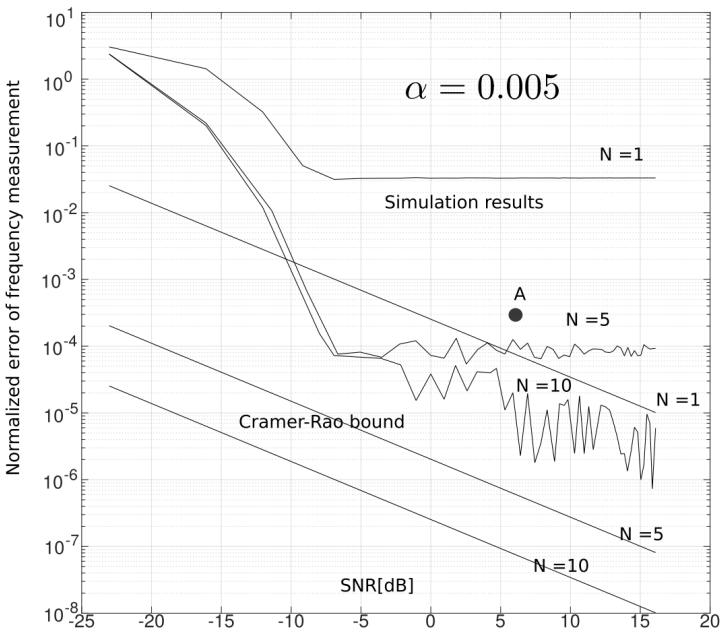


Figure 3.2 The best simulation results from Roberts et al. [2015], for signals of length 10, 50, and 100 (number of periods $N = 1$, 5, and 10). The exponential drop-off rate for the Gaussian window function is $\alpha = 0.005$. The point labelled "A" marks the performance of the technique described by Grishin [2010], which was the inspiration for Roberts et al. [2015]. Note that the rapid increase in error at low SNR is a well-known effect in non-linear frequency estimators [Rao 2013]. Figure © 2015 IEEE.

3.3 PRELIMINARY WORK

Before beginning design of the novel fiducial marker presented in this chapter, several preliminary steps were taken: a localization system was built and tested with existing fiducial markers, the noise distribution of the camera to be used was investigated, and improvements were made to the method proposed by Roberts et al. [2015].

3.3.1 Fiducial localization prototype

A prototype of a fiducial marker-based localisation system was implemented using ArUco markers [Garrido-Jurado et al. 2014], the g2o graph optimisation library [Kümmerle et al. 2011], and Robot Operating System (ROS) [Quigley et al. 2009]. The system consists of three nodes. The first node subscribes to camera messages, detects markers in each camera frame, and publishes the list of detected markers. The second node is used to set up a map of the markers in an area; it builds up a pose graph with a vertex for each camera position and each marker and an edge for each observation of a marker in a camera frame, and uses g2o to optimise the graph. Once the graph contains loops (e.g., markers 1 and 2 have been observed together, markers 2 and 3 have been observed together, and markers 1 and 3 have been observed together), the graph optimisation process is constrained enough that it outperforms a simple average. The graph is saved and used by the third node, which simply uses the map and any observed markers to estimate the camera position. Initially, a constant-valued and isotropic covariance model was used for each observation; investigation into the noise properties of fiducial markers (detailed in Section 3.4.2) is what inspired the high-precision marker approach. If an appropriate (i.e., anisotropic and distance-dependent) covariance model was used, the graph optimisation would correctly take into account the decreased precision of distant markers.

3.3.2 Image sensor noise distribution

An experiment was performed to investigate the noise distribution in images from a Point Grey (now Flir) Bumblebee2 BB2-08S2C global shutter machine vision camera. A marker was set up so that it was approximately in the centre of the image frame of the left head of the camera, first at about 2 m then at about 4 m from the camera. No special lighting was used, just overhead fluorescent lights. Over 400 8-bit colour images with $1024\,\text{px} \times 768\,\text{px}$ resolution were captured in each position, with the first of each shown in Figure 3.3. For analysis, each frame was converted to greyscale, corrected for lens distortion, and cropped to only the area containing the marker (as shown in Figure 3.4). Each array of cropped frames can then be considered as a time-varying signal. The standard deviation σ of each pixel intensity ($I \in [0, 255]$) gives a measure of the noise intensity for that pixel. The standard deviations range from about 0.4 to 1.3, with a

(a) The first image taken at 2 m from the camera.

(b) The first image taken at 4 m from the camera.

Figure 3.3 The first image in each of the two sets of images captured for the camera noise analysis experiment.

mean of 0.8. Plotting the standard deviations as an image shows that there is a clear link between intensity value and standard deviation (see Figure 3.5). Assuming the image noise is predominately shot noise would lead to a square root model; this fits well, as shown in Figure 3.6.[2] One side-effect is that the signal-to-noise ratio of a pixel is dependent on its intensity. Since the marker pattern is fixed and known, brighter areas could be given more weight in the optimisation process. Whether this would have an effect on estimation precision requires further investigation.

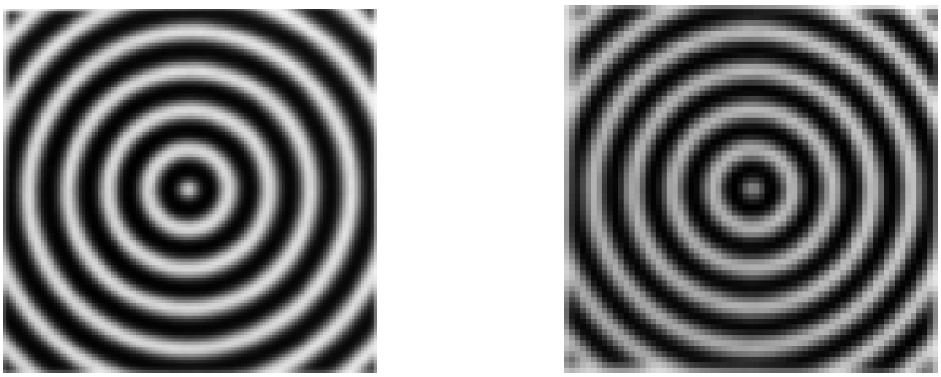
3.3.3 High-precision one-dimensional markers

A few simple changes can be made to the method proposed by Roberts et al. [2015] which drastically improve its performance. Zero-padding the signal to length 2^{14} before taking the fast Fourier transform (FFT) and using quadratic interpolation when measuring the peak (to reduce spectral leakage) brings the error very close to the CRLB when the signal-to-noise ratio (SNR) is greater than 0 dB. Furthermore, using the Levenberg-Marquardt algorithm [Levenberg 1944, Marquardt 1963] to fit a signal directly achieves errors close to the CRLB for the full SNR range that was considered. These results are shown in Figure 3.7.

3.4 BACKGROUND

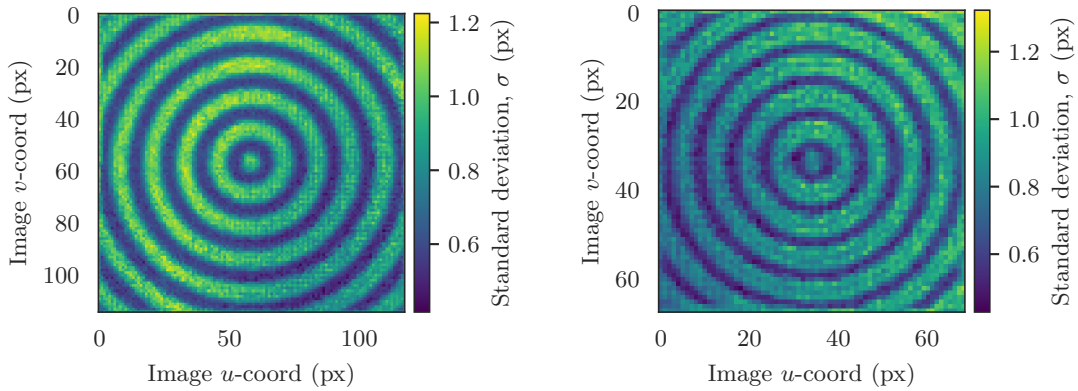
This section gives an overview of the geometric distortion and noise processes involved in a fiducial marker system and presents a sensitivity analysis of pose estimation.

[2] This is examined in greater detail (but with different equipment) in Section 6.3.1.



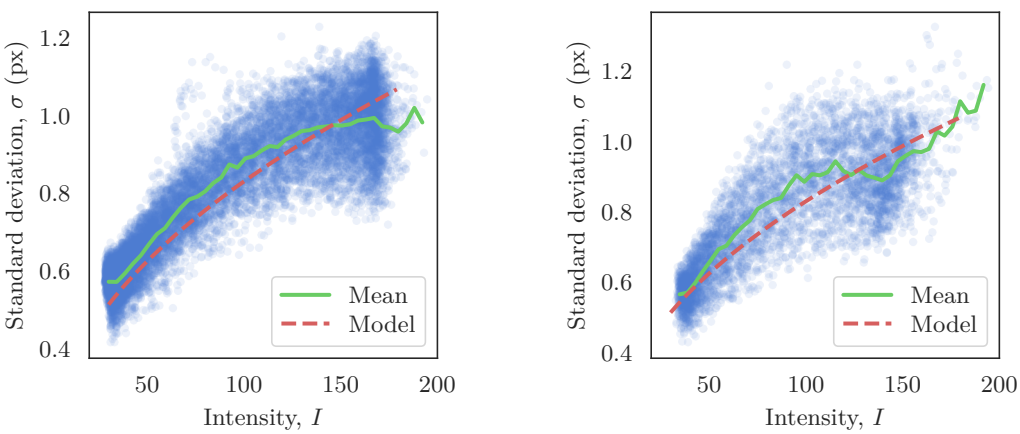
(a) 2 m (b) 4 m

Figure 3.4 The images from Figure 3.3 after converting to greyscale, correcting for lens distortion, and cropping to only the area containing the marker.



(a) 2 m (b) 4 m

Figure 3.5 The standard deviations of the intensity of each pixel in the two sets of images.



(a) 2 m (b) 4 m

Figure 3.6 Scatter plots of the standard deviations of the intensity of each pixel in the two sets of images. The mean is calculated by sorting the intensities into 40 bins and taking the mean of the standard deviations in each bin. The model is $\sigma = 0.13 + 0.07\sqrt{I}$, which was fitted using least squares to illustrate that a square-root-shaped model fits the data.

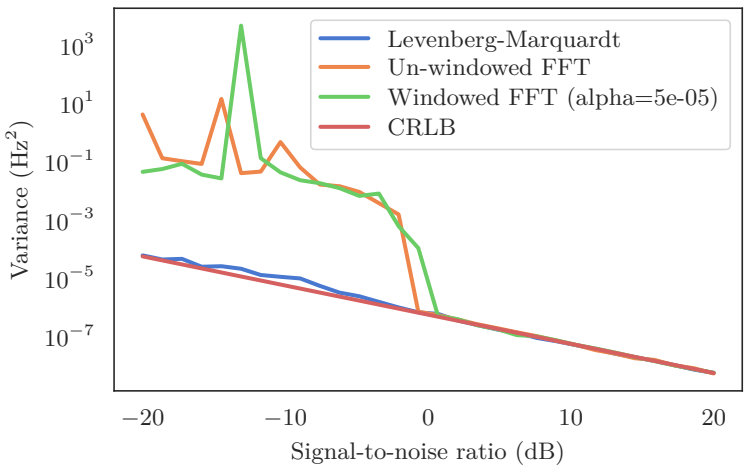


Figure 3.7 Simulation results for the improved methods presented in Section 3.3.3 when applied to a signal of length 100 (number of periods N=10), for comparison with Figure 3.2. The three methods shown are a least-squares fit of frequency, phase, and amplitude, Fourier frequency estimation with zero-padding to length 2^{14} and quadratic peak interpolation, and the same Fourier method using a Gabor window with $\alpha = 0.00005$.

3.4.1 Image distortion and noise

Many processes take place between generating a marker and detecting it in an image, each of which introduces distortion or noise. When a marker is printed, the intensity of each pixel is translated into the density of the pattern of dots on the paper. Since printers only have black ink (ignoring colour printers, since the marker is greyscale), this is the only way to print shades of grey. From sufficient distance, the printed pattern resembles the original pattern, and from up close, the individual dots are visible (as shown in Figure 3.8). When examined carefully, however, there are visible bands in the intensity gradient. Black areas are neither completely black nor particularly uniform (instead, a textured dark grey), and they have specular reflections in some lighting conditions as toner is slightly reflective.

Within the camera, the lens assembly of the camera adds perspective projection (like an ideal pinhole camera) as well as various distortions [Hartley and Zisserman 2004, sections 6.2 and 7.4]. Depending on the exposure, the intensity of the marker pattern is scaled and offset. Gamma correction is often applied [Szeliski 2011, section 2.3]. There may also be blur, depending on the camera's focal length, the distance from the camera to the marker, and any relative motion of the camera and marker. The camera sensor itself adds various types of noise—thermal noise, electronic noise, amplifier noise and quantisation noise [Forne 2007]. Since the sensor elements are rectangular rather than point-like, there is an additional blurring process.

These processes can be divided into geometric effects (perspective projection and lens distortion), noise processes, and transformations of the marker's intensity. Using camera calibration, the geometric effects can be reduced to those of a pinhole cam-

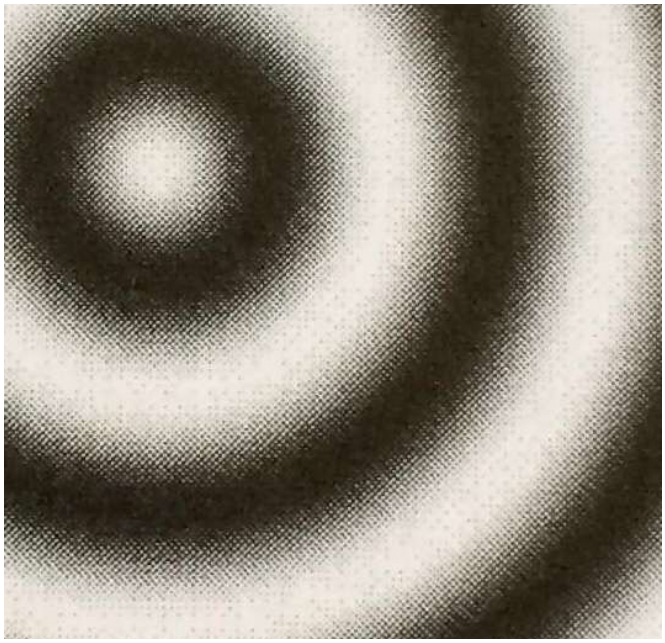
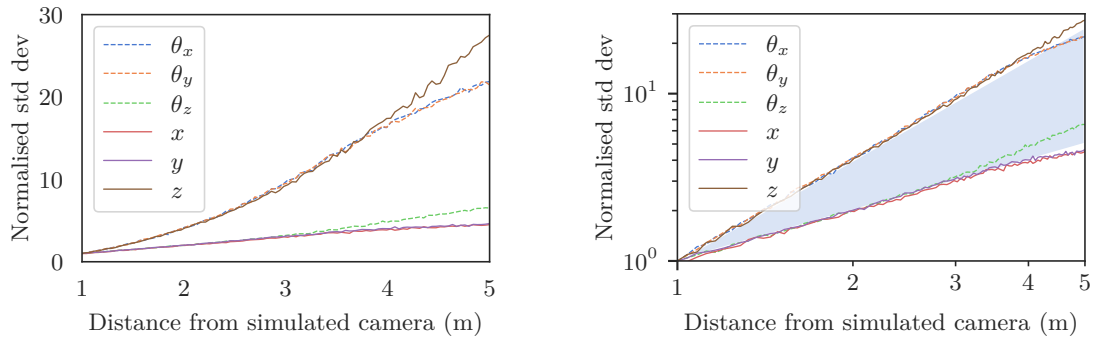


Figure 3.8 A photograph of the printed marker, enlarged to show printing artefacts.

era [Hartley and Zisserman 2004, section 7.4]. The noise processes are modelled well by a Gaussian distribution with intensity-dependent variance (see Section 6.3.1). The intensity transformations could be modelled as a combination of processes, but in this work, they are simply corrected using histogram matching [see Gonzales and Fittes 1977].

3.4.2 Parameter sensitivity

The *precision* of an estimate of a parameter is the inverse of its variance [Gelman et al. 2013, section 2.5]. It is well-known that when estimating the position of an object visually, the estimated distance along the camera's optical axis is less precise than the distances along the two axes perpendicular to the optical axis. The basic mechanism behind pose estimation in most square fiducial markers is to solve the PnP problem using the detected marker corners and their known relative positions (see Section 2.5). To investigate the trend in each parameter's precision with distance, a Monte Carlo simulation of fiducial marker pose estimation was performed. The positions of the corners of a marker at various distances from the camera were projected into image coordinates, then independent and identically distributed (i.i.d.) zero-mean Gaussian noise was added and the `solvePnP` function from the OpenCV library [Bradski 2000] was used to estimate the marker's pose. This was repeated 1000× for each distance, and the standard deviation of each pose parameter was taken as a measure of precision. The results are shown in Figure 3.9. An approximately linear increase with increasing distance is visible in the x- and y-axis translations and the z-axis rotation θ_z, and an



(a) linear axes

(b) log axes, with linear-to-quadratic region shaded

Figure 3.9 The results of a Monte Carlo simulation of fiducial marker pose estimation, showing the standard deviation of each pose parameter as a function of the marker's distance from the camera. Each parameter is normalised such that the first point's standard deviation is one. In (b), the area between linear functions ($\sigma \propto z$) and quadratic functions ($\sigma \propto z^2$) is shaded.

approximately quadratic increase is visible in the x- and y-axis rotations θ_x and θ_z and the z-axis translation.[3]

Consider a featureless square marker held in the centre of an image, parallel to the focal plane. A movement of the marker left or right (or up or down) will produce more visual difference than a movement of the same distance forward or back. Similarly, a rotation of the marker about the camera's optical axis will produce more visual difference than a rotation through the same angle about either of the other axes. Note that although this effect will vary depending on the pose of the marker, the same basic trends still apply. Thus, a high-precision fiducial marker system should aim to improve on the estimation precision for z, θ_x and θ_y. The marker pattern presented here is invariant under z-axis rotation (up to outline), so can not be used to estimate θ_z (beyond the PnP result). Since θ_z is one of the parameters which can be estimated with reasonable precision from the marker corners alone, this rotational invariance reduces the dimensionality of the optimisation problem without sacrificing much precision.

3.4.3 Pattern design

The proposed marker uses a radial sinusoid pattern, as shown in Figure 3.10. Every pixel of the pattern is used to estimate the marker's pose, in contrast to traditional markers where only corners or edges are used. When considering the design of this pattern, an analogy can be made to various aspects of sinusoid parameter estimation. For a marker that is parallel to the camera, estimating its distance to the camera is a similar problem to estimating the frequency of a sinusoid signal as in Section 3.2.2, the CRLB for which, (3.2), depends largely on SNR and number of samples. Estimating the marker's horizontal and vertical position is similar to estimating the time delay of

[3] See also Section 4.3.2, where this relationship is arrived at more rigorously.

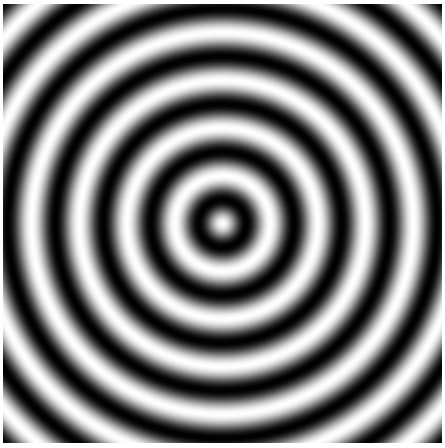


Figure 3.10 A section of the radial sinusoid pattern used in the proposed fiducial marker. In the final marker, the number of periods in the pattern was limited to produce a solid outline that could be detected reliably, as discussed in Section 3.5.3.

a wave. The time delay is related to frequency and phase by

$$\theta = \omega\tau. \tag{3.5}$$

When finding the time delay there is an ambiguity due to phase unwrapping. Assuming the ambiguity is resolved, the time delay is

$$\tau = \frac{\theta}{\omega}. \tag{3.6}$$

Using (3.3) to find the CRLB for $\hat{\tau}$ (with known frequency),

$$\mathrm{Var}\left(\hat{\tau}\right) \geq \frac{\mathrm{Var}\left(\hat{\theta}\right)}{\omega} \tag{3.7}$$

$$= \frac{\sigma^2}{\omega b_0^2 N}, \tag{3.8}$$

it is clear that the variance of this estimate decreases with increasing frequency. This is a useful observation for the design of the marker: increasing the spatial frequency of the pattern ought to improve the estimation precision for at least some pose parameters in some circumstances, assuming the spatial frequency is still sufficiently low as to be sampled adequately. This is also analogous to sonar and radar, where higher-frequency signals produce more precise range estimates [Rihaczek 1996].

3.5 METHOD

The image is initially rectified to remove lens distortion. Then, the marker outline is detected with an algorithm based on the one proposed by Garrido-Jurado et al. [2014]. Once a marker has been detected in the image, the corner locations are used to

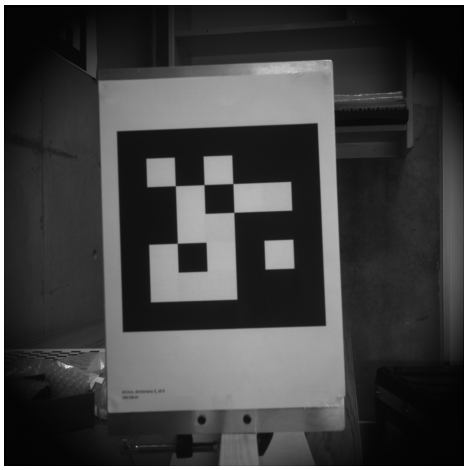
Figure 3.11 An image of the proposed marker with 9 periods of the sinusoid pattern mounted on the rail at 1200 mm from the camera, as used in the experiment.

produce a coarse pose estimate $(x, y, z, \theta_x, \theta_y, \theta_z)$, where (x, y, z) is the position of the marker centre in camera coordinates, θ_x and θ_y are the rotations of the marker about its x- and y-axis, and θ_z is the rotation about its z-axis (which is perpendicular to the marker surface). Histogram matching is used to match the intensity distribution of the camera image to that of the rendered marker. Then, the coarse pose estimate is refined through a series of non-linear optimisations. Each time, a subset of the parameters is refined using Nelder-Mead optimisation [Nelder and Mead 1965]. The optimisation process compares the image from the camera to a rendering of the marker at the current estimated pose, as described in Sections 3.5.3 and 3.5.4. The resulting optimised pose estimate is significantly more precise than the coarse pose estimate.

3.5.1 Data collection

The three markers used were the proposed marker with nine periods of the sinusoid pattern (shown in Figure 3.11), the proposed marker with five periods (shown in Fig-

(a) An image of the proposed marker with 5 periods of the sinusoid pattern mounted on the rail at 600 mm from the camera.



(b) An image of an ArUco marker mounted on the rail at 600 mm from the camera.

Figure 3.12 Rectified images of the other markers used in the experiment.

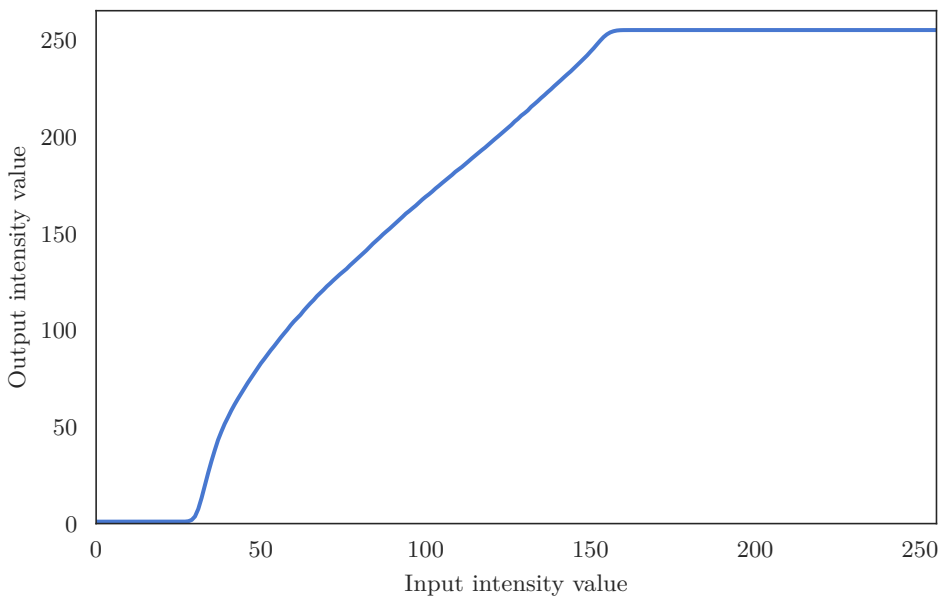


Figure 3.13 The estimated intensity mapping that transforms the intensity distribution of Figure 3.14a to match Figure 3.14b.

ure 3.12a), and an ArUco marker [Garrido-Jurado et al. 2014] (shown in Figure 3.12b). Each marker was laser-printed on plain A4 paper and mounted in turn to a vertical plate attached to a guide rail. A Point Grey (now Flir) Grasshopper3 GS3-U3-41C6NIR-C global shutter machine vision camera with a 16 mm Sony lens set to 1 m focus and 8 mm aperture was arranged such that the marker was 600 mm ± 50 mm from the lens and in the centre of the image, and the rail ran approximately parallel to the camera's optical axis. The camera was set to 15 fps, 8-bit monochrome format, zero gain, and the max shutter time of 67 ms. The exposure was left on automatic mode. The marker was then moved along the rail in 300 mm ± 1 mm steps (up to 2400 mm), and 150 images were taken at each position. This was repeated for each marker, then the whole experiment was repeated without moving the equipment, for a total of 2100 images of each marker. The position of the marker in each image was calculated, and the standard deviation for each position was calculated for each parameter and used as a measure of estimation precision.

3.5.2 Histogram matching

In order for the sum of squared intensity differences to be a reliable objective function, the image needs to have approximately the same intensity distribution as the rendering. This is ensured using histogram matching. The histogram matching algorithm is as follows [Gonzales and Fittes 1977]. Take a template image and a sample image. Let $\mathbf{I}_t$ be a list of the intensity values of each pixel of the template image, and let $\mathbf{I}_s$ be the same for the sample image. Let the empirical cumulative distribution function $F_t(i)$ be defined as

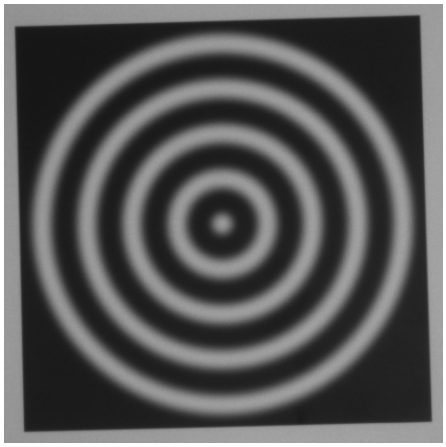
$$F_t(i) = \frac{|i_t \in \mathbf{I}_t : i < i_t|}{|I_t|}, \tag{3.9}$$

and let $F_s(i)$ be the same for the sample image. To perform histogram matching, for each intensity i_s in the sample image, the corresponding intensity i_t is found such that

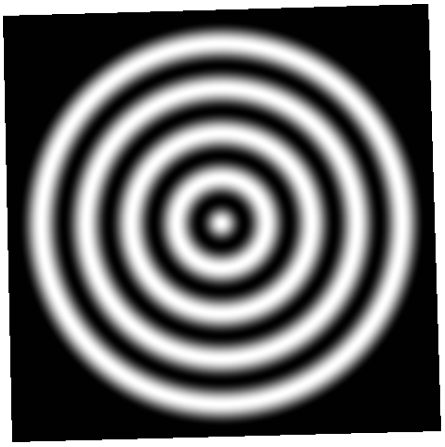
$$i_s \leq i_t \quad \text{iff} \quad F_s(i_s) \leq F_t(i_t). \tag{3.10}$$

These correspondences then form a mapping that, when applied to the intensity of each pixel in the sample image, produces an image with an intensity distribution much like that of the template image.

In this case, the sample image is a section of an image that contains the marker (as shown in Figure 3.14a) and the template image is a marker rendering of the same size using the coarse pose estimate (as shown in Figure 3.14b). The histogram matching process is performed on one image in the dataset and the resulting intensity mapping is re-used for the others, as all the images have similar intensity distributions. Figure 3.14c shows the image in Figure 3.14a after the intensity mapping, which is shown in Figure 3.13, has been applied. Figure 3.15 shows the intensity along corresponding

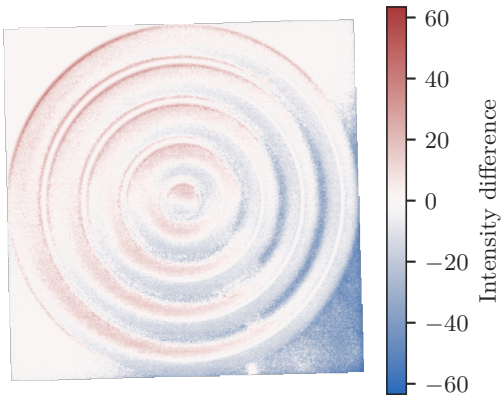


(a) The section of the image in Figure 3.11 containing the marker. The cropped area has been expanded slightly for illustration purposes; in the optimisation process, only the area inside the marker is used.



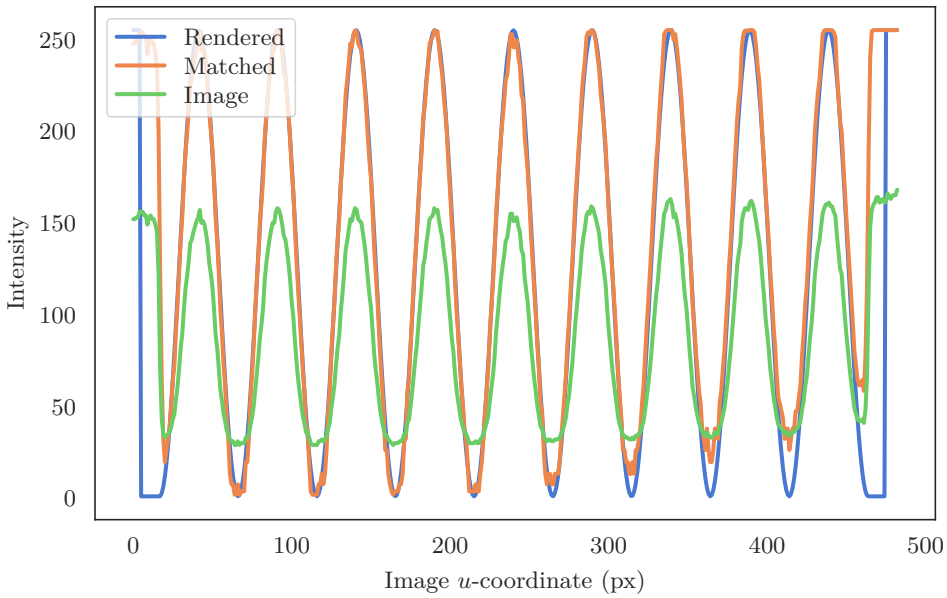
(b) A marker rendering to match the image section in Figure 3.14a using the coarse pose estimate.

(c) A histogram-matched image produced by applying the intensity mapping in Figure 3.13 to the image section in Figure 3.14a.

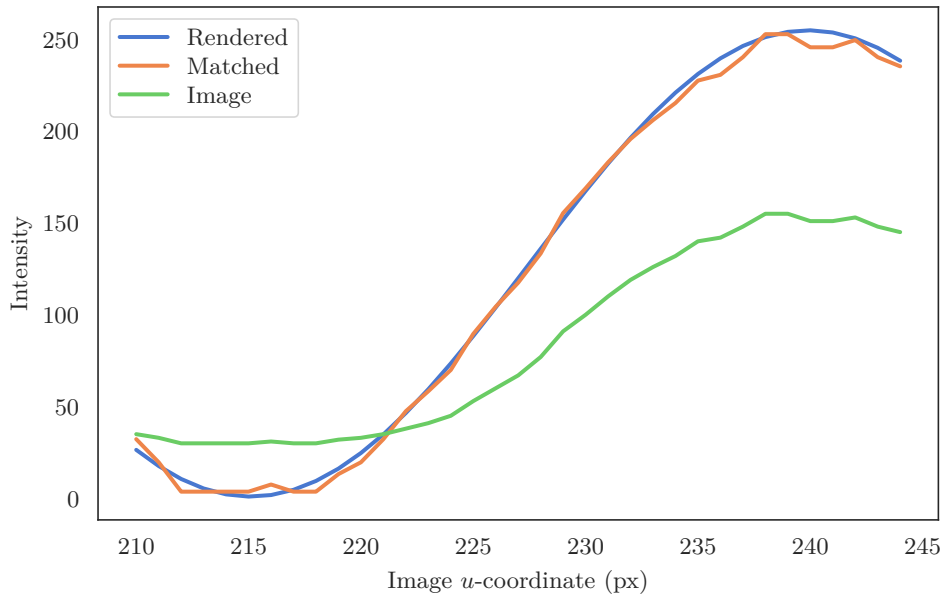


(d) The difference between the histogram-matched image in Figure 3.14c and the marker rendering in Figure 3.14b.

Figure 3.14 Images showing each stage of the optimisation process, using the image in Figure 3.11 as an example.



(a) The intensity along a horizontal line through the centres of the three images.



(b) A close-up of one section near the middle, showing one period of the pattern, showing that the intensity of the histogram-matched image closely follows that of the rendered image.

Figure 3.15 The intensity along a horizontal line through the centres of the image section in Figure 3.14a, the corresponding marker rendering in Figure 3.14b, and the histogram-matched image section in Figure 3.14c.

horizontal lines through the centres of the image section in Figure 3.14a, the corresponding marker rendering in Figure 3.14b, and the histogram-matched image section in Figure 3.14c.

3.5.3 Marker rendering

The pattern on the marker plane is given by

$$P(x, y) = \cos\left(2\pi n\sqrt{x^2 + y^2}\right), \tag{3.11}$$

where $\mathbf{p}_m = (x, y)$ is a point on the marker plane and n is the spatial frequency of the pattern. Evaluating this function at each point on a grid in image coordinates renders an image of the proposed marker, with the marker in some pose $\mathbf{T}_{cm}$. This can be performed by inverting

$$\mathbf{u} = \text{proj}\left(\mathbf{T}_{cm}\mathbf{p}_m\right), \tag{3.12}$$

where $\mathbf{u}$ is a point in image coordinates. The rendering has one major flaw: the pattern extends all the way to the edges (as shown in Figure 3.10), which interferes with the marker outline detection algorithm used for coarse pose estimation. To work around this, the sinusoid section of the pattern was limited to a whole number of periods chosen such that the centre of each edge is an intensity minimum:

$$P(x, y) = \cos\left(2\pi n \ \min\left(\sqrt{x^2 + y^2}, \frac{T}{n}\right)\right), \tag{3.13}$$

where T is the number of periods of the pattern from the centre to the edge. This produces the "bullseye" pattern shown in Figure 3.11.

3.5.4 Optimisation process

The objective function is the sum of squared differences between the intensities of the pixels in the section of the image containing the marker and a marker pattern rendered using the current parameter estimate:

$$e(\boldsymbol{\beta}) = \sum_{u,v} M(u, v; \boldsymbol{\beta})\left(I_{uv} - R(u, v; \boldsymbol{\beta})\right)^2, \tag{3.14}$$

where $\boldsymbol{\beta} = (x, y, z, \theta_x, \theta_y, \theta_z)$ is the parameter vector, I_{uv} is the pixel at position (u, v) in the image, $R(u, v; \boldsymbol{\beta})$ is a function which gives pixels from a rendering of the marker pattern on an infinite plane with the given parameters, and

$$M(u, v; \boldsymbol{\beta}) = \begin{cases} 1, & (u, v) \text{ inside marker} \\ 0, & \text{otherwise} \end{cases} \tag{3.15}$$

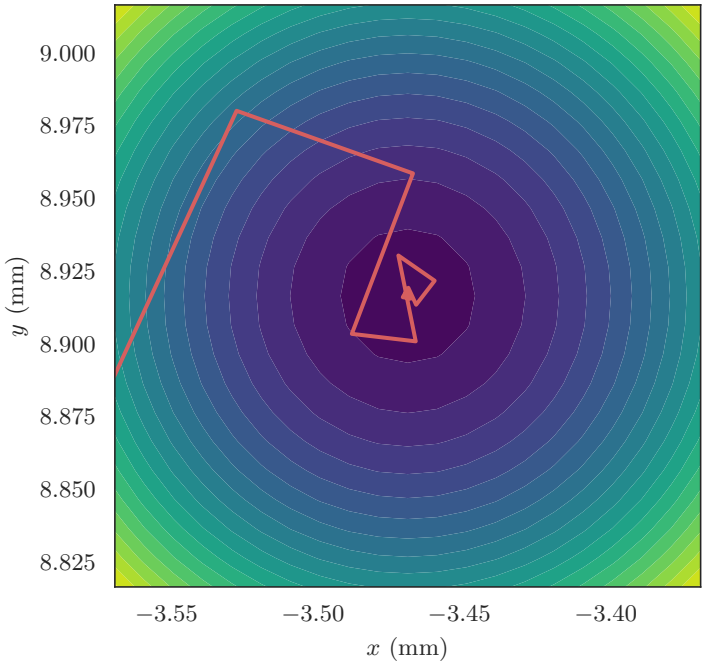


Figure 3.16 The error surface for the first step in optimising the pose estimate for the marker in Figure 3.11 (optimising x and y only, using the central section of the marker), showing the path the optimiser took as it converged.

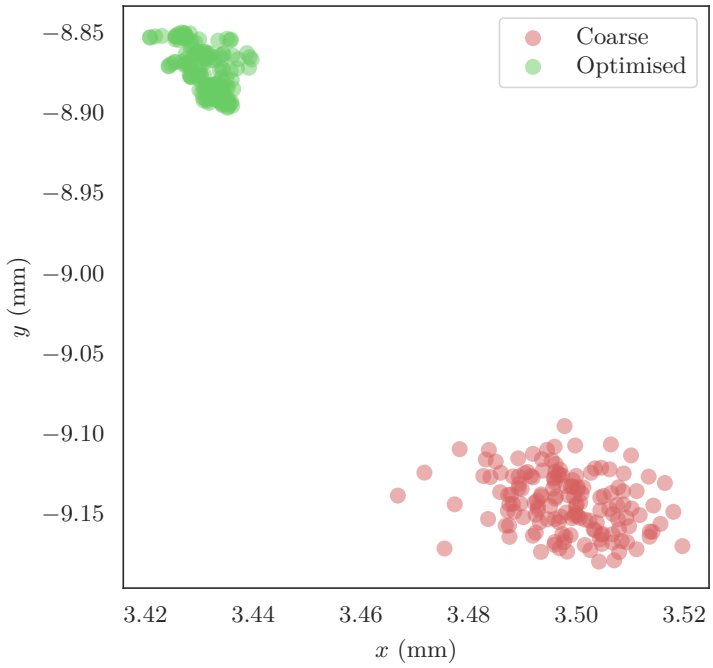


Figure 3.17 A scatter plot of the distribution of x- and y-coordinates in the coarse and optimised pose estimates for the set of 150 images containing Figure 3.11. The optimised estimates have significantly lower spread than the coarse estimates, and also a different mean.

is a mask function. The optimum parameter estimate is then given by

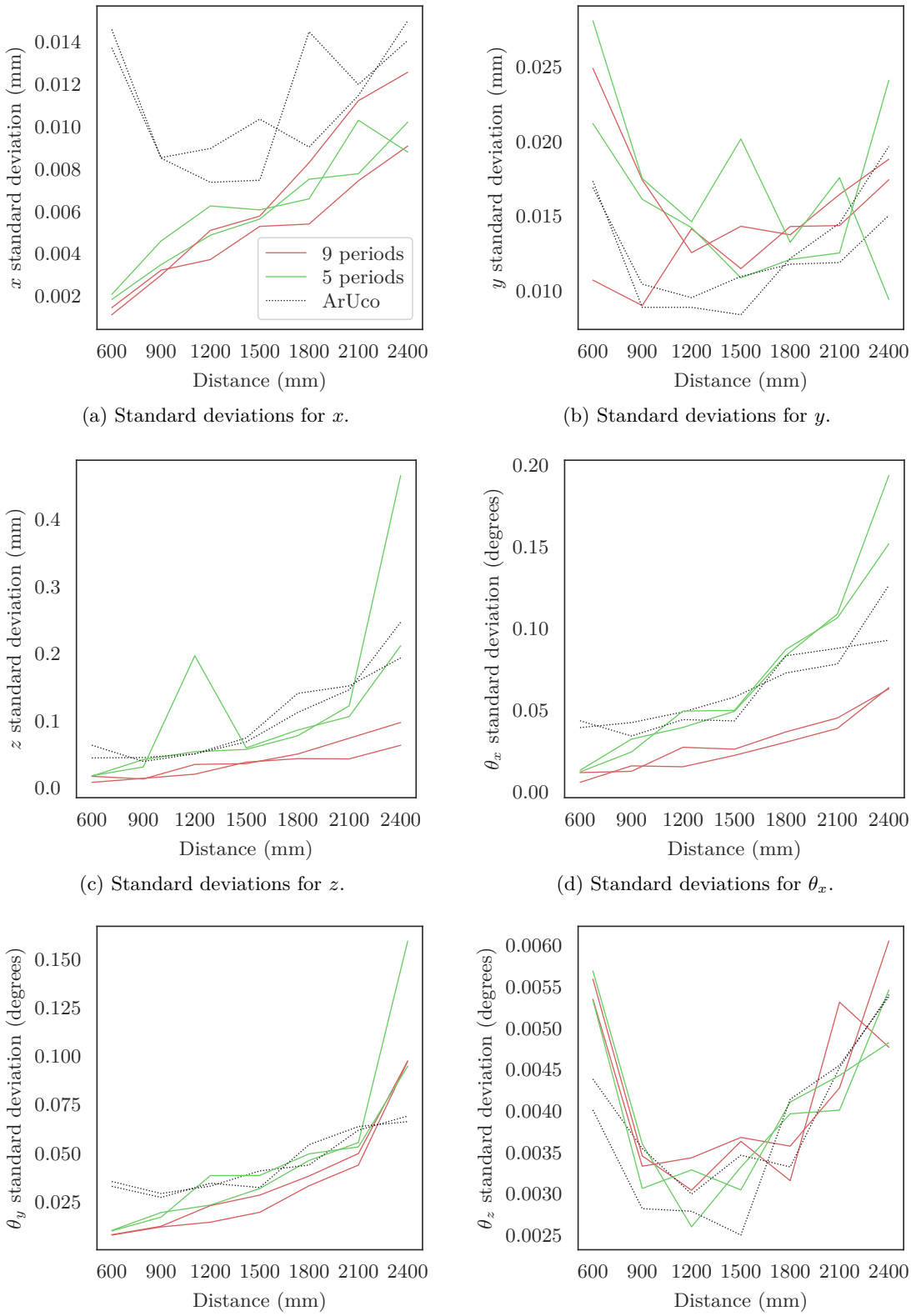
$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} e(\boldsymbol{\beta}), \tag{3.16}$$

which is found using Nelder-Mead optimisation [Nelder and Mead 1965]. An example of the (unsquared) difference image used here is shown in Figure 3.14d. Note that although the optimisation process does not vary θ_z, it is required in order to produce the correct mask.

The main limitation in this process is that the error surface must be sufficiently smooth and convex for the optimisation process to converge to a useful result. In practice, this is only true when the parameters are close to the true values, so to work around this, a series of optimisations are performed in which only a subset of the parameters are refined while the others are held constant to reduce the dimensionality of the error surface. In the first step, x and y are refined. This step is not as sensitive to error in the other parameters; error in θ_x and θ_y have a greater effect further from the centre, so only the central section of the marker is used. The error surface and convergence path for this step in refining the pose for the marker from Figure 3.11 are shown in Figure 3.16. Next, z is refined, then x, y, and z together. Finally, θ_x and θ_y are refined using the whole marker, then all five parameters together. In Figure 3.17, the distributions of x and y before and after optimisation for the set of images containing Figure 3.11 are compared.

3.6 RESULTS AND DISCUSSION

The position of the marker in each image was estimated (as described in Section 3.5) for each of the six datasets (one with the nine-period marker, one with the five-period marker, one with an ArUco marker, then a replication of each), and the standard deviation for each parameter-distance-dataset combination was calculated. In Figure 3.18, the standard deviations are plotted against distance for each dataset. In Table 3.1, the standard deviations for each of the three markers (i.e., the average of standard deviations for the two runs for each marker) are shown. The same data is shown normalised by the ArUco standard deviation for each point in Figure 3.19, such that normalised standard deviations below one are improvements on ArUco's performance. In Table 3.2, the ratio of the nine-period marker results to the ArUco marker results from Table 3.1 are shown. The nine-period marker generally does better than the five-period marker, which is unsurprising given that the sinusoid pattern is well-sampled throughout (approximately 25 pixels peak-to-peak for the nine-period marker at the greatest distance) but the nine-period marker has high spatial frequency, as discussed in Section 3.4.3. The nine-period marker estimates θ_x, θ_y, x, and z with significantly more precision than ArUco. Overall, the nine-period marker has 13.8 %–48.4 % lower standard deviation



(a) Standard deviations for x.

(b) Standard deviations for y.

(c) Standard deviations for z.

(d) Standard deviations for θ_x.

(e) Standard deviations for θ_y.

(f) Standard deviations for θ_z.

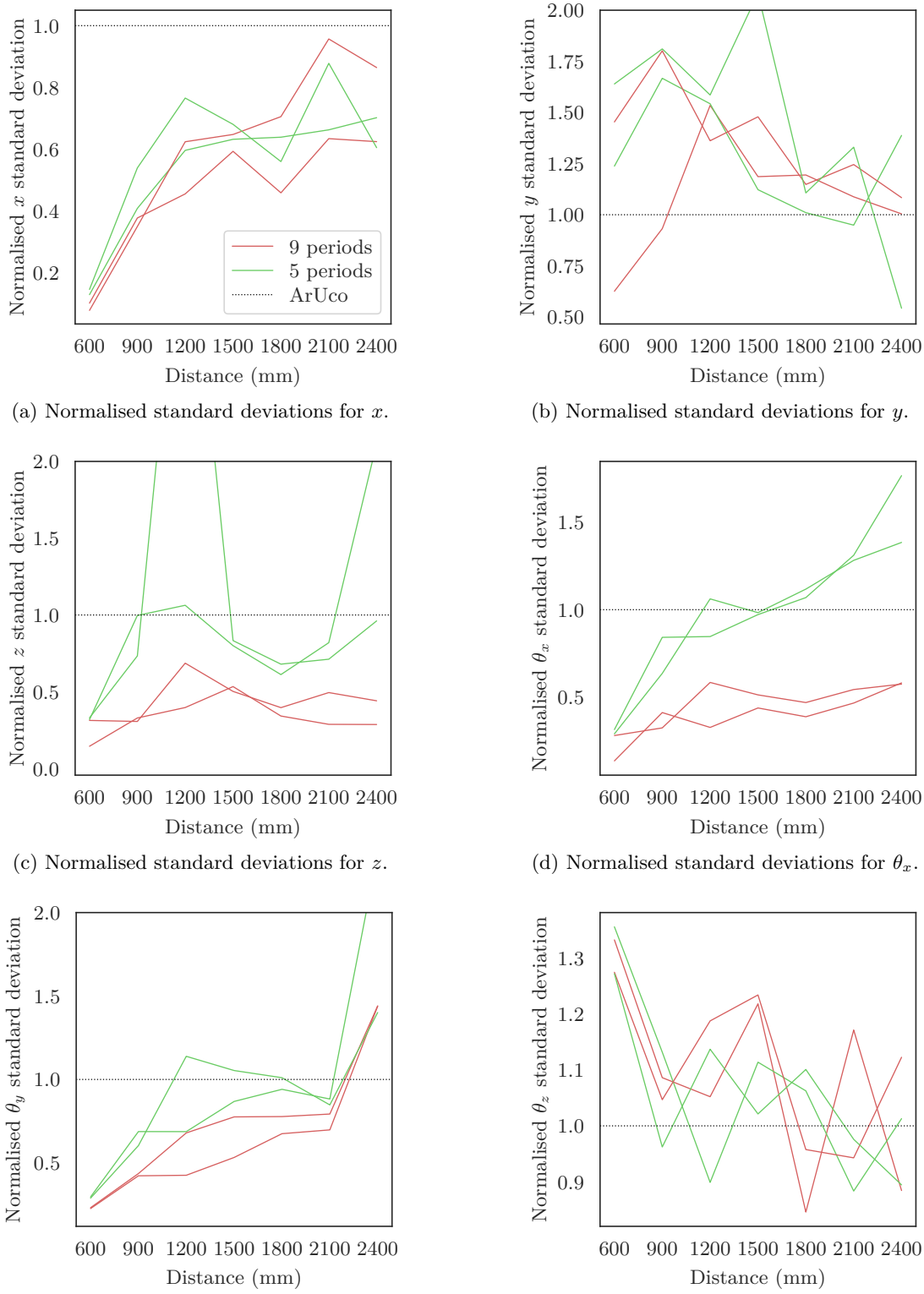
Figure 3.18 The standard deviations of the estimates of each parameter for each dataset and distance. There are two datasets for each marker type, as each experiment was repeated twice. Note that θ_z is not optimised beyond the coarse estimate, so its three estimates use effectively the same method.

Table 3.1 The standard deviation for each parameter (in millimetres and degrees) at each distance, averaged between the two datasets for each of the three markers evaluated. The lowest standard deviation for each parameter-distance pair appears in boldface.

		Distance (mm)							Mean
		600	900	1200	1500	1800	2100	2400	
9 periods	x	**0.0013**	**0.0031**	**0.0044**	**0.0055**	**0.0068**	0.0093	0.011	**0.0059**
	y	0.018	0.013	0.013	0.013	0.014	0.015	0.018	0.015
	z	**0.012**	**0.013**	**0.027**	**0.037**	**0.047**	**0.058**	**0.08**	**0.039**
	θ_x	**0.0085**	**0.014**	**0.021**	**0.024**	**0.033**	**0.042**	**0.063**	**0.029**
	θ_y	**0.0078**	**0.012**	**0.019**	**0.024**	**0.036**	**0.047**	0.097	**0.035**
	θ_z	0.0055	0.0034	0.0032	0.0037	**0.0034**	0.0048	0.0054	0.0042
5 periods	x	0.002	0.004	0.0056	0.0058	0.007	**0.009**	**0.0095**	0.0061
	y	0.025	0.017	0.014	0.016	0.013	0.015	**0.017**	0.017
	z	0.017	0.036	0.13	0.058	0.082	0.11	0.34	0.11
	θ_x	0.012	0.028	0.044	0.049	0.085	0.11	0.17	0.071
	θ_y	0.01	0.018	0.031	0.035	0.048	0.054	0.13	0.046
	θ_z	0.0055	0.0033	0.0029	0.0032	0.004	**0.0042**	**0.0051**	0.0041
ArUco	x	0.014	0.0085	0.0082	0.0089	0.012	0.012	0.015	0.011
	y	**0.017**	**0.0097**	**0.0092**	**0.0097**	**0.012**	**0.013**	0.017	**0.013**
	z	0.054	0.042	0.05	0.071	0.13	0.15	0.22	0.1
	θ_x	0.041	0.038	0.046	0.05	0.078	0.083	0.11	0.064
	θ_y	0.034	0.028	0.034	0.036	0.049	0.063	**0.068**	0.045
	θ_z	**0.0042**	**0.0032**	**0.0029**	**0.003**	0.0037	0.0045	0.0054	**0.0038**

Table 3.2 Normalised standard deviations of the estimate of each parameter for the nine-period marker at each distance, presented as the ratio of the mean of the two nine-period datasets divided by the mean of the two ArUco datasets.

	Distance (mm)							Mean
	600	900	1200	1500	1800	2100	2400	
x	0.09	0.36	0.54	0.62	0.58	0.80	0.74	0.53
y	1.04	1.37	1.45	1.33	1.17	1.17	1.04	1.22
z	0.23	0.32	0.54	0.52	0.37	0.39	0.36	0.39
θ_x	0.21	0.37	0.45	0.47	0.43	0.50	0.58	0.43
θ_y	0.23	0.43	0.55	0.65	0.73	0.75	1.44	0.68
θ_z	1.30	1.07	1.12	1.23	0.90	1.06	1.00	1.10
Mean	0.52	0.65	0.78	0.80	0.70	0.78	0.86	0.73



(a) Normalised standard deviations for x.

(b) Normalised standard deviations for y.

(c) Normalised standard deviations for z.

(d) Normalised standard deviations for θ_x.

(e) Normalised standard deviations for θ_y.

(f) Normalised standard deviations for θ_z.

Figure 3.19 The standard deviations of the estimates of each parameter for each dataset and distance as in Figure 3.18, but with each data point normalised by the standard deviation for the corresponding ArUco marker. There are two datasets for each marker type, as each experiment was repeated twice. Note that θ_z is not optimised beyond the coarse estimate, so its three estimates use effectively the same method.

than the ArUco marker (average 27.4 %).

Note that the camera is not sharply focused for the first few distances, as can be seen by comparing the edges in Figures 3.12a and 3.12b to those in Figure 3.11. This makes comparing the proposed marker to ArUco a bit unfair since neither claims to provide precise estimates in blurry images.

The similarity in θ_z between ArUco and the proposed marker is because θ_z is not optimised in the procedure described in Section 3.5.4. In fact, a shortcoming of the proposed algorithm is that the marker outline is only used for the coarse pose estimate. If the optimisation process incorporated the edge or corner locations of the marker, then θ_z could be jointly optimised along with the other parameters, which could result in better estimation overall.

The experiment was initially carried out with both an ArUco marker and an April-Tag marker [Olson 2011], but the AprilTag detection algorithm failed in many of the images so the results are not included here.

This experiment gives a measure of precision, not accuracy; evaluating accuracy would require a method for measuring the position of the camera's focal point with sub-millimetre accuracy and the direction of the camera's optical axis with sub-degree accuracy. A robot arm could be used to evaluate relative positioning accuracy; this could be performed in future work. Another issue with the experimental design is that estimation precision was only evaluated for markers in the centre of the image, oriented parallel to the camera. In Chapter 4, it is shown in simulation that this is the worst case for estimating θ_x and θ_y with a checkerboard. This finding most likely applies to the ArUco marker and may or may not apply to the proposed marker; a more thorough experiment would include a variety of positions and angles in order to avoid unfairly favouring one marker over another.

The sinusoid pattern was originally chosen with the hope that frequency-domain methods could be used (in a similar way to Roberts et al. [2015]), however, this approach did not give good results. In the one-dimensional case (i.e., when estimating only distance with every other parameter held constant) frequency-domain methods are perfectly adequate, and this could be extended to four parameters (all but θ_x and θ_y) without much trouble using the proposed marker. As soon as the marker is not parallel to the camera there is no longer a single dominant spatial frequency in the image, and since the only advantage of the frequency-domain method is its simplicity, there remains no motivation for its use.

It should be noted that, unlike many other markers, this system does not allow markers to be identified individually. Although it is possible to modify the marker design to add this function, it is not strictly required for use as ground truth—an accurate relative position is sufficient. An easier solution for other applications could be to simply use these markers in conjunction with standard fiducial markers, or incorporate

a standard fiducial marker into the design.

The sinusoid pattern is also convenient because it can be rendered pixel-by-pixel, which is fast for small markers. Simply warping the marker texture gives similar runtime performance and precision in many cases (although comparing the optimised perspective warp routine from OpenCV with the reasonably naive Python marker renderer is unfair and a detailed comparison has yet to be made), so this may not be an advantage. In particular, Hannemose et al. [2019] later developed a camera calibration routine with a very similar approach to the pose estimation algorithm presented here (rendering an image, updating parameters to minimise squared image difference, and repeating until convergence) but simply used a checkerboard pattern with Gaussian blur, which improved performance slightly over previous methods for checkerboard-based camera calibration. The simpler calibration pattern let them produce analytical derivatives for the rendering function, which is a significant advantage for the numerical optimisation process. Similarly, Schöps et al. [2020] developed a new Siemens star-like feature for their camera calibration routine, with a rendering and optimisation-based feature refinement method. First, approximate feature locations are calculated relative to a central AprilTag. Then, these locations are refined by evaluating a rendering function at each of a large set of random points within a local window, comparing the intensities to the image, then optimising for a translation and a brightness transformation. This is followed by a further refinement step which exploits the symmetry of the feature to optimise a local homography, notably using random supersampling to avoid introducing bias from bilinear interpolation.[4] These improvements could be incorporated into the approach presented here to further increase performance and robustness, although it is unclear how close this approach is currently to achieving the highest possible precision.

3.7 CONCLUSION

A new approach to pose estimation using fiducial markers was proposed that is more precise than existing algorithms. Rather than using only corners or edges for pose estimation, the marker has a radial sinusoid pattern that has a predictable appearance under perspective projection. The marker pose is initially estimated using traditional methods, then refined using a novel optimisation method in which a rendering of the marker is compared with the image. On average, pose estimation precision was increased by 27 % compared to traditional fiducial markers. Future work could investigate alternate marker patterns and evaluate the algorithm's performance in less-controlled conditions.

[4] See Chapter 6.

Chapter 4

CHECKERBOARD POSE ESTIMATION PRECISION

In Chapter 3, a fiducial marker system was developed that attempted to estimate pose with high precision by effectively increasing the salient area from only the corners to the whole face of the marker. This could be conceptualised as the limiting case when increasing the number of features used for pose estimation until the marker is filled with features. The resulting marker did give increased precision, but it was not the "magic bullet" for outdoor ground truth that was its motivation. In this chapter, which is based on published work [Edwards et al. 2017], checkerboards are used as generic, easily modellable substitutes to reason about the effects of various characteristics of a fiducial marker/camera system on its pose estimation precision. First, a statistical lower bound is derived analytically for the variance of an optimal pose estimate performed using a checkerboard. This bound is then used, along with simulations and some real imagery, to broadly illustrate how the maximum achievable precision for each pose parameter (i.e., x-/y-/z-axis translations and rotations) depends on the corner estimation noise and the checkerboard's pose, size, and number of squares.

4.1 INTRODUCTION

A checkerboard is a grid of alternating black and white squares, as shown in Figure 4.1. Checkerboards are most commonly used in computer vision for camera calibration [as in Zhang 2000],[1] where the camera's intrinsic parameters are estimated using a collection of images of a checkerboard in different poses, but they can also be used for pose estimation [Siebert and Bogusław 2009].

A fiducial marker is an object used to determine relative camera pose. The analysis in this chapter applies to square planar fiducial markers (as shown in Figures 3.1a to 3.1c), which use the marker corners for pose estimation, and other fiducial markers that estimate pose from a series of interest points (like RUNE-Tag, as shown in Figure 3.1d), but not necessarily to markers using conics or other methods for pose estimation (like the less typical markers shown in Figures 3.1e to 3.1g).

[1] Although Zhang [2000] used a slightly different pattern to the modern checkerboard, the method remains the same.

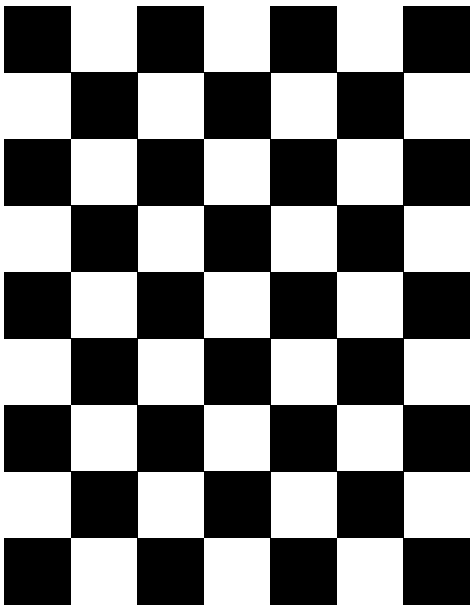


Figure 4.1 An 8×6 checkerboard (with 9×7 squares).

From a pose estimation perspective, a square planar fiducial marker is equivalent to a checkerboard, in that for either object, visual methods are used to estimate the image locations of a set of known points on the marker, the points are matched to their known 3D structure, and then these 3D-to-2D correspondences are used to solve the PnP problem (see Section 2.5). Many VO algorithms contain a 3D-to-2D matching step [Scaramuzza and Fraundorfer 2011] which is also equivalent.

When designing a system around any of these fiducial markers, it is desirable to have a predictive model of pose estimation performance, and for sensor fusion, a full covariance model is required. Mihalyi et al. [2013] presented models of the rotation and translation covariances for a fiducial marker, which were then used in a pose graph optimisation problem. Swapna et al. [2009] performed a Design of Experiments study of camera calibration using a checkerboard and gave the distribution of each intrinsic parameter in a particular configuration. Matsunaga and Kanatani [2000] presented some analysis of the problem of estimating camera pose and focal length from an infinite checkerboard pattern, including deriving an estimate of the CRLB from the residual. Their formulation enables reliability analysis given actual data but cannot make predictions for a scenario before data is collected. Davis et al. [2003] presented an error propagation study of the pose estimation precision for two specific optical tracking probes (and then verified their results experimentally), but only considered marker size and detection noise and did not fit a model to their results. Kanatani and Ohta [2001] derived an analytic model for the CRLB for visual localisation but did not evaluate it in practice. Rohde et al. [2016] proposed an analytical model of an upper bound

for localisation uncertainty in terms of odometry covariance, landmark covariance, and observation probability. Their method does not use PnP, but instead compares landmark observations to a map with the same dimensionality. The lower bounds on precision for related visual estimation problems, for example, sub-pixel image registration [Robinson and Milanfar 2004, Uss et al. 2014] and homography estimation [Acuna and Willert 2018, Chen and Suter 2009], have been studied in depth.

Some notable related research was published after the research presented in this chapter was performed. Acuna and Willert [2018] focused on planar pose estimation via homography decomposition, a special case of the pose estimation problem considered here which is specific to images of planes.[2] Their main contribution is a method for finding optimal point configurations, which was informed by a CRLB for homography estimation. The conclusion was that the optimal point configuration for a given situation is the one in which the points are the greatest distance apart. Zhang and Scaramuzza [2018] focused on trajectory planning for UAVs, presenting a navigation system that includes perception quality in its trajectory search (*active SLAM*). For each trajectory it considered, the Fisher information matrix (FIM) for each pose along the trajectory was calculated; the combined information was used as a measure of perception quality for the trajectory. They later extended this approach with a new information representation (the *Fisher information field*) to dramatically reduce the computation required [Zhang and Scaramuzza 2019].

The rest of this chapter is structured as follows: Section 4.2 introduces notation and background theory, Section 4.3 derives the CRLB for checkerboard pose estimation, Section 4.4 presents and discusses results from models, simulations, and real image data, and Section 4.5 draws conclusions.

4.2 BACKGROUND

An $N \times M$ checkerboard is an $N \times M$ grid of corner points, $\mathbf{x}_k = [x_m, y_n, 0]^{\mathrm{T}}$, where $k = mN + n$ and $m = 0, 1, \ldots M-1$, $n = 0, 1, \ldots N-1$. In this context, a "corner" specifically refers to the internal corners of the checkerboard, i.e., the points where the corners of two squares touch. Each of the $(N+1) \times (M+1)$ squares has width and height d, so the positions of the corners relative to the centre of the checkerboard, $\mathbf{c}_o^{(m)}$, are

$$\begin{aligned} c_{o,x}^{(m)} &= md - \frac{(M-1)d}{2} \\ c_{o,y}^{(n)} &= nd - \frac{(N-1)d}{2} \\ c_{o,z} &= 0, \end{aligned} \tag{4.1}$$

[2] Although this chapter is about estimating pose using images of planes, the method analysed here does not depend on the points being planar.

where $m = 0, 1, \ldots, M-1$ and $n = 0, 1, \ldots, N-1$. The method for estimating the pose of a checkerboard from an image is the same as for a fiducial marker,[3] as discussed in Section 2.5.

4.2.1 The Cramér-Rao lower bound

The *Fisher information matrix* is a measure of the information that a sequence of observations carries about a parameter to be estimated. Given a sequence of observations,

$$\{\mathbf{X}\} = (X_0, X_1, \ldots X_k),$$

where each observation X_n is a random variable, a parameter vector $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$, and an unbiased estimator $\hat{\boldsymbol{\beta}}(\mathbf{X})$, the FIM is the expected value of the Hessian of the negative log-likelihood of the observations,

$$[\mathbf{I}(\boldsymbol{\beta})]_{ij} = -E\left[\frac{\partial^2 \ln l(\{\mathbf{X}\}\,;\boldsymbol{\beta})}{\partial\beta_i \partial\beta_j}\right], \tag{4.2}$$

where $l(\{\mathbf{X}\}\,;\boldsymbol{\beta})$ is the likelihood function. Note that the expectation is taken with respect to $l(\{\mathbf{X}\}\,;\boldsymbol{\beta})$, and the true value of $\boldsymbol{\beta}$ is used. The *Cramér-Rao lower bound* gives the minimum variance of any unbiased estimator [Kay 1993, chapter 3], which can be found as the diagonal elements of the inverse of the FIM:

$$\mathrm{Var}\left(\hat{\beta}_i\right) \geq [\mathbf{I}(\boldsymbol{\beta})]_{ii}^{-1}. \tag{4.3}$$

More generally, the covariance matrix of any unbiased estimator cannot be made smaller than the inverse of the FIM [Watanabe 2009, chapter 1].

4.3 CHECKERBOARD CRLB DERIVATIONS

In this section, the CRLB for the variance of a checkerboard pose estimate is derived, first for a simplified 1D case then for the full case. The pose is parametrised as $\boldsymbol{\beta} = (x, y, z, \theta_x, \theta_y, \theta_z)$, where x is a translation along the x-axis, y is a translation along the y-axis, z is a translation along the z-axis, θ_x is a rotation about the x-axis, θ_y is a rotation about the y-axis, and θ_z is a rotation about the z-axis.

For an $N \times M$ checkerboard, the observations are a sequence of measured corner locations,

$$\{\mathbf{U}\} = \{\mathbf{U}_0, \mathbf{U}_1, \ldots, \mathbf{U}_{NM}\},$$

[3] Although almost any checkerboard has more corner points than a fiducial marker.

in image coordinates. Each observation,

$$\begin{aligned}\mathbf{U}_i &= \begin{pmatrix} U_i \\ V_i \end{pmatrix} \\ &= \mathbf{u}_i + \mathbf{W}_i,\end{aligned} \tag{4.4}$$

is a random vector, where $\mathbf{u}_i$ is the true corner location and $\mathbf{W}_i$ is an additive random noise vector. Assuming each observation is independent, the likelihood function for the measured data is

$$l\left(\{\mathbf{U}\};\boldsymbol{\beta}\right) = \prod_{i=1}^{NM} f\left(\mathbf{U}_i;\boldsymbol{\beta}\right), \tag{4.5}$$

where $f\left(\mathbf{U}_i;\boldsymbol{\beta}\right)$ is the probability distribution function (PDF) of $\mathbf{U}_i$. Since the noise is additive, the joint PDF of $\mathbf{U}_i$ is

$$f\left(\mathbf{U}_i;\boldsymbol{\beta}\right) = f_{\mathbf{W}_i}\left(\mathbf{U}_i - \mathbf{u}_i\right). \tag{4.6}$$

Assuming the noise is zero-mean Gaussian, the joint noise PDF is

$$f_{\mathbf{W}_i}(\mathbf{w}_i) = \frac{1}{\sqrt{(2\pi)^2 \det\left(\Sigma_{\mathbf{W}_i}\right)}} \exp\left(-\frac{1}{2}\mathbf{w}_i^{\mathrm{T}}\Sigma_{\mathbf{W}_i}^{-1}\mathbf{w}_i\right), \tag{4.7}$$

where $\Sigma_{\mathbf{W}_i}$ is the noise covariance matrix for the ith observation. Assuming the noise is independent for the u- and v-coordinates of each observation,

$$\Sigma_{\mathbf{W}_i} = \mathbf{I}\sigma^2, \tag{4.8}$$

where $\mathbf{I}$ is a 2×2 identity matrix, so the joint noise PDF is

$$f_{\mathbf{W}_i}(\mathbf{w}_i) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{2\sigma^2}\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i\right). \tag{4.9}$$

The joint PDF of $\mathbf{U}_i$ is then

$$f\left(\mathbf{U}_i;\boldsymbol{\beta}\right) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{2\sigma^2}\left(\mathbf{U}_i - \mathbf{u}_i\right)^{\mathrm{T}}\left(\mathbf{U}_i - \mathbf{u}_i\right)\right). \tag{4.10}$$

Substituting (4.10) into (4.5) gives the likelihood function

$$l\left(\{\mathbf{U}\};\boldsymbol{\beta}\right) = \prod_{i=1}^{NM} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{2\sigma^2}\left(\mathbf{U}_i - \mathbf{u}_i\right)^{\mathrm{T}}\left(\mathbf{U}_i - \mathbf{u}_i\right)\right), \tag{4.11}$$

and hence the log-likelihood function is

$$\ln\left(l\left(\{\mathbf{U}\};\boldsymbol{\beta}\right)\right) = \sum_{i=1}^{NM}\left(-\ln 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left(\mathbf{U}_i - \mathbf{u}_i\right)^{\mathrm{T}}\left(\mathbf{U}_i - \mathbf{u}_i\right)\right). \tag{4.12}$$

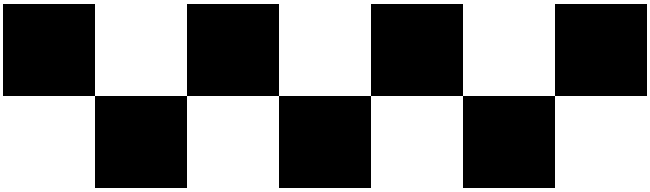


Figure 4.2 A 1×6 checkerboard (with 2×7 squares), as in the simplified CRLB derived in Section 4.3.1.

4.3.1 Simplified 1D case

Consider a simplified case with an $1 \times M$ "checkerboard" (as illustrated in Figure 4.2) and all parameters aside from x and z fixed to 0 and not estimated. The positions of the corners relative to the centre of the checkerboard (i.e., in the object frame, $\{o\}$), $\mathbf{c}_o^{(m)}$, are

$$c_{o,x}^{(m)} = md - \frac{(M-1)d}{2} \tag{4.13}$$

$$c_{o,y} = c_{o,z} = 0, \tag{4.14}$$

where d is the width of the checkerboard. The positions in world coordinates are

$$\mathbf{c}_w = \mathbf{c}_o + \mathbf{t}_{wo} \tag{4.15}$$

$$= \begin{pmatrix} c_{o,x} + x \\ 0 \\ z \end{pmatrix}; \tag{4.16}$$

note that variables are referred to without their indices except when necessary to avoid ambiguity. Given a pinhole camera at the origin of the world frame, $\{w\}$, the image location $\mathbf{u}_m = (u_m, v_m)$ of each corner is given by

$$\mathbf{u} = \frac{f}{c_{w,z}} \begin{pmatrix} c_{w,x} \\ c_{w,y} \end{pmatrix} \tag{4.17}$$

$$= \begin{pmatrix} \frac{f}{z}(c_{o,x} + x) \\ 0 \end{pmatrix}, \tag{4.18}$$

as in Section 2.4.

Once an image of the checkerboard has been taken, a corner detection algorithm is used to estimate the corner locations in image coordinates,

$$U = \frac{f}{z}(c_{o,x} + x) + W, \tag{4.19}$$

where W is a zero-mean Gaussian random process that models the corner detection errors.[4] Assuming the errors are i.i.d.[5] with variance σ^2, the likelihood function for the measured data is

$$l\left(\{\mathbf{U}\};\boldsymbol{\beta}\right) = \prod_{m=0}^{M-1} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{2\sigma^2}\left(U_m - \frac{fx_\Delta}{z}\right)^2\right), \tag{4.20}$$

where $\boldsymbol{\beta}$ is the parameter vector (x, z) and $x_\Delta = x + x_m$. The Hessian of the negative log-likelihood is

$$\mathbf{H} = \sum_{m=0}^{M-1} \frac{f}{\sigma^2 z^4} \begin{bmatrix} fz^2 & -\left(2fx_\Delta - uz\right)z \\ -\left(2fx_\Delta - uz\right)z & 3fx_\Delta^2 - 2ux_\Delta z \end{bmatrix}. \tag{4.21}$$

Substituting (4.13) and the expected value of U_m then evaluating the sum gives the FIM,

$$\mathbf{I}\left(\beta\right) = \frac{Mf^2}{\sigma^2 z^3} \begin{bmatrix} z & -x \\ -x & \frac{1}{12z}\left((M^2-1)d^2 + 12x^2\right) \end{bmatrix}. \tag{4.22}$$

Taking the inverse gives the CRLB matrix,

$$\mathbf{I}^{-1}\left(\beta\right) = \frac{12\sigma^2 z^2}{M\left(M^2-1\right)d^2 f^2} \begin{bmatrix} \frac{d^2}{12}\left(M^2-1\right) + x^2 & xz \\ xz & z^2 \end{bmatrix}. \tag{4.23}$$

The diagonal elements give lower bounds for the variances of the estimates of x and z:

$$\mathrm{Var}\left(\hat{x}\right) \geq \frac{\left(\left(M^2-1\right)d^2 + 12x^2\right)\sigma^2 z^2}{M\left(M^2-1\right)d^2 f^2}, \tag{4.24}$$

$$\mathrm{Var}\left(\hat{z}\right) \geq \frac{12\sigma^2 z^4}{M\left(M^2-1\right)d^2 f^2}. \tag{4.25}$$

Note that $\hat{x}$ or $\hat{x}\left(\{\mathbf{U}\}\right)$ is a random variable representing an estimator of x from the measured data $\{\mathbf{U}\}$, and so on for $\hat{z}$ and later the other parameters. For large M, this simplifies to

$$\begin{pmatrix} \mathrm{Var}\left(\hat{x}\right) \\ \mathrm{Var}\left(\hat{z}\right) \end{pmatrix} \geq \frac{\sigma^2 z^2}{M^2 d^2 f^2} \begin{pmatrix} Md^2 + 12x^2 \\ 12z^2 \end{pmatrix}. \tag{4.26}$$

A clearer expression can be obtained by combining the focal length and square size into one variable roughly equivalent to the width (and height) of each square in pixels,[6]

[4] From (4.19) onward, the v-coordinate is neglected for simplicity; $\hat{x}$ and $\hat{z}$ are estimated from the u-coordinate alone.

[5] Isotropy, homogeneity, and independence are not trivial assumptions; see Chapters 5 and 6.

[6] For a marker that is parallel to the camera, this description is accurate. For a marker that is rotated about the x- or y-axis, d_i is the size in pixels of a square placed parallel to the camera at the centre of the marker. Either way, it is easier to reason about than camera focal length.

$$d_i = \frac{df}{z}, \tag{4.27}$$

giving

$$\begin{pmatrix} \mathrm{Var}\,(\hat{x}) \\ \mathrm{Var}\,(\hat{z}) \end{pmatrix} \geq \frac{\sigma^2}{M^2 {d_i}^2} \begin{pmatrix} Md^2 + 12x^2 \\ 12z^2 \end{pmatrix}. \tag{4.28}$$

These results show that the precision of the pose estimate increases with increasing corner detection precision, number of corners, and square size in pixels. Thus a checkerboard with more corners should give a better estimate. However, the corners need to be resolvable, which places an upper limit on the number of corners for a given size.

4.3.2 Full case

Consider now the full problem, where the parameter vector to be estimated is $\boldsymbol{\beta} = (x, y, z, \theta_x, \theta_y, \theta_z)$ and the checkerboard has $N \times M$ corners. The corner positions in world coordinates are

$$\mathbf{c}_w = \mathbf{R}_{wo}\left(\theta_z, \theta_y, \theta_x\right)\mathbf{c}_o + \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \tag{4.29}$$

and the image location of each corner is

$$\begin{aligned} \mathbf{u} &= g(\mathbf{c}_w; \boldsymbol{\beta}) \\ &= \frac{f}{c_{w,z}} \begin{pmatrix} c_{w,x} \\ c_{w,y} \end{pmatrix}. \end{aligned} \tag{4.30}$$

Corner detection is used to measure their locations in image coordinates,

$$\mathbf{U} = g(\mathbf{c}_w; \boldsymbol{\beta}) + \mathbf{W}, \tag{4.31}$$

where $\mathbf{W}$ is a zero-mean multivariate Gaussian random process that models the corner detection errors. Assuming again that the errors are i.i.d. with variance σ^2, the likelihood function for the measured data is

$$l\left(\{\mathbf{U}\}; \boldsymbol{\beta}\right) = \prod_{n=0}^{N-1} \prod_{m=0}^{M-1} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{2\sigma^2} \boldsymbol{\Delta}_{m,n}^{\mathrm{T}} \boldsymbol{\Delta}_{m,n}\right), \tag{4.32}$$

where

$$\boldsymbol{\Delta}_{m,n} = U_{m,n} - g(\mathbf{c}_w^{(m,n)}; \boldsymbol{\beta}). \tag{4.33}$$

The FIM is

$$[\mathbf{I}(\boldsymbol{\beta})]_{ij} = -E\left[\frac{\partial^2 \ln l\left(\{\mathbf{U}\};\boldsymbol{\beta}\right)}{\partial\beta_i\partial\beta_j}\right], \tag{4.34}$$

which, when evaluated symbolically, is too complicated to print. It is also too complicated to invert symbolically, although it is computable numerically.

Some insight into the effect of the various parameters on estimation variance can be obtained by fixing some parameters in order to simplify the FIM expression. Setting $M = N$, $y = 0$, $\theta_x = \theta_y = \theta_z = 0$ (that is, considering a square checkerboard held parallel to the camera moving sideways along the x-axis) is enough to allow the CRLB to be calculated symbolically using the Sympy computer algebra system [Meurer et al. 2017], giving

$$\begin{pmatrix} \mathrm{Var}\left(\hat{x}\right) \\ \mathrm{Var}\left(\hat{y}\right) \\ \mathrm{Var}\left(\hat{z}\right) \\ \mathrm{Var}\left(\hat{\theta_z}\right) \\ \mathrm{Var}\left(\hat{\theta_x}\right) \\ \mathrm{Var}\left(\hat{\theta_y}\right) \end{pmatrix} \geq \alpha(\sigma, d, f, x, z, N) \begin{pmatrix} \left(N^2-1\right)\left(7N^2-13\right)d^4 + 12\left(N^2-4\right)d^2x^2 + 180x^4 \\ \left(N^2-1\right)\left(\left(7N^2-13\right)d^2 + 15x^2\right)d^2 \\ 9\left(\left(3N^2-7\right)d^2 + 20x^2\right)z^2 \\ 9\left(3N^2-7\right)d^2 + 180x^2 \\ 360z^2 \\ 360z^2 \end{pmatrix}, \tag{4.35}$$

where

$$\alpha(\sigma, d, f, x, z, N) = \frac{2\sigma^2 z^2}{3N^2\left(N^2-1\right)\left(\left(3N^2-7\right)d^2 + 10x^2\right)d^2 f^2}. \tag{4.36}$$

For large N, this simplifies to

$$\begin{pmatrix} \mathrm{Var}\left(\hat{x}\right) \\ \mathrm{Var}\left(\hat{y}\right) \\ \mathrm{Var}\left(\hat{z}\right) \\ \mathrm{Var}\left(\hat{\theta_z}\right) \\ \mathrm{Var}\left(\hat{\theta_x}\right) \\ \mathrm{Var}\left(\hat{\theta_y}\right) \end{pmatrix} \geq \alpha^*(\sigma, d, f, z, N) \begin{pmatrix} \frac{7d^4}{6} \\ \frac{7d^4}{6} \\ \frac{9d^2z^2}{2N^2} \\ \frac{9d^2}{2N^2} \\ \frac{60z^2}{N^4} \\ \frac{60z^2}{N^4} \end{pmatrix}, \tag{4.37}$$

where

$$\alpha^*(\sigma, d, f, z, N) = \frac{4\sigma^2 z^2}{3N^2 d^4 f^2}. \tag{4.38}$$

Simplifying again using (4.27) gives

$$\alpha'(\sigma, d, d_i, z, N) = \frac{4\sigma^2}{3N^2 d^2 {d_i}^2}, \tag{4.39}$$

which shows the same trend as (4.28) with some additional dependence on d.

4.4 RESULTS AND DISCUSSION

This section presents three kinds of data. Firstly, numerical calculation of the CRLB for various checkerboard configurations was performed by evaluating and inverting (4.34). Secondly, Monte Carlo simulations were performed in which Gaussian noise was added to the true 2D points and the OpenCV `solvePnP` function [Bradski 2000] was used to calculate the pose in the same way as for a real checkerboard. These simulations are representative of real pose estimation implementations, assuming the corner detection noise distribution matches the model used here and there is no estimation bias. Finally, an experiment was performed in which a real checkerboard was imaged with a camera, its pose was estimated using a similar process, and the variances of the resulting pose estimates were compared to the numerically obtained CRLB.

Where not specified, the results presented in this section use $\sigma = 0.05\,\text{px}$, $f = 2952\,\text{px}$, $d = 0.12\,\text{m}$, $M = N = 2$, and $\beta = (0, 0, 1, 0, 0, 0)$. These values are chosen

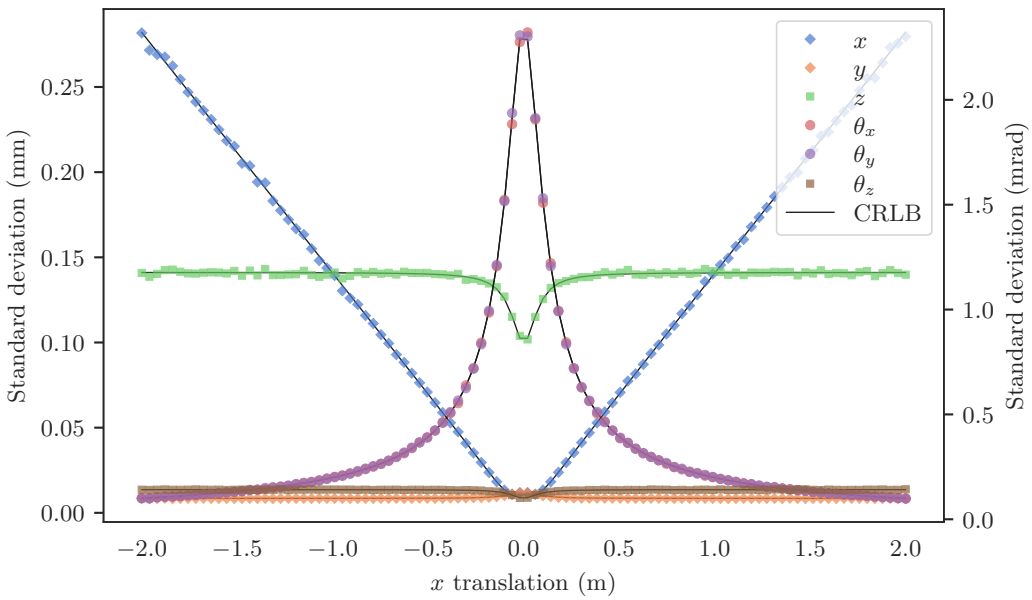


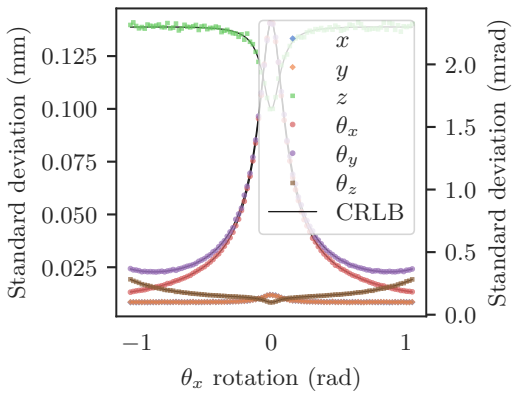
Figure 4.3 Simulation results and numerically obtained bound for a checkerboard moved along the x-axis, showing the dramatic variation with pose.

to match the real checkerboard used in the experiment. The figures show standard deviation rather than variance to make the trends more visible.

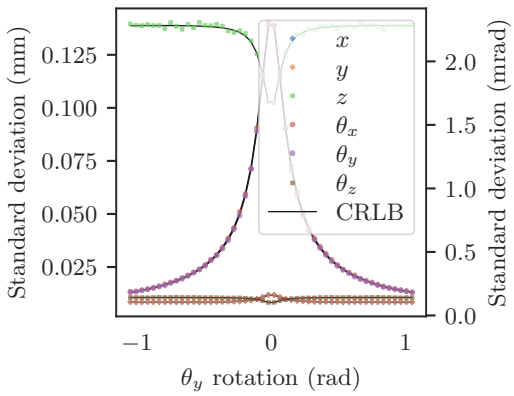
4.4.1 Numerical and simulation results

In Figures 4.3 to 4.6 and 4.8 to 4.11, simulation results and numerically obtained bounds are compared for various checkerboard configurations. Figures 4.7, 4.12 and 4.13 show the numerically obtained bounds only. The simulation results closely follow the CRLB in every scenario (even for the figures in which simulation results are omitted), which verifies that the model correctly predicts pose estimation precision.

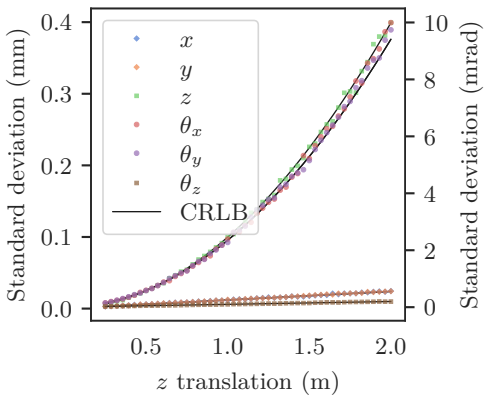
In Figures 4.3 and 4.4, the checkerboard is moved along or rotated around various axes. Setting $x = 0$ and varying y gives the same graph as Figure 4.3 but with x and y reversed. Figures 4.3, 4.4a and 4.4b show that roll and pitch estimation is imprecise



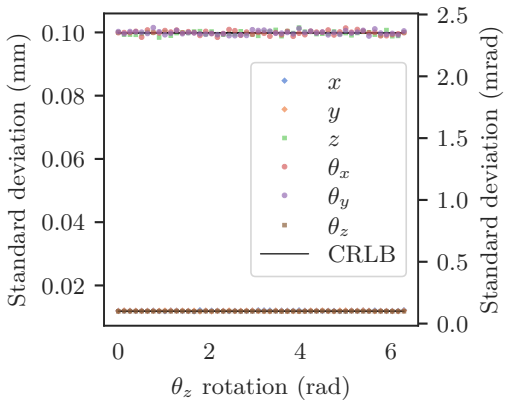
(a) Simulation results and numerically obtained bound for a checkerboard rotated around the x-axis.



(b) Simulation results and numerically obtained bound for a checkerboard rotated around the y-axis.



(c) Simulation results and numerically obtained bound for a checkerboard moved along the z-axis.



(d) Simulation results and numerically obtained bound for a checkerboard rotated around the z-axis.

Figure 4.4 Simulation results and numerically obtained bounds for a 2×2 checkerboard moved along or rotated around various axes.

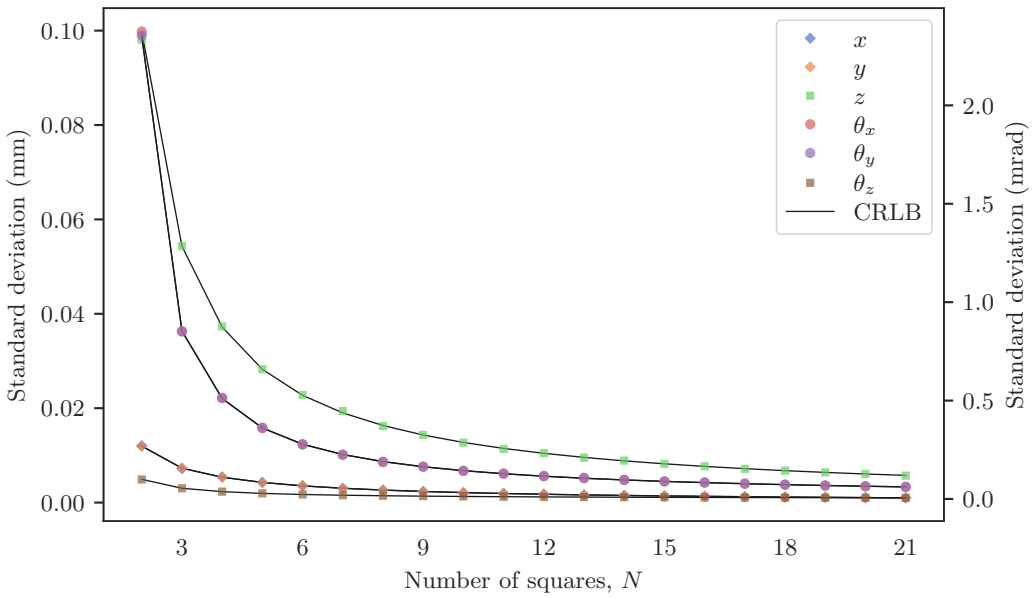


Figure 4.5 Simulation results and numerically obtained bounds for checkerboards of the same physical size with varying numbers of squares, from 2×2 to 21×21.

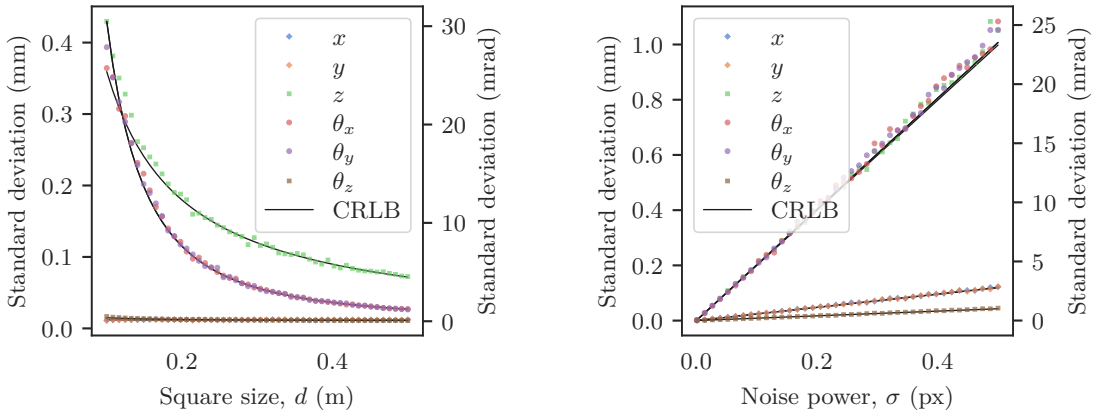
when the checkerboard is parallel to the image plane and near the centre of the image. This effect is well known in the context of fiducial markers [e.g., Tanaka et al. 2012]. Figure 4.4c shows that the estimation precision for all parameters (but particularly θ_x and θ_y) decreases rapidly as the distance from the camera increases. Figure 4.4d shows that rotations around the z-axis do not affect estimation precision (for a checkerboard parallel to the image plane).

In Figures 4.5 and 4.6, non-pose parameters are varied. Figures 4.5 and 4.6a shows that estimation precision increases as the number of corners or the checkerboard size increases, but with rapidly diminishing returns. Figure 4.6b simply illustrates estimation precision decreasing linearly as corner detection precision decreases.

In Figure 4.7, the checkerboard is rotated about the x-axis in two different positions, with the position and orientation parameters graphed separately, illustrating how the high variance at the centre of the image observed above decreases as the checkerboard moves away from the origin. This is further illustrated in Figures 4.8 to 4.11, which repeat the plots from Figures 4.3 to 4.6 but with the checkerboard positioned at $y = 0.5$ instead of $y = 0$, showing similar effects across most parameters.

Figure 4.12 illustrates the increase in estimation precision for a checkerboard of the same size with more squares, as in (4.37). Figure 4.13 shows how the estimation precision varies for each orientation parameter depending on which rotation parametrisation is used ($z - x - y$, $z - y - x$ or $y - z - x$).

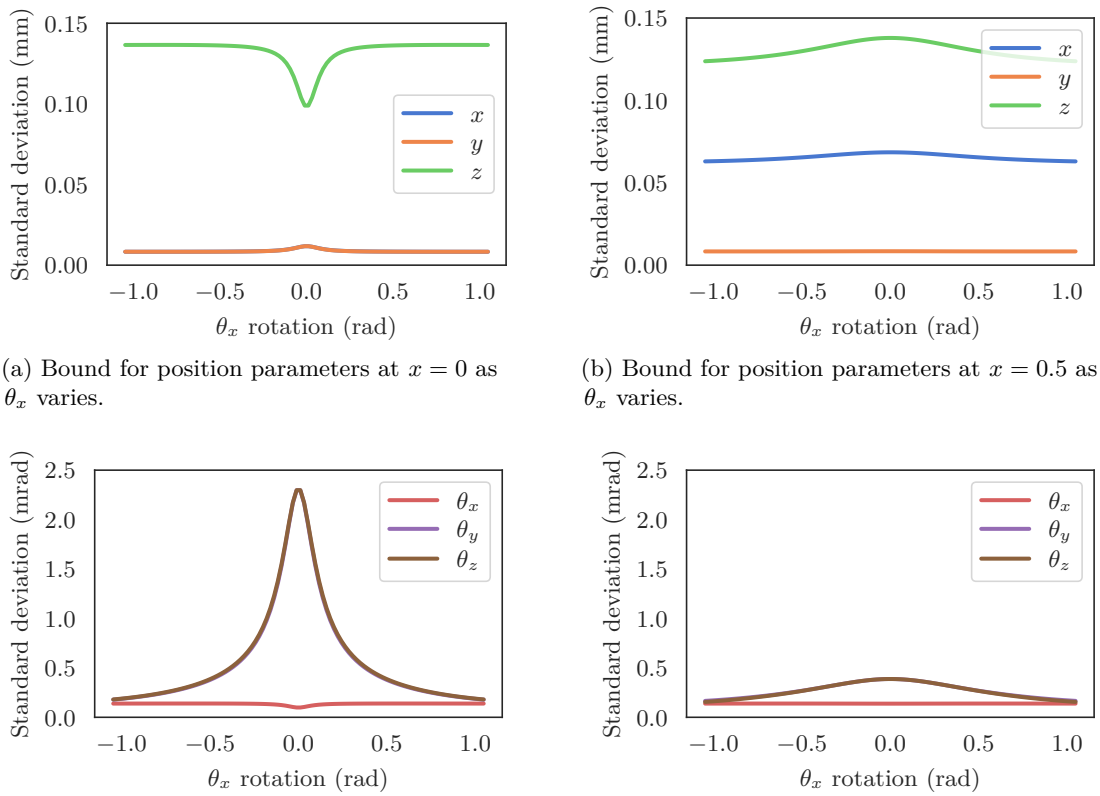
Another property to consider is the covariance, which is more intuitively expressed as the correlation between parameters in each estimate. The correlation coefficient of



(a) Simulation results and numerically obtained bounds for checkerboards with varying square sizes, d.

(b) Simulation results and numerically obtained bounds with varying corner estimation noise standard deviations, σ.

Figure 4.6 Simulation results and numerically obtained bounds for varying corner estimation noise standard deviations and checkerboard square sizes.



(a) Bound for position parameters at $x = 0$ as θ_x varies.

(b) Bound for position parameters at $x = 0.5$ as θ_x varies.

(c) Bound for orientation parameters at $x = 0$ as θ_x varies.

(d) Bound for orientation parameters at $x = 0.5$ as θ_x varies.

Figure 4.7 Numerically obtained bounds for a 2×2 checkerboard rotating around the x-axis at $x = 0$ and $x = 0.5$, with (x, y, z) and $(\theta_x, \theta_y, \theta_z)$ on separate axes.

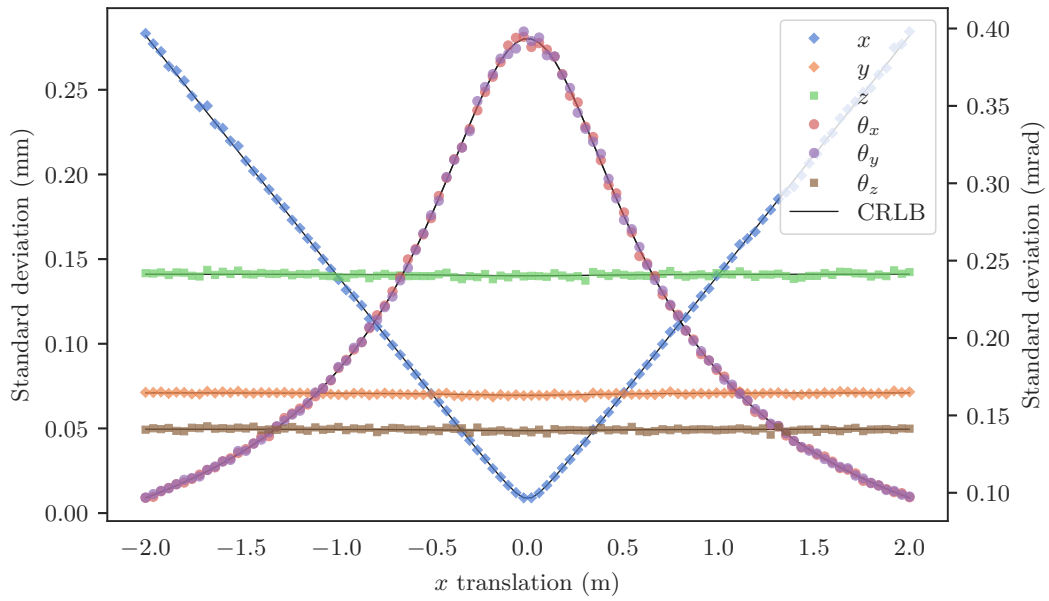


Figure 4.8 Simulation results and numerically obtained bound for a checkerboard moved along the x-axis at $y = 0.5$. Note that the axis scale is not the same as in Figure 4.3.

two random variables X and Y is defined as

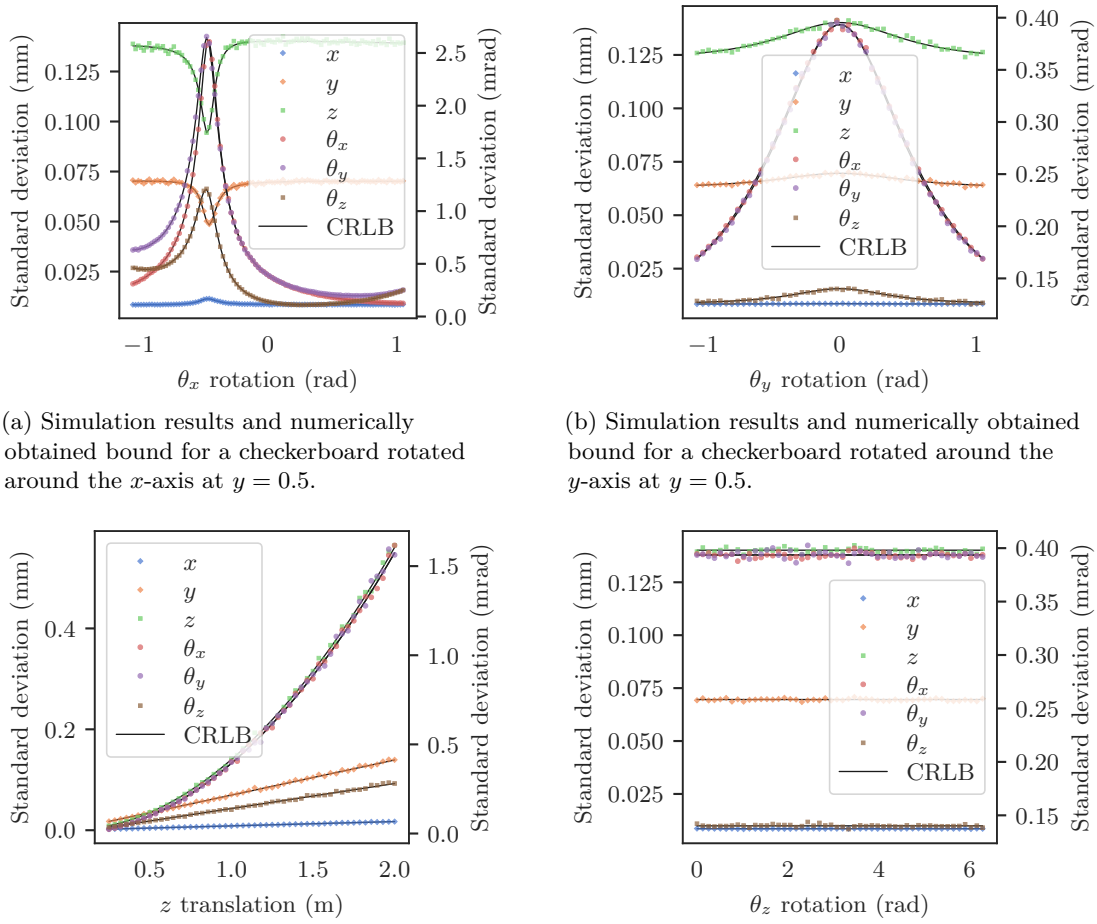
$$\mathrm{Corr}\,(X, Y) = \frac{\mathrm{Cov}\,(X, Y)}{\sqrt{\mathrm{Var}\,(X)\,\mathrm{Var}\,(Y)}}. \tag{4.40}$$

The correlation coefficient is 0 for uncorrelated variables, 1 for perfectly positively correlated variables, and −1 for perfectly negatively correlated variables. The correlation coefficient matrix is defined as

$$[\mathbf{C}]_{ij} = \mathrm{Corr}\,(\beta_i, \beta_j)\,. \tag{4.41}$$

Since the CRLB gives a lower bound for each element of the covariance matrix [Watanabe 2009, chapter 1], it can be used to calculate the correlation coefficient matrix for a minimum-variance unbiased estimator using isotropic, homogeneous, independent corner position estimates. The correlation coefficient matrix for $d = \frac{1}{3}$ m and $x = y = 0$ is

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -0.71 \\ 0 & 1 & 0 & 0 & 0.71 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0.71 & 0 & 0 & 1 & 0 \\ -0.71 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.42}$$



(a) Simulation results and numerically obtained bound for a checkerboard rotated around the x-axis at $y = 0.5$.

(b) Simulation results and numerically obtained bound for a checkerboard rotated around the y-axis at $y = 0.5$.

(c) Simulation results and numerically obtained bound for a checkerboard moved along the z-axis at $y = 0.5$.

(d) Simulation results and numerically obtained bound for a checkerboard rotated around the z-axis at $y = 0.5$.

Figure 4.9 Simulation results and numerically obtained bounds for a 2×2 checkerboard moved along or rotated around various axes at $y = 0.5$. Note that the axis scales are not the same as in Figure 4.4.

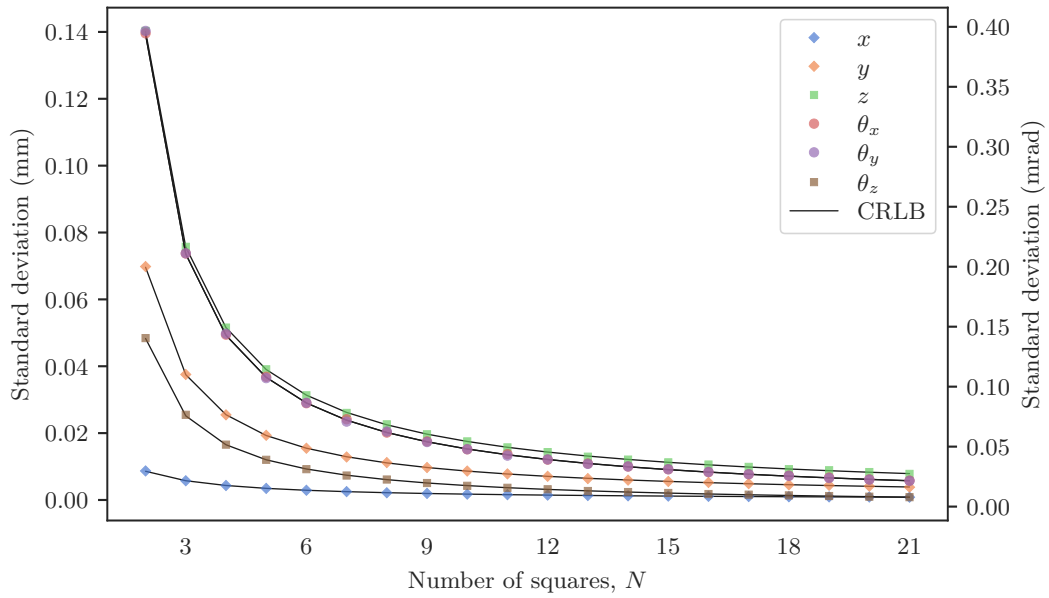
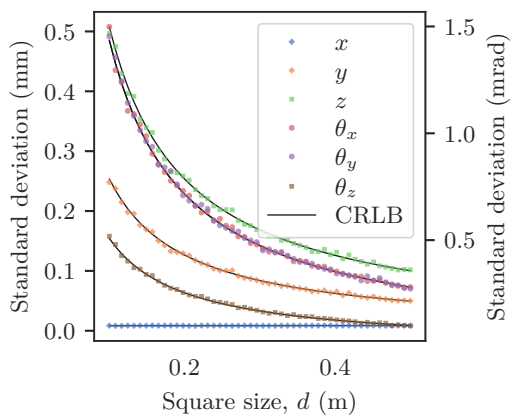
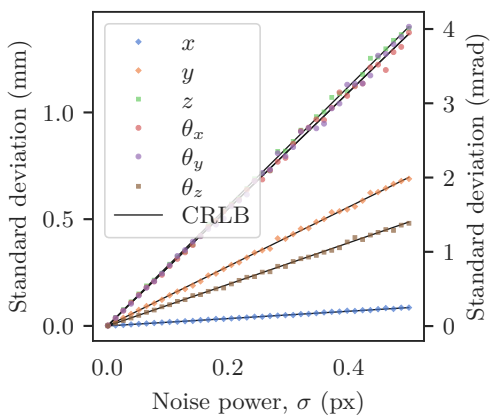


Figure 4.10 Simulation results and numerically obtained bounds for checkerboards with varying numbers of squares, from 2×2 to 21×21, at $y = 0.5$. Note that the axis scale is not the same as in Figure 4.5.

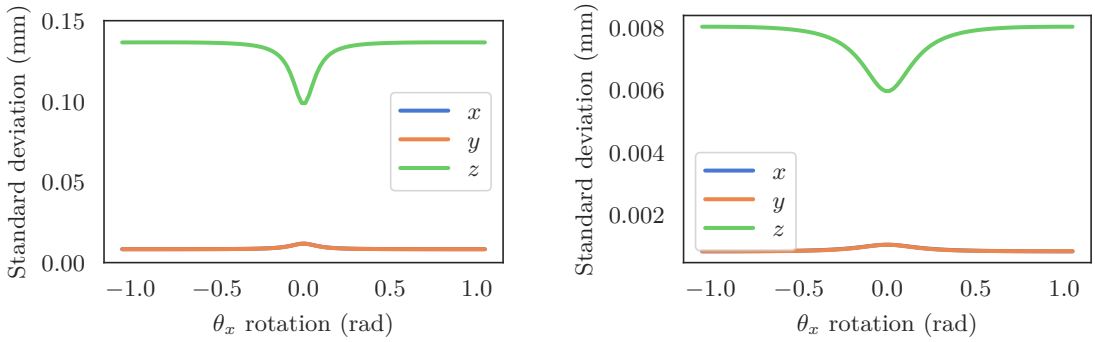


(a) Simulation results and numerically obtained bounds for checkerboards with varying square sizes, d, at $y = 0.5$.



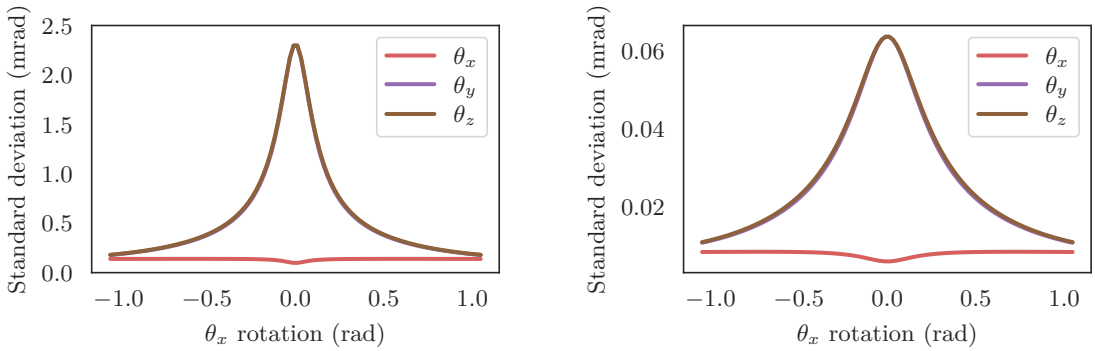
(b) Simulation results and numerically obtained bounds with varying corner estimation noise standard deviations, σ, at $y = 0.5$.

Figure 4.11 Simulation results and numerically obtained bounds for varying corner estimation noise standard deviations and checkerboard square sizes, at $y = 0.5$. Note that the axis scales are not the same as in Figure 4.6.



(a) Bound for position parameters for a 2×2 checkerboard as θ_x varies. This is the same as Figure 4.7a.

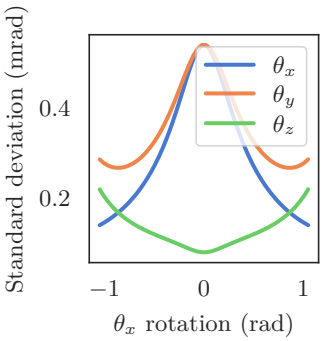
(b) Bound for position parameters for a 20×20 checkerboard of the same size as θ_x varies.



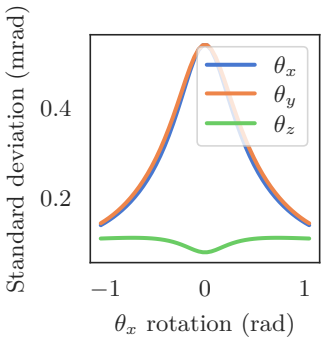
(c) Bound for position parameters for a 20×2 checkerboard as θ_x varies. This is the same as Figure 4.7c.

(d) Bound for orientation parameters for a 20×20 checkerboard of the same size as θ_x varies.

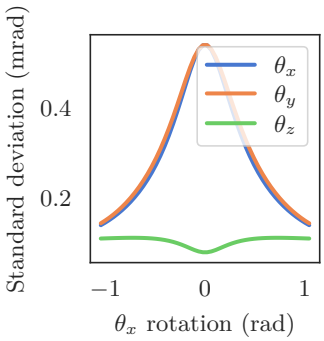
Figure 4.12 Numerically obtained bounds for 2×2 and 20×20 checkerboards of the same size rotating around the x-axis, with (x, y, z) and $(\theta_x, \theta_y, \theta_z)$ on separate axes. Note that the axis scales are not the same as in Figure 4.7.



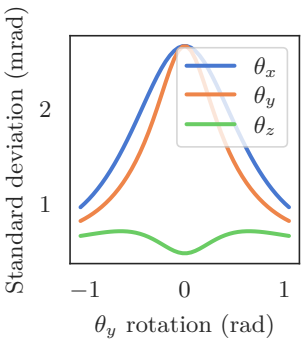
(a) Bound for the $z-x-y$ parametrisation as the checkerboard rotates around the x-axis.



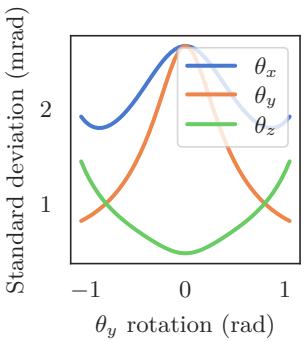
(b) Bound for the $z-y-x$ parametrisation as the checkerboard rotates around the x-axis.



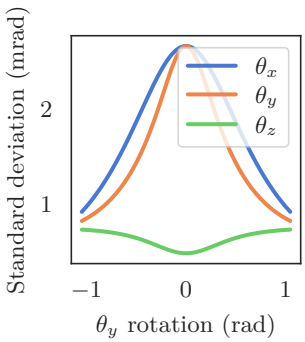
(c) Bound for the $y-z-x$ parametrisation as the checkerboard rotates around the x-axis.



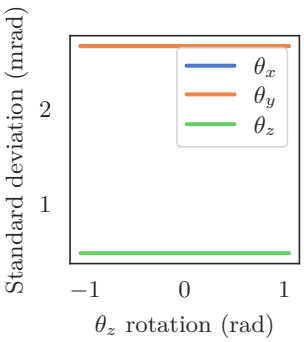
(d) Bound for the $z-x-y$ parametrisation as the checkerboard rotates around the y-axis.



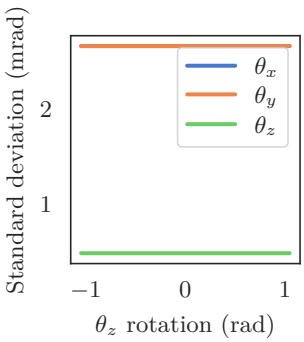
(e) Bound for the $z-y-x$ parametrisation as the checkerboard rotates around the y-axis.



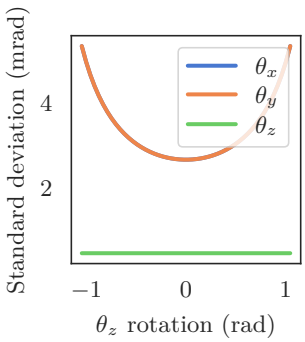
(f) Bound for the $y-z-x$ parametrisation as the checkerboard rotates around the y-axis.



(g) Bound for the $z-x-y$ parametrisation as the checkerboard rotates around the z-axis.



(h) Bound for the $z-y-x$ parametrisation as the checkerboard rotates around the z-axis.



(i) Bound for the $y-z-x$ parametrisation as the checkerboard rotates around the z-axis.

Figure 4.13 Numerically obtained bounds for each rotation parametrisation as a checkerboard is rotated around each axis, with $f = d = 1$ to exaggerate the differences.

With $x = y = 0.1$ it is

$$\mathbf{C} = \begin{bmatrix} 1 & -0.02 & 0.15 & 0.21 & -0.12 & -0.54 \\ -0.02 & 1 & 0.15 & -0.21 & 0.54 & 0.12 \\ 0.15 & 0.15 & 1 & 0 & -0.32 & 0.32 \\ 0.21 & -0.21 & 0 & 1 & -0.32 & -0.32 \\ -0.12 & 0.54 & -0.32 & -0.32 & 1 & 0 \\ -0.54 & 0.12 & 0.32 & -0.32 & 0 & 1 \end{bmatrix}, \tag{4.43}$$

and with $x = y = 0.5$ it is

$$\mathbf{C} = \begin{bmatrix} 1 & 0.87 & 0.94 & 0.05 & -0.48 & 0.37 \\ 0.87 & 1 & 0.94 & -0.05 & -0.37 & 0.48 \\ 0.94 & 0.94 & 1 & 0 & -0.49 & 0.49 \\ 0.05 & -0.05 & 0 & 1 & -0.49 & -0.49 \\ -0.48 & -0.37 & -0.49 & -0.49 & 1 & 0 \\ 0.37 & 0.48 & 0.49 & -0.49 & 0 & 1 \end{bmatrix}. \tag{4.44}$$

These correlation coefficient matrices are representative of the general case: the correlation between the estimate of each parameter is high.[7] Although not shown here, $\mathrm{Corr}\,(z, \theta_z)$ and $\mathrm{Corr}\,(\theta_x, \theta_y)$ increase as θ_x, θ_y or θ_z increase. From this it can be concluded that most of the parameters are correlated with each other to some extent, depending on the marker position. Thus, for an application like sensor fusion that uses a full covariance matrix, the proposed model is more accurate than a model of estimate variance that assumes zero covariance.

4.4.2 Real data

A Point Grey (now Flir) Grasshopper3 GS3-U3-41C6NIR-C global shutter machine vision camera was mounted on a tripod. An 8×5 checkerboard with 35 mm squares, professionally printed on aluminium composite board, was mounted on a whiteboard and lit from the side with an overhead projector. 150 images with a resolution of 1024 px $\times$ 768 px were taken at a series of 15 distances, moving from the distance at which the checkerboard filled the image (as shown in Figure 4.14) to the distance at which the squares began to be hard to resolve (as shown in Figure 4.15) in approximately 0.1 m steps, with the marker approximately (but not precisely) in the centre of the frame

[7] A full analysis of the correlation between parameters is beyond the scope of this chapter, but would be useful future work.

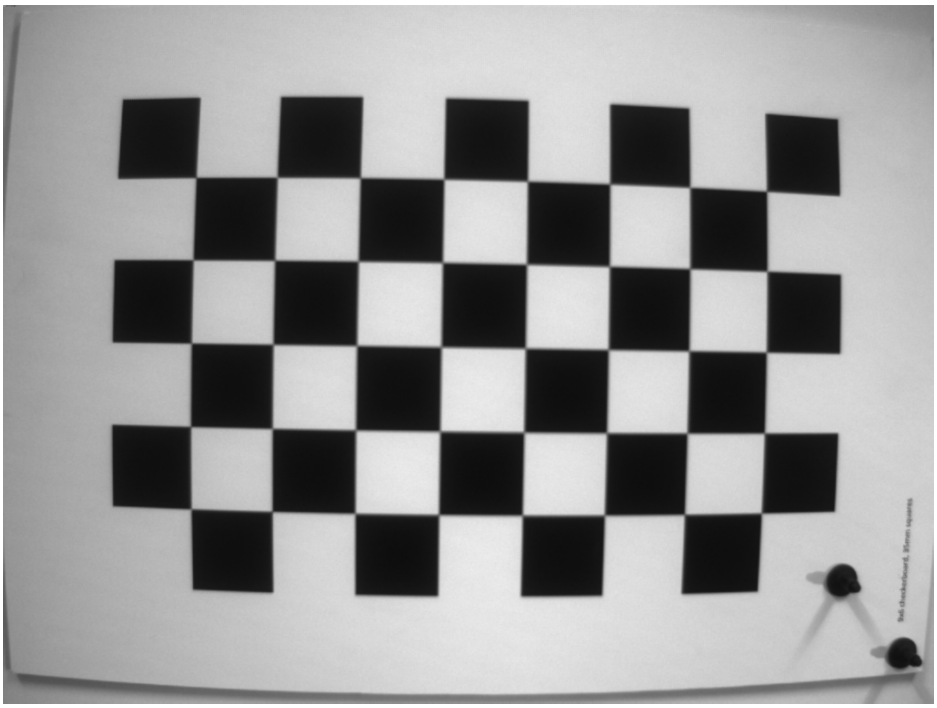


Figure 4.14 An image taken from the closest distance (approximately 0.5 m), where the checkerboard fills the image but is still detectable (the checkerboard corner detection algorithm fails when the checkerboard is any closer to the edges of the image). Notice how the edges are slightly out of focus; this distance is likely too short for the lens. The two black objects on the lower right of the checkerboard are adhesive bases for motion capture markers (not used here).

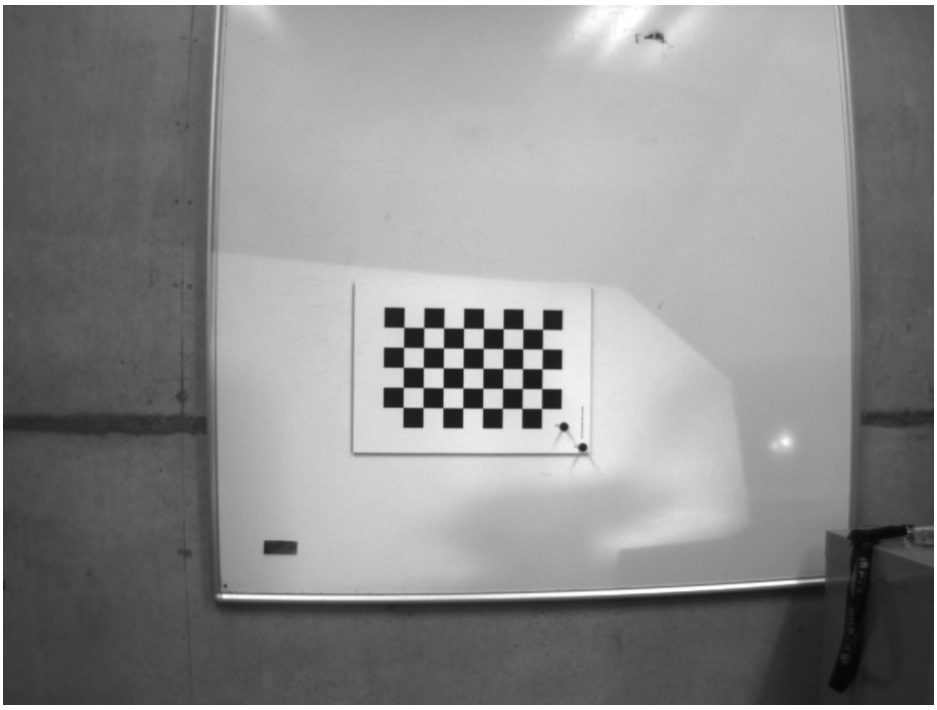


Figure 4.15 An image taken from the furthest distance (approximately 2.1 m), just before the checkerboard squares become too small for checkerboard detection to be reliable. The lighting patterns from the overhead projector and the room's ceiling lights are visible.

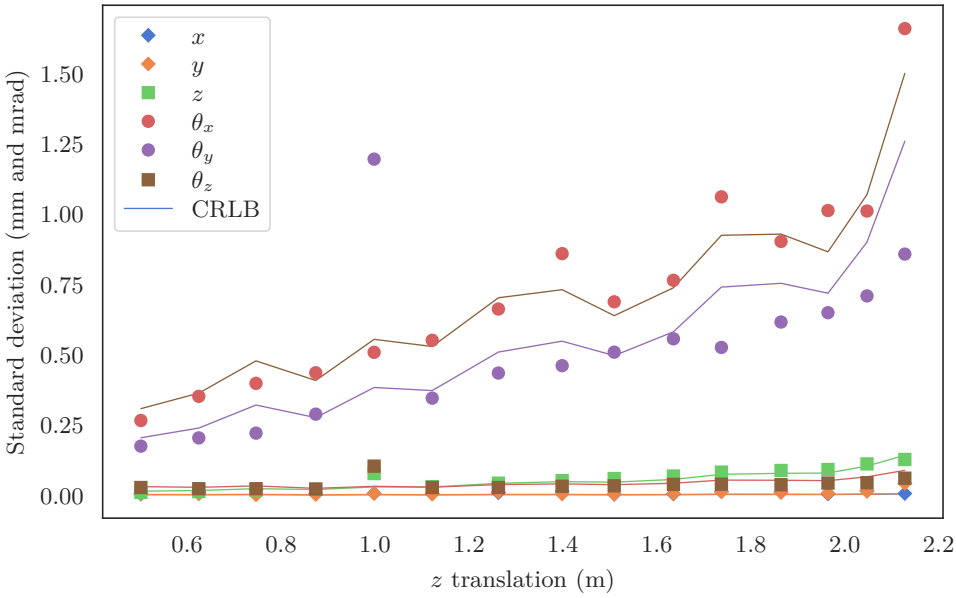


Figure 4.16 The standard deviation of each parameter of the estimated poses from 150 images of checkerboard at 15 distances, compared to the bound obtained by evaluating (4.34) using the mean of the estimated poses and inverting. In this graph, it is expected that the bound does not form a smooth curve as in the simulations, since the camera is not moving in a perfectly straight line (rather, the camera parameters at each point are all approximate, as described in Section 4.4.2).

at each point. Each image was rectified,[8] then for each image, the checkerboard corners were detected using the OpenCV function `findChessboardCorners` and refined to sub-pixel precision using `cornerSubpix`, then the checkerboard pose was estimated in the same way as in the simulation.

In order to evaluate the CRLB, the mean of all the standard deviations of each estimated corner at each distance was used as the corner detection error noise standard deviation σ,[9] the camera focal length obtained from camera calibration was used as f, and the mean of the estimated poses at each distance was used as the pose for that distance, then the FIM was evaluated and inverted as in the simulations. In Figure 4.16, the square root of this bound is compared to the standard deviation of the pose estimates at each distance. The variances of the estimates generally lie near the CRLB, although not above it as they should (due either to bias or poor modelling of corner detection noise), and there are some significant spikes. Given that the OpenCV implementation is not necessarily unbiased and the corner detection noise is not entirely uncorrelated (as discussed in Chapters 5 and 6), this is a good result. Overall, the model successfully predicts the best-case precision of checkerboard pose estimation from real images.

[8] Note that rectification introduces a non-linear blurring to the image. The effect this has on corner detection is examined in Section 5.3.1.

[9] This is potentially problematic: in the close-up images the edges appear blurred (so those images likely have higher corner detection error), and since rectified images are used, the corners nearer the edges of the image will be less sharp than those in the centre. Nevertheless, the results as presented are sufficient to demonstrate that the CRLB derived here is a reasonable fit for real data. The details of corner detection error distribution are explored in greater detail in Chapters 5 and 6.

4.4.3 Discussion

This work aimed to investigate how each characteristic of a fiducial marker/camera system affects pose estimation performance, in the context of the fiducial marker system from Chapter 3. The relationship between minimum pose estimation variance and the checkerboard configuration, checkerboard pose, camera focal length, and corner detection noise variance derived here is a powerful tool for this investigation. It shows that increasing the number of corners of a checkerboard, which is roughly equivalent to the approach from Chapter 3, gives rapidly diminishing returns. Increasing the size of the checkerboard or decreasing the distance to the checkerboard has a similar effect at the cost of taking up more of the image; a significant issue when recording ground truth for vision algorithms. Reducing corner detection noise variance directly reduces pose estimate variance. It is difficult to improve on the precision of modern checkerboard corner detection algorithms, but pose estimation using other features like circles [Benligiray et al. 2019, Bergamasco et al. 2011, Lightbody et al. 2017, Xu and Dudek 2011] (and possibly the continuous circular pattern from Chapter 3) may still offer some potential. Another approach is to use markers augmented with lenses or gratings to produce Moiré patterns [Armstrong et al. 2007, Banks et al. 2019, Banks 2020, Tanaka et al. 2012, 2015, 2017]. These markers vary their appearance depending on viewing angle, allowing their orientation to be estimated more precisely. The final issue is that the precision to which each pose parameter can be estimated for a single checkerboard varies substantially depending on the checkerboard's pose (as shown in, for example, Figures 4.3 and 4.7). This is a serious issue for a system where the aim is to estimate the pose of a camera with bounded precision using a single view of a distant marker.

One implication of these findings is that measuring the pose estimation performance of a fiducial marker, or comparing the performance of two fiducial markers, must be done with great care. If a marker is evaluated in one pose, or in a series of poses where few parameters change (as in Section 3.5.1), its performance will not be representative of the general case. Consider, for example, the dramatic difference in performance resulting from the small translation between Figure 4.7c and Figure 4.7d. Furthermore, pose estimation performance depends on the characteristics of the camera, lens, and lighting used.[10] In practice, any precision result given in a paper is so specific to the authors' situation that comparisons to results from other papers are of questionable validity.

Common advice for using checkerboards for camera calibration is to use a large calibration target with as many features as can be imaged reliably, to use good lighting and an appropriately focused camera, and to collect images with a variety of poses [Wilm

[10] The focal length of the lens affects the variance of the pose estimate directly. The other characteristics (camera resolution, sensor noise, lens resolution, and lighting quality) affect corner detection noise variance.

2018]. This follows from the findings here: estimation variance decreases with checkerboard size and number of squares and increases with corner detection noise variance, and different poses have very different variance characteristics. Note that many factors are linked; for example, the camera field of view, calibration target size, number of squares, and maximum working distance all affect the square size in pixels. In particular, choosing parameters to achieve a minimum square size in pixels implicitly depends on the window size for the corner refinement algorithm, which can impact corner estimation noise (which is modelled here as independent), which is also dependent on image sharpness and noise, which are dependent on the camera, lens, and lighting as mentioned above. Note also that the optimisation problem involved in camera calibration is a superset of the problem examined here,[11] so this analysis may not translate directly; a full model for camera calibration precision would be useful future work.

Knowing the precision of a pose estimate is important for many applications, for example, sensor fusion and active SLAM. The approach in this chapter has the advantage of giving a symbolic result, but the disadvantage of being computationally intensive and specific to a particular object structure. For real-time applications, numerical approaches which use point correspondences directly to calculate a single covariance matrix [e.g., Kanatani and Ohta 2001, Zhang and Scaramuzza 2019] are more appropriate.

4.5 CONCLUSION

In this chapter, an analytic model was derived for the CRLB of pose estimation using a checkerboard (or fiducial marker) pose estimator. The model gives a lower bound for the variance (and covariance) of the estimate, which effectively predicts the precision of the most accurate pose estimate from a given set of data. Both a Monte Carlo simulation and real data validate this model. The model can both predict estimator precision before data is available and evaluate the performance of a real estimator on real data compared to the theoretical precision limit. One significant finding is that generalising measurements of pose estimation precision is difficult: results from different cameras are not directly comparable, and the performance for a marker in one pose does not trivially predict the performance for another pose. This approach for predicting and evaluating checkerboard pose estimation precision has not been considered in prior research.

[11] In camera calibration, the parameters for the optimisation problem are the pose of the calibration target in each frame along with the parameters of the camera model.

Chapter 5

CORNER DETECTION ERROR MODELLING

In Chapter 4, a theoretical lower bound was derived for checkerboard pose estimation error and used to investigate its dependence on various parameters. This bound assumed that the error from corner detection could be modelled as zero-mean, independent, and identically distributed Gaussian random variables. The aim of this chapter, which is based on published work [Edwards et al. 2018], is to evaluate the suitability of that model.

5.1 INTRODUCTION

In computer vision, the terms *corner*, *feature*, and *interest point* are used interchangeably to refer to the intersection of two edges, or a point with low self-similarity, or one of many similar concepts.[1] In this chapter, *corner detection* refers to the process of finding the approximate location of checkerboard corners in an image using Suzuki and Abe's [1985] contour detection algorithm, as implemented in the `findChessboardCorners` function in version 4.4 of the OpenCV library [Bradski 2000], then refining each corner's location to sub-pixel precision using the Förstner operator [Förstner and Gülch 1987], as implemented in OpenCV's `cornerSubPix` function. The difference between the result of this process and the corner's true location is the *corner detection error*.

The general approach for sub-pixel corner refinement is to take a window around a coarse estimate and then either interpolate between pixels in the image to find the saddle point [Lucchese and Mitra 2002] or construct an estimate directly as a function of the pixel values [Chen and Zhang 2005].

There has been some previous work on sources of error for checkerboard corner detection. Mallon and Whelan [2007] investigated two sub-pixel detection strategies for checkerboard patterns and two for circle patterns, evaluating their robustness to perspective distortion, lens distortion, and blur. They concluded that lens and per-

[1] This can lead to phrasing which is confusing out of context, for example using a corner detector to detect spots.

spective distortion were the most significant source of bias for most methods.[2] Strobl and Hirzinger [2008] pointed out that off-the-shelf printing equipment generally prints with slightly incorrect scale in each direction and proposed a camera calibration routine that estimated scale and aspect ratio (and improved accuracy). Albarelli et al. [2010] went further, finding that fully estimating the checkerboard geometry gave further improvements in calibration accuracy. Ha et al. [2017] presented a camera calibration method using a triangle grid, where the corner detection process was shown to be dramatically more robust to blur, image noise, and perspective distortion than traditional checkerboards. Duda and Frese [2018] presented a new method for checkerboard corner detection and showed it to be consistently more accurate than the OpenCV method. Hannemose et al. [2019] presented a camera calibration method that uses a similar rendering and optimisation process to the one in Chapter 3 and showed it to be significantly more accurate than the OpenCV method and slightly more accurate than the method from Ha et al. [2017]. Schöps et al. [2020] presented a camera calibration method using a target with Siemens star-like features (as shown in Figure 5.1), including a corner detector that was specifically designed to minimise bias. These new targets and algorithms offer significant advantages for camera calibration, and should be preferred over traditional methods when precision is required. However, checkerboards are still widely used in practice, and the underlying sub-pixel corner refinement method is common in other applications as well, hence the focus of this chapter.

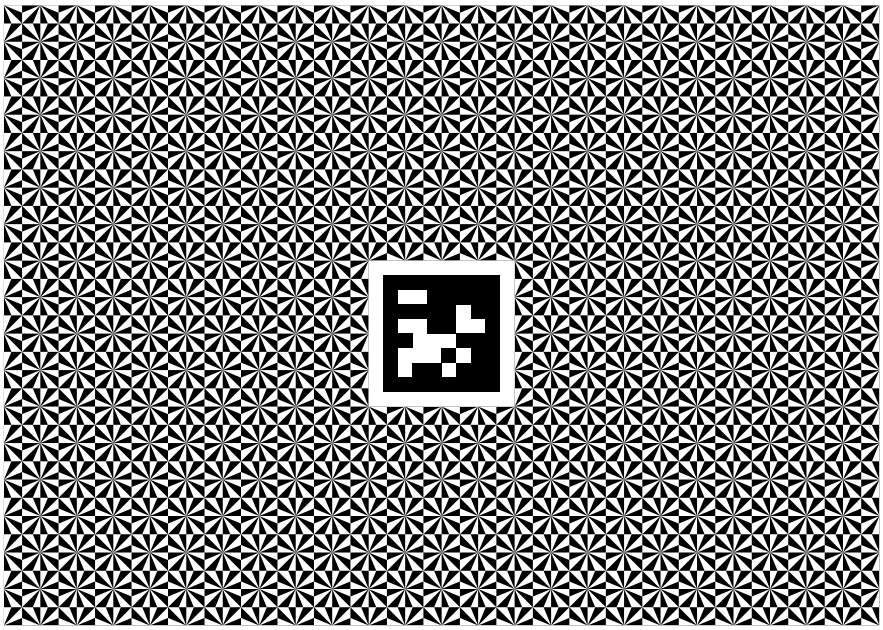


Figure 5.1 An example of the camera calibration target introduced by Schöps et al. [2020] that uses Siemens star-like features instead of traditional checkerboard corners.

[2] Note that Mallon and Whelan [2007] found estimation bias increases slightly with blur kernel size (which they described as "remaining relatively constant"), while the results in this chapter show that the bias in a blurred image *decreases* with blur kernel size.

The rest of this chapter is structured as follows: Section 5.2 presents background theory, Section 5.3 outlines the methods used for simulation and data collection and presents experimental results, and Section 5.5 summarises the results.

5.2 BACKGROUND

Förstner and Gülch [1987] proposed a method for precisely locating corners that calculates the intersection of the edges within an image window using a sum of pixel gradients. This was an improvement on the state of the art at the time (Moravec's [1980] corner detection algorithm, one of the earliest such algorithms), but was followed by the Harris [Harris and Stephens 1988] and Shi-Tomasi [Shi and Tomasi 1994] corner detectors, which are still widely used. The lasting contribution of Förstner and Gülch [1987] is a method for calculating the location of a corner with sub-pixel precision (see Appendix A).

An intuitive understanding of this method can be reached by considering the pixels in an image window containing a checkerboard corner. Each pixel falls either on an edge or in a flat-colour region. For a pixel in a flat-colour region, the image gradient is approximately zero. In a window centred exactly on a corner, the edge is a straight line through the pixel toward the centre and the image gradient at the pixel is perpendicular to that line, so their dot product is zero. Otherwise, their dot product is non-zero.

This can be formalised as an error function. For an $M \times N$ image window $\mathbf{I}$ with a corner at $\mathbf{c} = \left(c_x, c_y\right)^{\mathrm{T}}$, the dot product of a vector from a point $\mathbf{p}^{(i)}$ to the centre $\mathbf{c}$ with the image gradient $\mathbf{D}^{(i)}$ at the point is

$$e^{(i)} = \mathbf{D}^{(i)\mathrm{T}} \left(\mathbf{c} - \mathbf{p}^{(i)}\right), \tag{5.1}$$

where $e^{(i)}$ is the error for the ith pixel. Note that the ith pixel in the window has pixel indices (m, n) and so

$$\begin{aligned} \mathbf{p}^{(i)} &= \mathbf{p}^{(m,n)} \\ &= \begin{pmatrix} p_x^{(m)} \\ p_y^{(n)} \end{pmatrix}. \end{aligned} \tag{5.2}$$

The gradient images are calculated using the central difference:

$$\begin{aligned} D_x[m,n] &= \mathbf{I}[m, n+1] - \mathbf{I}[m, n-1], \\ D_y[m,n] &= \mathbf{I}[m+1, n] - \mathbf{I}[m-1, n], \\ \mathbf{D}[m,n] &= \begin{pmatrix} D_x[m,n] \\ D_y[m,n] \end{pmatrix}. \end{aligned} \tag{5.3}$$

The error, (5.1), can be expanded as

$$\begin{aligned} e^{(i)} &= \mathbf{D}_i^{\mathrm{T}} \left(\mathbf{c} - \mathbf{p}^{(i)} \right) \\ &= D_x^{(i)} c_x + D_y^{(i)} c_y - D_x^{(i)} p_x^{(i)} - D_y^{(i)} p_x^{(i)}, \end{aligned} \tag{5.4}$$

and so the error vector can be expressed as

$$\mathbf{e} = \begin{pmatrix} e^{(1)} \\ e^{(2)} \\ \vdots \\ e^{(MN)} \end{pmatrix} = - \begin{pmatrix} D_x^{(1)} p_x^{(1)} + D_y^{(1)} p_y^{(1)} \\ D_x^{(2)} p_x^{(2)} + D_y^{(2)} p_y^{(2)} \\ \vdots \\ D_x^{(M)} p_x^{(M)} + D_y^{(N)} p_y^{(N)} \end{pmatrix} + \begin{bmatrix} D_x^{(1)} & D_y^{(1)} \\ D_x^{(2)} & D_y^{(2)} \\ \vdots & \vdots \\ D_x^{(M)} & D_y^{(N)} \end{bmatrix} \begin{pmatrix} c_x \\ c_y \end{pmatrix}, \tag{5.5}$$

or in vector form,

$$\mathbf{e} = \mathbf{u} + \mathbf{A}\mathbf{c}. \tag{5.6}$$

Minimising this error function finds an estimate $\hat{\mathbf{c}}$ of the corner. Since (5.6) is linear in the corner position, the least-squares estimate is given by[3]

$$\hat{\mathbf{c}} = \left(\mathbf{A}^{\mathrm{T}} \mathbf{A} \right)^{-1} \mathbf{A}^{\mathrm{T}} \mathbf{u}. \tag{5.7}$$

This can be expressed as

$$\hat{\mathbf{c}} = \mathbf{G}^{-1} \mathbf{b}, \tag{5.8}$$

where

$$\mathbf{G} = \mathbf{A}^{\mathrm{T}} \mathbf{A} \tag{5.9}$$

and

$$\mathbf{b} = \mathbf{A}^{\mathrm{T}} \mathbf{u}. \tag{5.10}$$

The $\mathbf{G}$ matrix can be expressed as

$$\mathbf{G} = \begin{bmatrix} \sum_i \left(D_x^{(i)} \right)^2 & \sum_i D_x^{(i)} D_y^{(i)} \\ \sum_i D_x^{(i)} D_y^{(i)} & \sum_i \left(D_y^{(i)} \right)^2 \end{bmatrix}. \tag{5.11}$$

Similarly, the $\mathbf{b}$ vector can be expressed as

$$\mathbf{b} = \begin{bmatrix} \sum_i \left(D_x^{(i)} \right)^2 p_x^{(i)} + \sum_i D_x^{(i)} D_y^{(i)} p_y^{(i)} \\ \sum_i D_x^{(i)} D_y^{(i)} p_x^{(i)} + \sum_i \left(D_y^{(i)} \right)^2 p_y^{(i)} \end{bmatrix}. \tag{5.12}$$

[3] Note that with this method it is assumed that the residuals are i.i.d. Given that the noise power for each pixel depends on its intensity (as discussed in Section 6.3.1), weighted least squares may give slightly better results.

Thus the estimate of c_x is

$$\hat{c_x} = \frac{\sum_i \left(D_y^{(i)}\right)^2 \left(\sum_i \left(D_x^{(i)}\right)^2 p_x^{(i)} + \sum_i D_x^{(i)} D_y^{(i)} p_y^{(i)}\right) - \sum_i D_x^{(i)} D_y^{(i)} \left(\sum_i D_x^{(i)} D_y^{(i)} p_x^{(i)} + \sum_i \left(D_y^{(i)}\right)^2 p_y^{(i)}\right)}{\sum_i \left(D_x^{(i)}\right)^2 \sum_i \left(D_y^{(i)}\right)^2 - \left(\sum_i D_x^{(i)} D_y^{(i)}\right)^2}. \tag{5.13}$$

The estimate for c_y is similar.

The OpenCV sub-pixel corner refinement implementation (`cornerSubPix`) uses Förstner's method to estimate the corner location then uses bilinear interpolation to get a window centred on the new estimate, repeating until the estimate converges. This is true for all OpenCV versions dating back to at least 2002, beyond which the project history is lost.[4]

5.3 METHOD AND RESULTS

Multiple approaches were used to examine various characteristics of the distribution of checkerboard corner detection error. Real images of checkerboards in various configurations were collected and used to analyse the distribution of corner detection error in practice. Monte Carlo simulations of estimating the location of a checkerboard corner in a noisy image were performed for a range of true corner locations in order to investigate the effect of small changes in corner location.

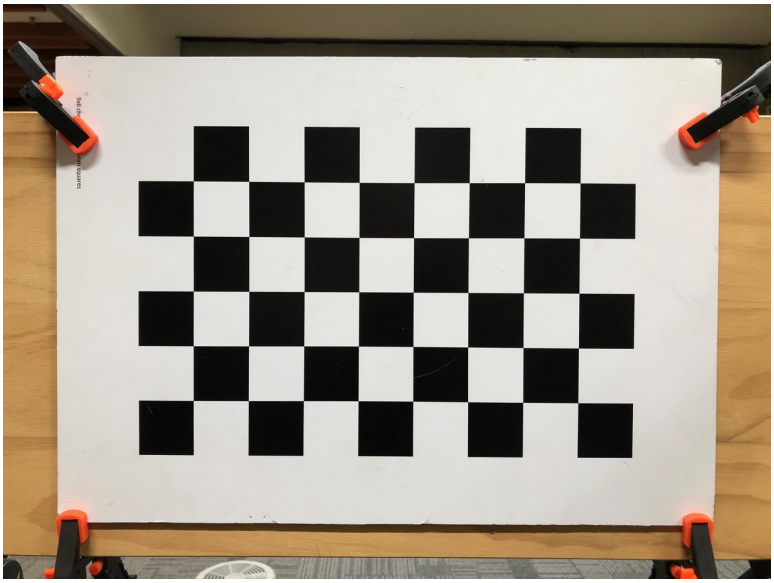


Figure 5.2 The checkerboard used in data collection, shown as configured for the follow-up study in Chapter 6.

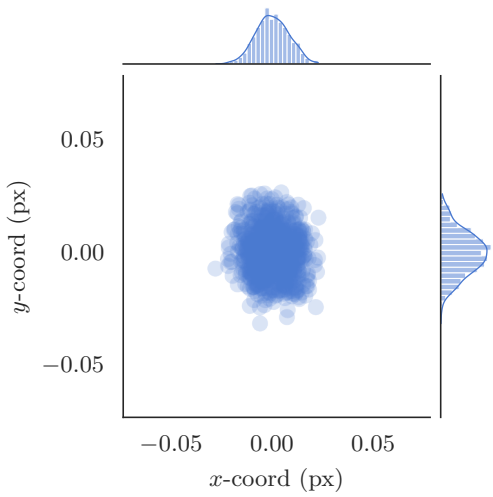
[4] The earliest version remaining online is at `https://github.com/opencv/opencv_attic/blob/27e20eea2591744fa37317faeb3783794784862c/opencv/src/cv/cvcornersubpix.cpp`. The original implementation was likely by Jean-Yves Bouguet, as he worked at Intel at the time OpenCV was first released, was credited for camera calibration [Intel Corporation 2001], and already had an implementation in his MATLAB camera calibration toolkit [Bouguet 1996].

5.3.1 Real data

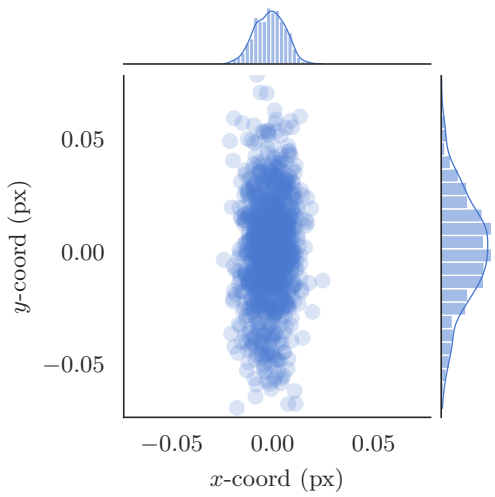
An 8×5 checkerboard pattern with 35 mm squares, offset-printed on aluminium composite board (as shown in Figure 5.2), was fixed to a wall and lit with diffused LED panels (as shown in Figure 5.3). Initially, a Point Grey (now Flir) Bumblebee2 BB2-08S2C global shutter machine vision camera on a tripod was used, positioned such that the checkerboard was centred and parallel to the image plane with a precision of 1 cm and 3 degrees. The camera was used to capture 1000 8-bit colour images with 1024 px $\times$ 768 px resolution, which were rectified using the Point Grey factory calibration. The OpenCV `findChessboardCorners` function was used to estimate the location of each checkerboard corner in each image, and the OpenCV `cornerSubPix` function was used to refine each corner's location to sub-pixel precision. The true corner locations are unknown, so neither the corner detection error for each image nor the overall bias could be calculated. The variances of the estimated locations for each of the 40 corners were calculated, which, assuming the corner detection error is independent and normally distributed and the corner location is constant, are the error variances.

Next, the camera was mounted on a Universal Robotics UR5 robot arm, which was used to position and orient the camera precisely. First, images were captured with the checkerboard parallel to the camera, initially positioned in the centre at the image and then positioned at each point in a grid with 50 mm spacing which extended 350 mm

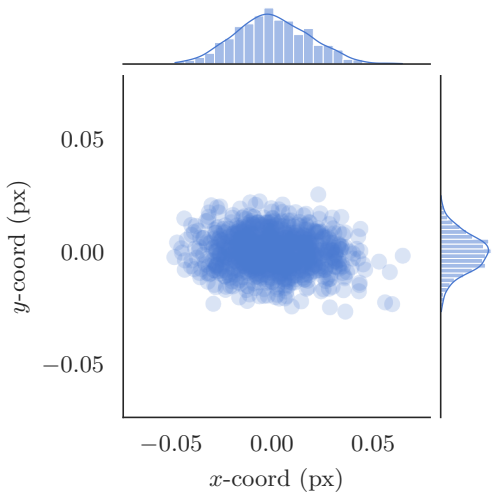
Figure 5.3 The experimental setup used for collecting the real data, showing the UR5 robot arm and teach pendant (centre), wall-mounted checkerboard (top), and LED panels (left, right). Another LED panel is out of frame, pointing up from below.



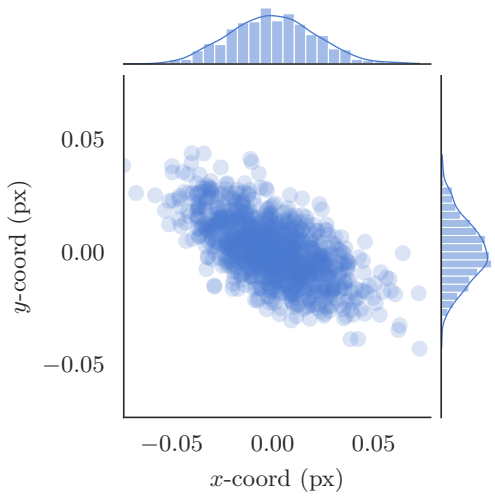
(a) An example of the common (isotropic) case, where the error distributions for the corner's x- and y-coord have similar variances.



(b) An example of a corner where the error distribution for the y-cooord has a higher variance than the x-coord.



(c) An example of a corner where the error distribution for the x-cooord has a higher variance than the y-coord.



(d) An example of a corner where the error for the x- and y-coords is correlated ($\mathrm{Corr}\,(X, Y) = -0.63$).

Figure 5.4 A selection of representative corner detection error distributions from the robot arm images, shown with the same axis scale.

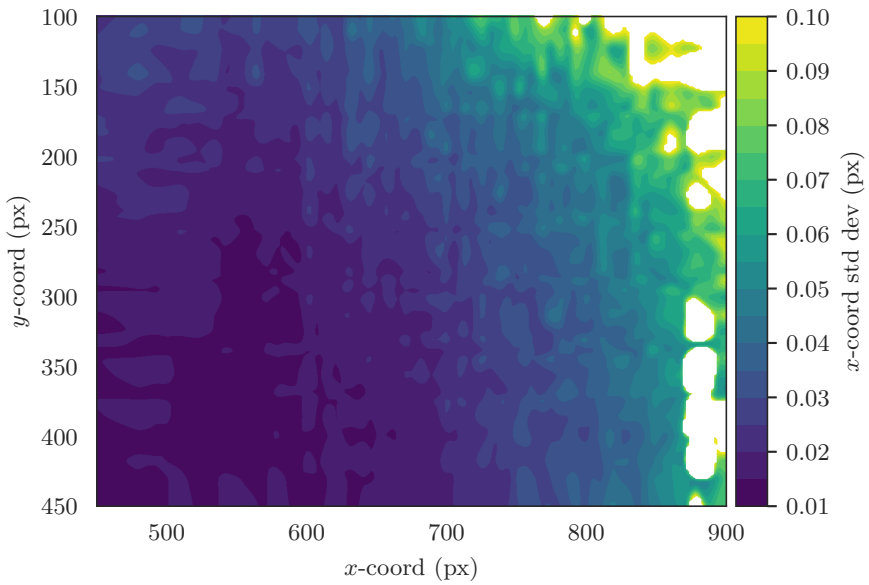


Figure 5.5 A heatmap of the standard deviations of the estimates of the x-coordinate of each corner in the upper right quadrant of the images from the first robot arm dataset, displaying a sharp increase towards the corner. The areas in white are where the image is sufficiently blurry that the initial corner detection process fails to consistently return the same (integer-valued) result, and so after sub-pixel refinement, the result is a group of clusters spaced 1 pixel apart.

along the x-axis and 300 mm along the y-axis, such that the resulting checkerboard corner locations spanned the upper right quadrant of the image. Then, another dataset was collected where the checkerboard started in the centre of the image and moved 50 mm along the x-axis in 0.5 mm steps.

The corner detection error is generally normally distributed, as shown in Figure 5.4. However, the variance is not uniform, as shown in Figures 5.5 and 5.6. The variance increases towards the edge of the image. This is likely a result of the warping operation used to correct for radial lens distortion during rectification, which resulted in images that are increasingly blurry toward the edges. When an image is too blurry, the corner detection process produces multiple clusters of results rather than a single result; in this case, a Gaussian distribution is inappropriate. The variance does not increase smoothly toward the edges, however; the inconsistency is likely due to the sub-pixel effects discussed in Section 5.3.2.

In the tripod images, the noise distributions for the x- and y-coord estimates for each corner are similar—although not identical—and approximately uncorrelated ($|\mathrm{Corr}\,(X, Y)| < 0.2$). Figure 5.7a shows the distribution of the ratio of standard deviations; for most corners, $0.8 < \frac{\sigma_Y}{\sigma_X} < 1.2$. Similarly, the estimates of each corner's location are independent from the other corners', as shown in Figure 5.8. The cross-correlation between any two corners, $|\mathrm{Corr}\,(X_i, Y_j)|$, is generally less than 0.2.

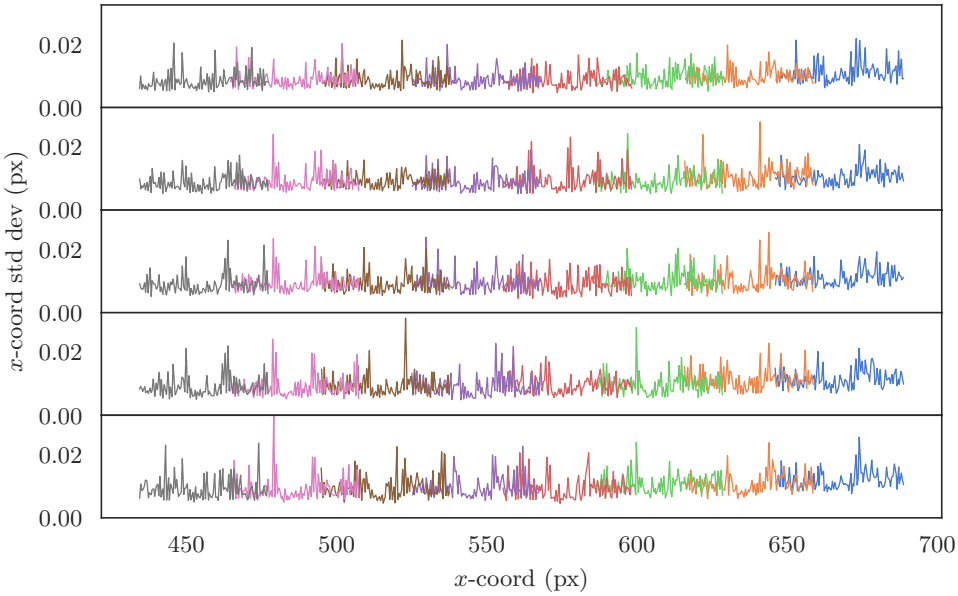
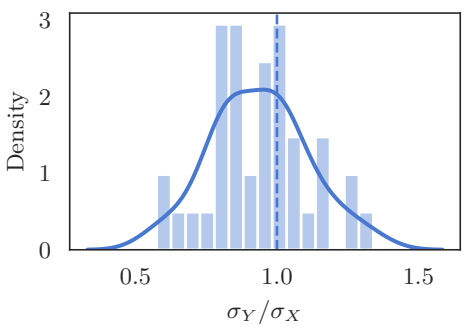
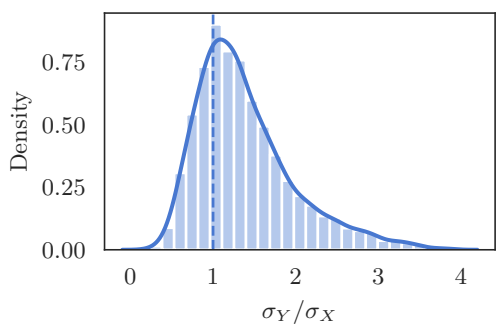


Figure 5.6 The standard deviation of the estimate of the x-coordinate of each corner in the second robot arm dataset (in which the camera moves 50 mm along the x-axis in 0.5 mm steps), plotted as a function of image location. The checkerboard has five rows of eight corners; each row is plotted separately, with the corners within each row are distinguished by colour. The spatial variability shown in this graph is the reason the contour plot in Figure 5.5 has such inconsistent contours.

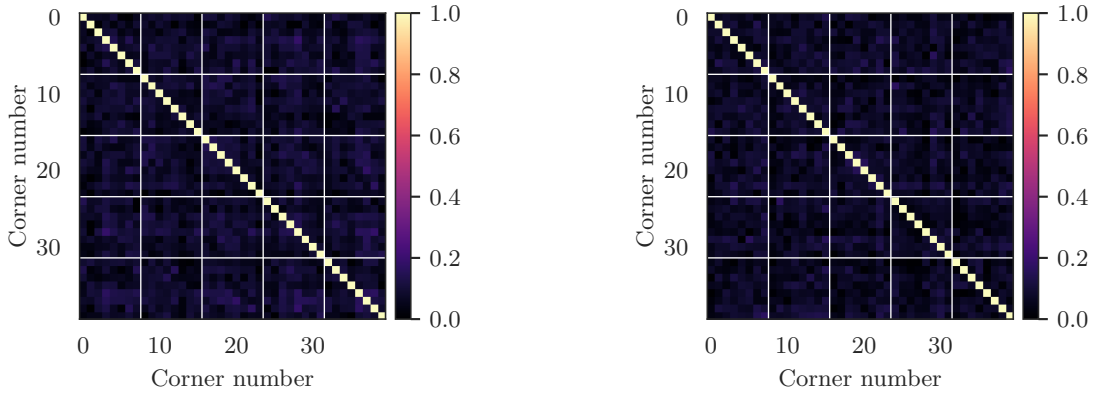


(a) The distribution of $\frac{\sigma_Y}{\sigma_X}$ for the tripod images (1,000 images). For most corners, $0.8 < \frac{\sigma_Y}{\sigma_X} < 1.2$, which indicates they have a similar noise distribution.



(b) The distribution of $\frac{\sigma_Y}{\sigma_X}$ for the robot arm dataset, a total of 4,000 corners (with 1,000 images of each). For most corners, $0.5 < \frac{\sigma_Y}{\sigma_X} < 2$.

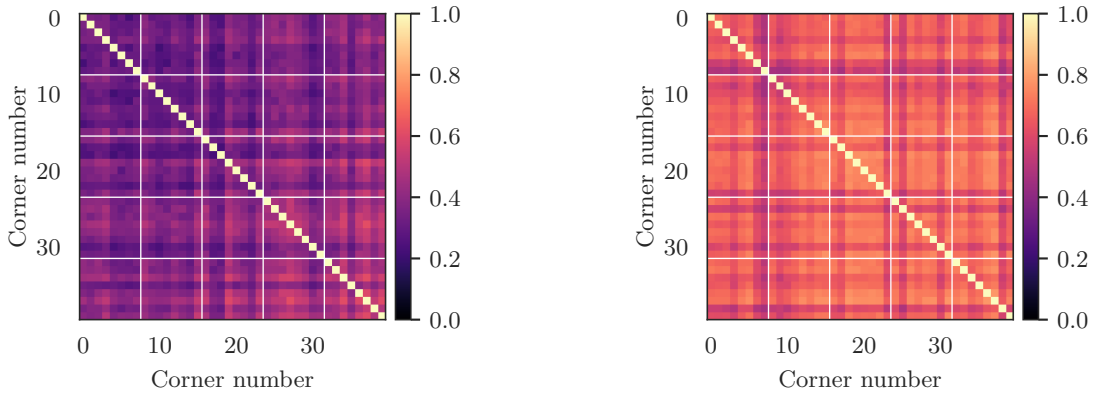
Figure 5.7 The estimate of the x- and y-coordinate of each corner have normally distributed noise with standard deviation σ_X and σ_Y. These graphs show histograms and kernel density estimates for the overall distribution of the ratio of the two standard deviations, $\frac{\sigma_Y}{\sigma_X}$, which indicates how close each corner is to having the same noise distribution. The vertical line is at $\sigma_X = \sigma_Y$; which would indicate identical noise distributions. The long tail in (b) is likely due to vibration in the robot arm, as discussed in Section 5.3.1.



(a) The correlation matrix for the estimates of the x-coordinates.

(b) The correlation matrix for the estimates of the y-coordinates.

Figure 5.8 Correlation matrices for the estimates of each corner's location in the tripod images, shown as heatmaps. The off-diagonal elements are small, indicating that estimates of the location of each corner are approximately uncorrelated with estimates of the locations of the other corners.



(a) The correlation matrix for the estimates of the x-coordinates.

(b) The correlation matrix for the estimates of the y-coordinates.

Figure 5.9 Correlation matrices for the estimates of each corner's location in an example from the robot arm images. These matrices show significant correlation between corners, particularly for the y-coordinates.

This is not true for the robot arm images, likely due to vibration.[5] The noise distributions for the x- and y-coord estimates for each corner are dissimilar, as illustrated by the $\frac{\sigma_Y}{\sigma_X}$ distribution shown in Figure 5.7b. Most of the corners satisfy $0.5 < \frac{\sigma_Y}{\sigma_X} < 2$; there is a bias towards larger σ_Y as the robot arm is oriented with the most-loaded joints moving vertically. Figures 5.4b and 5.4c show examples of corners where $\sigma_X \neq \sigma_Y$, and Figure 5.4d shows an example of a highly correlated corner. Figure 5.9 shows that the correlation between corners is high, particularly for the y-coord estimates.

[5] Although the images are not visibly blurry, the vibration in the arm while it is stationary is perceptible by touch. This vibration was not measured quantitatively due to external constraints, but doing so would be useful future work.

5.3.2 Simulated data

A simulation for corner detection of a single checkerboard corner was developed.[6] First, a 2000 px × 2000 px image of a checkerboard corner was generated, then a Gaussian blur was applied to simulate a lens [as in Lucchese and Mitra 2002], then the image was divided into a 20 × 20 grid of 100 px squares and each grid section was averaged to simulate a camera sensor array. Each step of this process is shown in Figure 5.10. This was repeated with the corner at each location in a grid spanning one pixel in the centre of the resulting image. For each of these images, a Monte Carlo experiment was performed in which i.i.d. Gaussian intensity noise was added to simulate image sensor noise[7] and then the OpenCV `cornerSubPix` function was used to produce a corner estimate in the same manner as for the real images [as in Chen and Zhang 2005].

The resulting corner detection error distribution is not uniform across each pixel. Figure 5.11 shows the result of a simulation with 0.2 px of lens blur (i.e., the Gaussian blur kernel used on the high-resolution image had $\sigma = 20$ px). There is significant bias in the estimates which depends on their location within the pixel. The variance of the estimates is also dependent on location. In a simulation with 1 px of lens blur,[8] the bias is negligible and the variance is higher (as shown in Figure 5.12).

To visualise the dependence of variance on location, the same simulation was performed with a much denser grid of corner locations, and the resulting x- and y-coordinate estimate standard deviations were plotted as heatmaps (shown in Figures 5.13 and 5.14). These show that the standard deviation of the error in each coordinate is lowest when the corresponding edge is at the edge of a sensor element (i.e., the vertical edge for the x-coordinate and the horizontal edge for the y-coordinate) and highest when the edge is at the centre of a sensor element. The standard deviation varies by a little over 30 % across the sensor element. Note that the overall distribution of distributions is not Gaussian, as shown in Figure 5.15. Horizontal cross-sections of Figure 5.13 with an offset of 0, 0.1, 0.2, and 0.5 pixels from the centre are shown in Figure 5.16. This shows that the standard deviation for the x-coordinate has a weak dependence on the value of the y-coordinate (the converse is also true).

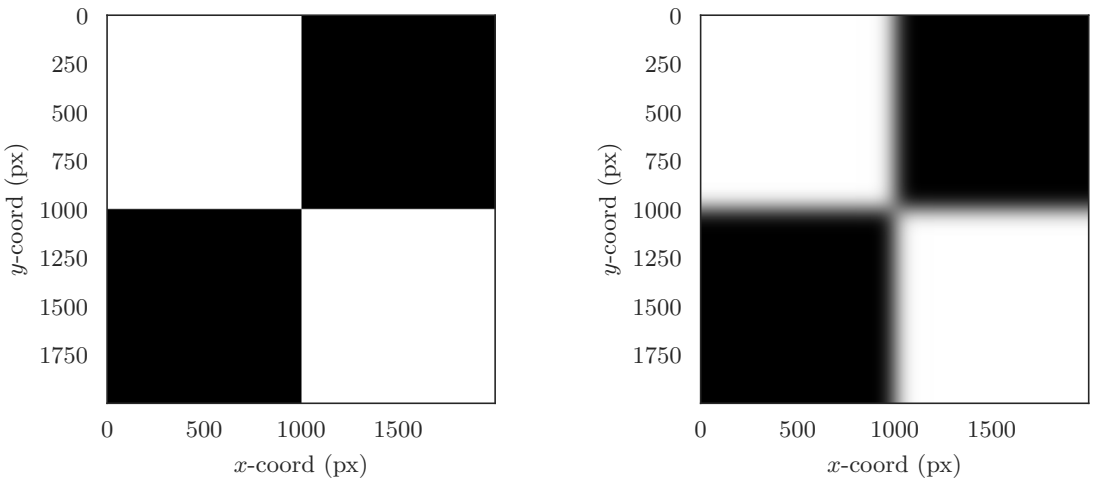
5.3.3 Mathematical modelling

Recall that (5.13) gives an expression for the estimated x-coordinate of a corner given an image window. Given a corner location (x, y) within the central pixel of the window

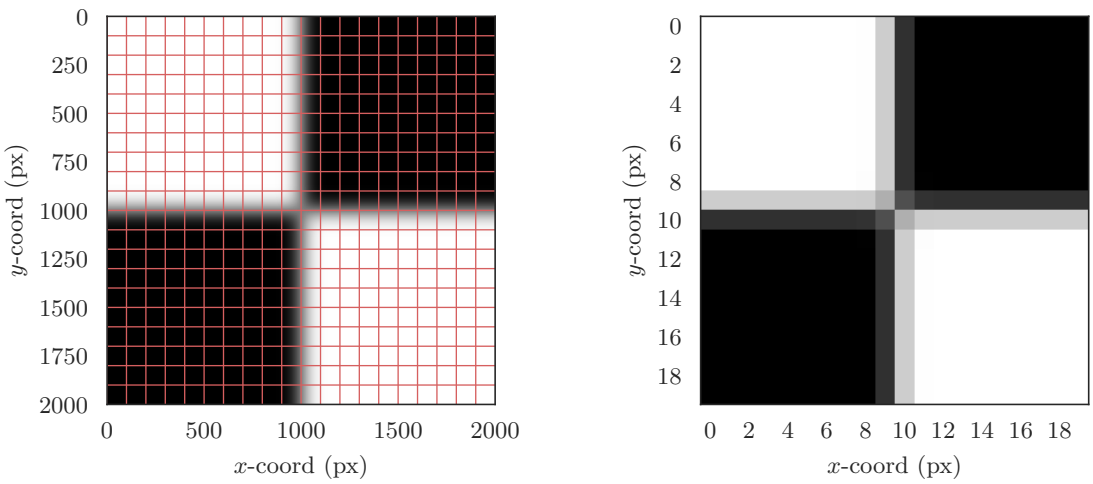
[6] Since no checkerboard corner detector is used, this is really a simulation of sub-pixel refinement, but since the last step in both cases is sub-pixel refinement, the results should have the same statistical properties.

[7] Note that real image sensor noise has intensity-dependent variance, as discussed in Section 6.3.1.

[8] For comparison, the camera used in Chapter 6 was found to have around 0.6 px of lens blur.



(a) A 2000×2000 px image of a checkerboard corner which was generated with the corner precisely in the centre.



(b) The image with Gaussian blur applied to simulate lens blur.

(c) The blurred image divided into a 20×20 grid of 100 px squares, the pixels in which will be averaged to simulate a camera sensor array.

(d) The resulting 20×20 px image, with the corner at $(9.5, 9.5)$, i.e., exactly between the pixels at $(9, 9)$ and $(10, 10)$.

Figure 5.10 Images of each step in the checkerboard corner image generation process from the simulation in Section 5.3.2.

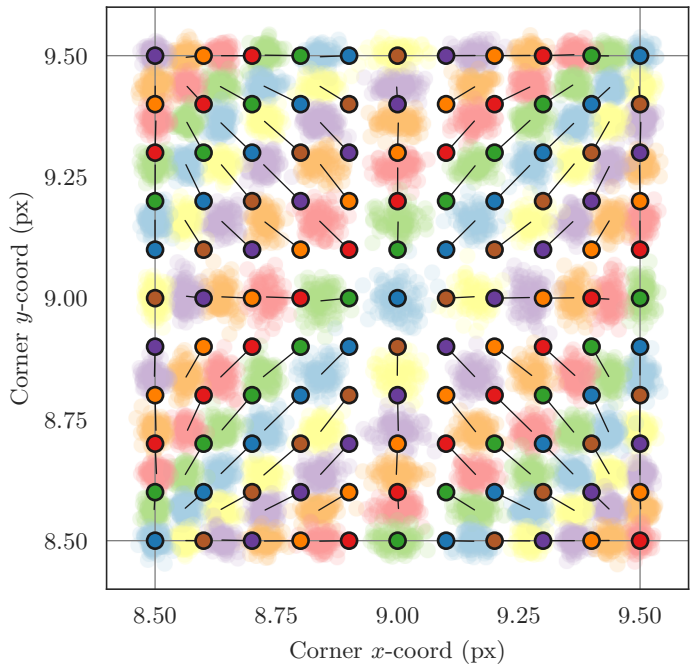


Figure 5.11 The results of a checkerboard corner detection simulation with 0.2 px lens blur, plotted over a single pixel. Simulated images of a checkerboard corner with the corner located at each of the outlined points were generated, such that the true corner locations fall on an 11×11 grid spanning the pixel at $(9, 9)$. For each location, the corresponding cloud shows the results of adding Gaussian noise to the image 1000 times and estimating the corner location from each noisy image. The lines point from the true location to the mean of the estimates; that is, they are the bias vectors. The magnitude of the bias is up to 0.08 px. Note how the bias and variance of the estimates depend on the location within the pixel.

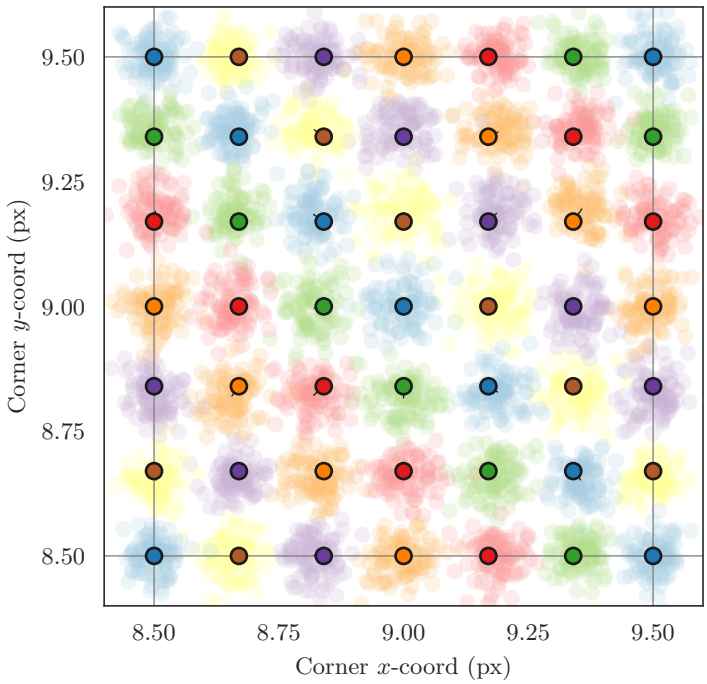


Figure 5.12 The results of a checkerboard corner detection simulation similar to the one in Figure 5.11 but with 1 px lens blur and a 7×7 grid of corner locations. Note how the bias is significantly smaller and the variance is significantly larger.

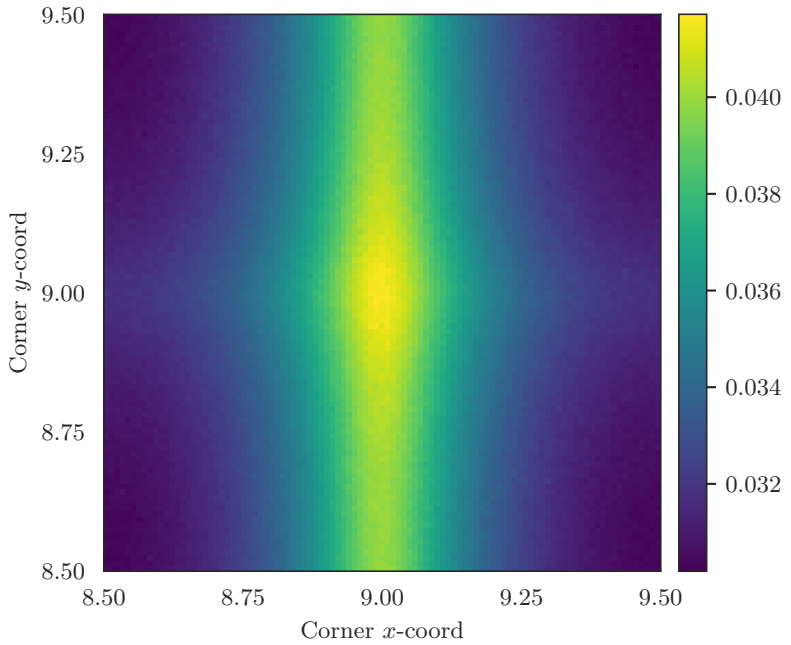


Figure 5.13 The results of a checkerboard corner detection simulation similar to the one in Figure 5.12, but with a 100×100 grid of corner locations. In this graph, the standard deviations of the estimates of the x-coordinate of each corner are plotted as a heatmap, showing how the estimate variance depends on the location within the pixel.

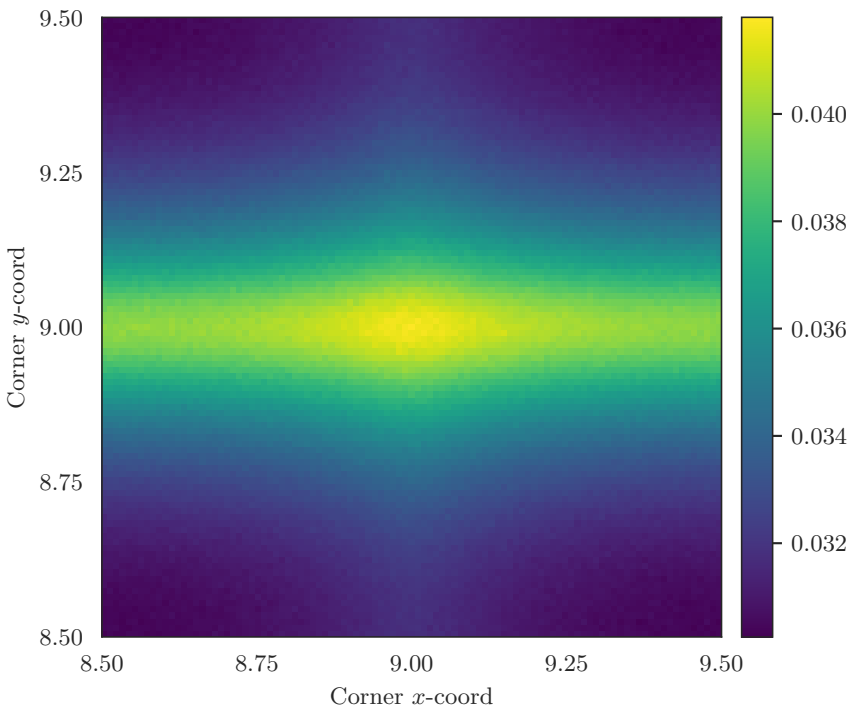


Figure 5.14 A heatmap of the standard deviations of the estimates of the y-coordinate of each corner from the simulation shown in Figure 5.13.

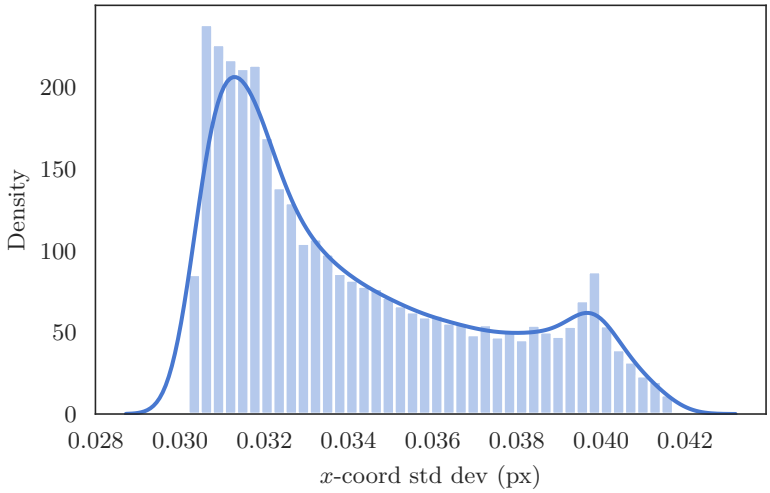


Figure 5.15 A histogram and kernel density estimate showing the overall distribution of x-coordinate standard deviations in Figure 5.13. In Chapter 4, it was assumed that the corner detection error had constant variance; this simulation shows that that assumption does not hold.

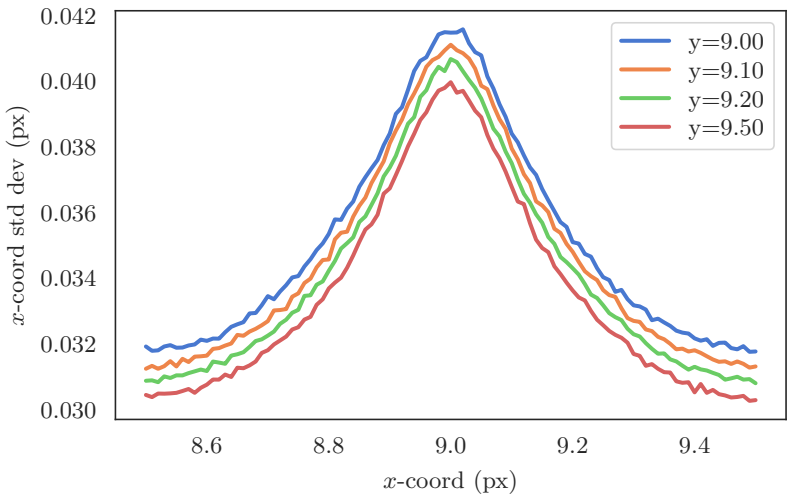


Figure 5.16 A graph of horizontal cross-sections through Figure 5.13 with offsets of 0, 0.1, 0.2, and 0.5 px from the centre of the pixel, showing how the x-coordinate standard deviation depends on the x-coordinate, and to a lesser extent, the y-coordinate. A similar relationship holds for the y-coordinate standard deviation. Note that the curve is not quite Gaussian-shaped; a Pearson VII model fits it well.

where $x, y \in [-0.5, 0.5]$, an image window can be constructed as follows:

$$\mathbf{I} = \begin{bmatrix} 1 & 1 & x+\frac{1}{2} & 0 & 0 \\ 1 & 1 & x+\frac{1}{2} & 0 & 0 \\ -y+\frac{1}{2} & -y+\frac{1}{2} & -2xy+\frac{1}{2} & y+\frac{1}{2} & y+\frac{1}{2} \\ 0 & 0 & -x+\frac{1}{2} & 1 & 1 \\ 0 & 0 & -x+\frac{1}{2} & 1 & 1 \end{bmatrix}. \tag{5.14}$$

This assumes a perfectly sharp lens. Substituting (5.14) into (5.13) gives

$$\hat{x} = \frac{4x}{4x^2+3}. \tag{5.15}$$

That is, the model predicts that sub-pixel corner refinement is biased depending on where the corner falls within the pixel. This appears in simulation, as shown in Figures 5.11 and 5.17.

Applying a 3×3 averaging filter to (5.14) to simulate lens blur and solving again for $\hat{x}$ gives

$$\hat{x} = \frac{10x}{4x^2+9}. \tag{5.16}$$

The general solution for an $L \times L$ averaging filter is

$$\hat{x} = \frac{Nx}{4x^2+N-1}, \tag{5.17}$$

where

$$N = \begin{cases} 4L-2, & \text{for } L > 1, \\ 4, & \text{for } L = 1. \end{cases} \tag{5.18}$$

That is, the model predicts that the bias decreases with increasing lens blur, as shown in Figure 5.18. This also appears in simulation, as shown in Figure 5.12.

5.4 DISCUSSION

The main result from these experiments is that i.i.d. Gaussian noise is not an accurate model for checkerboard corner detection error; in practice, the variance of the error is not identical for every corner or image location, and the corner error distributions are not always identical. This is unfortunate for work like that in Chapter 4 which needs a simple error model.

The robot arm experiment was problematic. Universal Robotics specifies the repeatability of the UR5 robot arm as ± 0.1 mm, so the idea was to use it to capture images from a wide variety of camera poses as recommended in Chapter 4. Unfortunately, the ROS driver did not come close to achieving this (at the time). In the experiment,

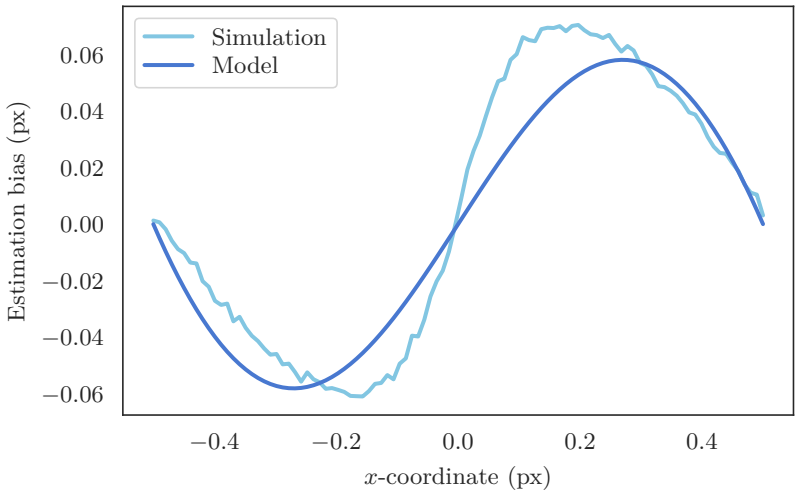


Figure 5.17 Sub-pixel refinement introduces a corner location-dependent bias. This graph of x-coordinate bias as a function of the corner location's x-offset from the centre of a pixel compares the bias observed in the simulation results from Section 5.3.2 with the bias predicted by the model from Section 5.3.3. For this graph, both the simulation and model used a 1×1 averaging filter. The increased bias in the simulation compared to the model is likely because the sub-pixel refinement process in the simulation (that is, the OpenCV `cornerSubPix` function) applies the Förstner operator and linear interpolation repeatedly until the estimate converges, while the model predicts the bias from a single application of the Förstner operator.

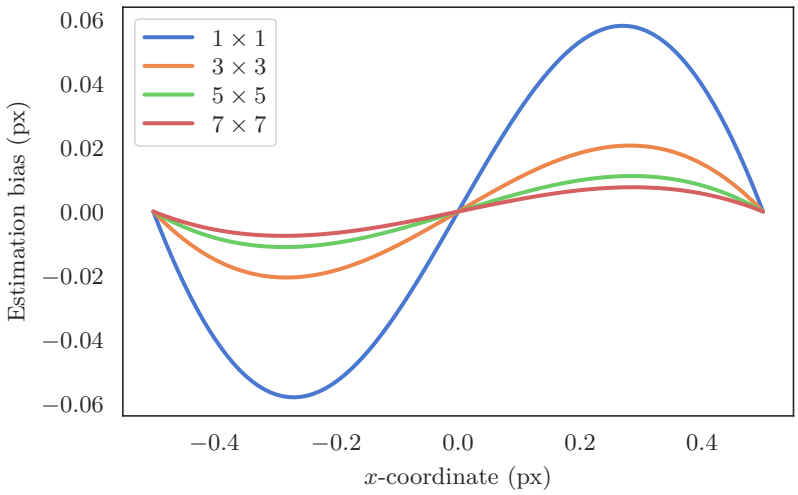


Figure 5.18 The sub-pixel refinement bias predicted by the model from Section 5.3.3 with 1×1, 3×3, 5×5, and 7×7 averaging filters used to simulate different levels of lens blur.

move commands were issued repeatedly until the desired pose was reached in order to work around a collection of driver bugs, but even after extensive troubleshooting, the repeatability was never consistently below a few millimetres. Another problem was that vibration in the robot arm produced motion-blurred images, resulting in correlated estimates. This was discovered late in the process, and due to time constraints the only mitigation used was to pause after movement while the transient response died down; a better workaround may be possible. The Universal Robotics ROS driver has since matured significantly; re-evaluating the suitability of the Universal Robotics robot arms for imaging research would be useful future work. A similar phenomenon likely occurs with images captured from a mobile platform, which could be an issue in, for example, ground truth for VO algorithms; this would also be useful future work. Another limitation of the robot arm experiment is that rather than performing hand-eye calibration to determine the pose of the checkerboard with respect to the robot arm base, the robot arm was simply moved until the checkerboard detector returned the desired pose each time. In this case, the position control was so imprecise that it would not have made a difference, but a future experiment with a better robot arm should use a proper calibration procedure.

The simulation results show that corner detection error has variance and bias which substantially depend on the location of the corner within a pixel and that this dependence decreases as lens blur increases. This can result in different variance and bias in different image regions in, for example, scenes with checkerboards at different distances so the lens blur varies and rectified images where some areas are warped significantly to correct radial distortion. The main limitation of the simulation experiment is that rotation, perspective distortion, and lens distortion were not considered. There is likely a similar effect in more general scenarios; investigating this would be useful future work. The mathematical model allows the bias to be predicted symbolically, with a further limitation: it predicts the bias from a single application of the Förstner operator, while the full sub-pixel refinement process repeatedly alternates the Förstner operator with linear interpolation (which is also known to introduce bias, as discussed in Chapter 6).

5.5 CONCLUSION

Modelling checkerboard corner detection error with i.i.d. Gaussian random variables is a good first approximation, with the following caveats:

- Corner detection error variance depends on the amount of lens blur in the image, so images that are not equally sharp throughout (for example, images with checkerboards at different depths or images which have been rectified) can have different error variance in different image regions.

- Corner detection in blurry images can produce results with outliers or multiple clusters, in which case a Gaussian distribution is not appropriate.
- Corner detection in images with any amount of motion blur can produce correlated results.
- In images with little blur (less than approximately one pixel), corner detection error has variance and bias which depends on the location of the corner within a pixel.

Chapter 6

SUB-PIXEL-VARYING BIAS AND ERROR IN CORNER REFINEMENT

In Chapter 5, simulations were used to show that the estimation noise and bias of OpenCV's sub-pixel corner refinement algorithm measurably depend on the sub-pixel location of the corner. In this chapter, which is based on published work [Edwards et al. 2020], that effect is demonstrated in realistic conditions using a real camera and examined in the context of research from other areas.

The rest of this chapter is structured as follows: Section 6.1 presents background theory, Section 6.2 outlines the methods used for data collection and simulation, Section 6.3 examines each aspect of the distribution of corner detection error, and Section 6.4 summarises the results.

6.1 BACKGROUND

The OpenCV corner refinement implementation relies on bilinear interpolation for calculating sub-pixel aligned windows (as explained in Section 5.2). In other areas of image processing, bilinear interpolation (and interpolation in general) is known to introduce amplitude attenuation and phase noise that depend on sub-pixel position [Bailey et al. 2005, Jähne 2004, Schöps et al. 2020, Wang et al. 2016]. In particle image velocimetry, the tendency for estimated particle positions to be biased toward certain locations relative to pixel edges is known as "peak-locking" or "pixel-locking" [Michaelis et al. 2016, Raffel et al. 2018].

6.2 METHOD

A large dataset of real images was collected using a camera on a slider and then reproduced in simulation for further analysis. These datasets were used to examine the error distribution of checkerboard corner estimation.

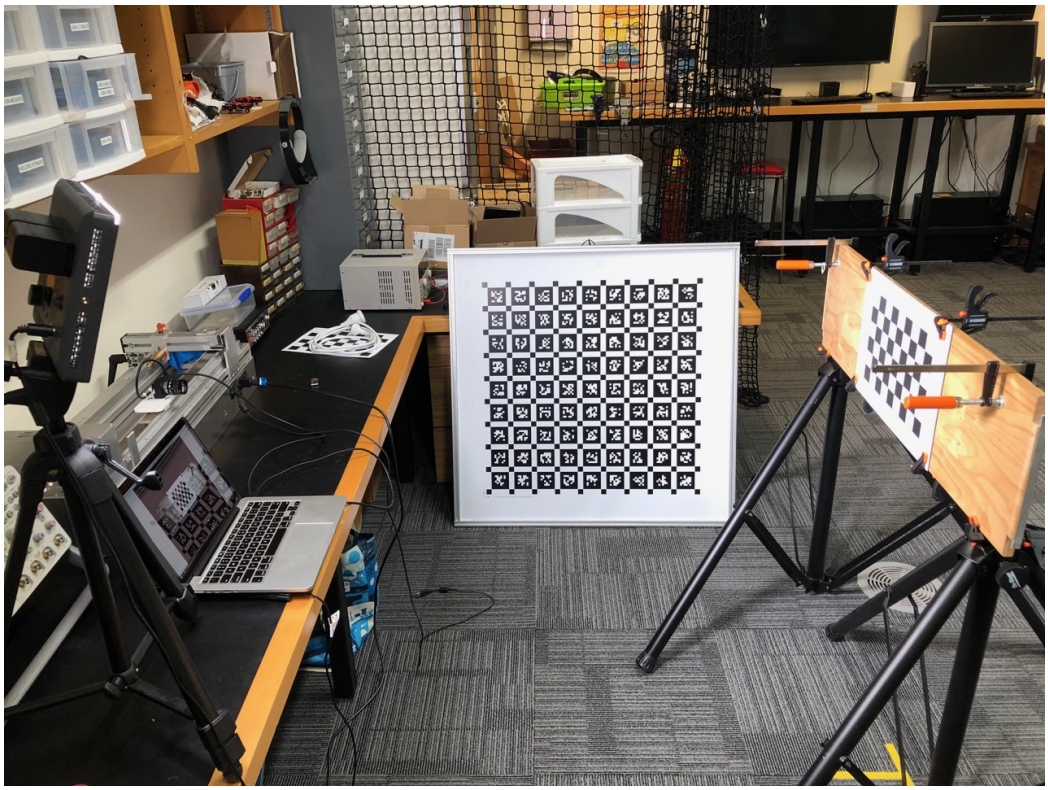


Figure 6.1 Equipment used in data collection, from left to right: LED panel, laptop, camera on slider, Aprilgrid used for camera calibration, and checkerboard on stand.

Figure 6.2 Data collection setup, showing positions and lighting used. From left to right: laptops, camera on slider, LED panels, and checkerboard on stand.

6.2.1 Real data

A Flir Chameleon3 CM3-U3-13Y3M-CS camera with a Computar M0814-MP2 lens was mounted on an SMC LEFB25T-500 slider-type electric actuator controlled by an Arcus Performax PMX-2ED-SA USB motion controller, allowing it to be moved in 0.06 mm steps along a fixed axis perpendicular to its optical axis (see Figure 6.1). A 720 mm 9×9 Aprilgrid calibration target offset-printed on aluminium composite board mounted in an aluminium extrusion frame was used for camera intrinsic calibration, with the Kalibr calibration software [Furgale et al. 2013]. An 8×5 checkerboard pattern with 35 mm squares, offset-printed on aluminium composite board (as shown in Figure 5.2), was mounted on a stand and aligned manually with the camera such that it was initially positioned in the centre of the image (within 0.5 mm) and aligned with the camera's horizontal and vertical axes (within 4, 0.4, and 0.1 degrees about the x, y, and z axes). Two LED panels were used to light the checkerboard approximately evenly from both sides (see Figure 6.2). The camera exposure time was adjusted such that the brightest image regions did not quite reach the maximum intensity value. The camera was moved 4 mm in total along the slider in 0.06 mm steps, taking 1000 12-bit monochrome images with 1280 px $\times$ 1024 px resolution of the checkerboard at each of the 70 steps

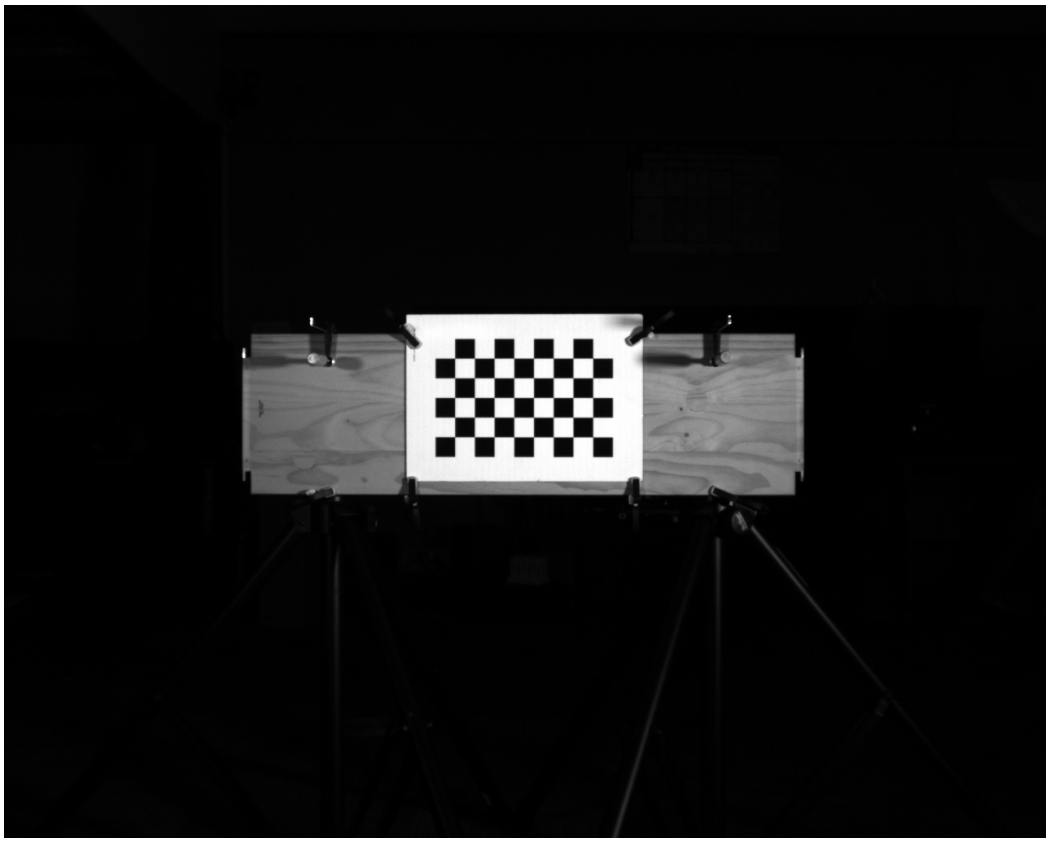


Figure 6.3 An example image from the first slider dataset showing the checkerboard on the stand under controlled lighting.

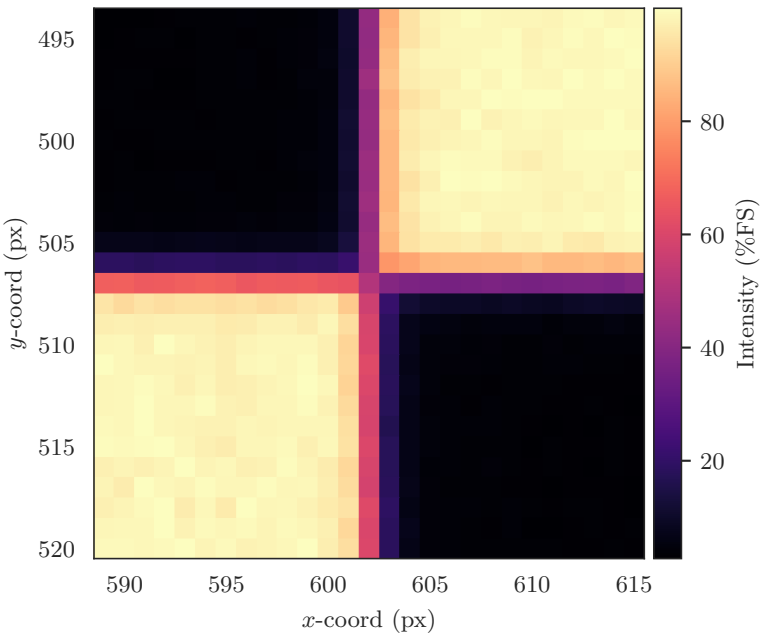


Figure 6.4 A zoomed-in section of Figure 6.3 showing the intensity values (as a percentage of the full-scale intensity) around a single corner. False colour is used to to distinguish mid-range values.

(see Figures 6.3 and 6.4). The slider is belt-driven, so in order to avoid backlash, the controller was set to never perform corrections after moves (i.e., to only ever move in one direction). This occasionally resulted in moves where the reported slider position did not change;[1] in these cases, the images from that camera position were not used for analyses involving camera movement.

The images were not rectified, in order to preserve their original sharpness characteristics. After processing, the images were found to be overexposed (see Figures 6.6 and 6.7), and the precise horizontal alignment resulted in gaps in the corner distribution (see Figure 6.11). To remedy this, the procedure was repeated with the camera exposure time reduced and the checkerboard rotated by two degrees about the z-axis (so that the corners were distributed more evenly across the pixel space as shown in Figure 6.12a). These two sets of images are referred to as the first and second slider datasets.

6.2.2 Simulation

Precise alignment is difficult with measured data and so a supplementary synthetic dataset was generated. This dataset was parametrised by the measured data from the second slider dataset, using the following method:

- For each camera position, the mean of all the images was taken.

[1] The reported position is rounded down to the nearest step, so for a move from, for example, 0.6 mm to 0.659 mm, the start and end points are both reported as 10 steps.

- The checkerboard was detected in the mean image using the `findChessboardCorners` function from OpenCV 4.4 [Bradski 2000].
- Sub-pixel refinement was performed on each detected corner using the `cornerSubPix` function.
- The refined corner locations were used to estimate the checkerboard position relative to the camera using the `solvePnP` function.
- A ray-casting algorithm was used to render an image of a checkerboard the same size and shape in the same position with the same camera intrinsics as the real camera (including lens distortion).
- Gaussian blur was added to approximate the real camera's lens blur, and the image intensities were transformed to match the intensity distribution in the real images (see Section 6.2.3).
- 1000 simulated images were produced by adding intensity-dependent Gaussian intensity noise with the same distribution as the intensity noise distribution in the real images (see Section 6.3.1) to the render.

Adaptive supersampling was used to accurately render edge pixels (see Section 6.3.2). Supersampling levels over 1000× (i.e., sampling a 1000×1000 grid of points for each edge pixel in the final image) were found to be computationally infeasible. For the final simulation 100× supersampling was chosen, producing renderings that matched the 1000×-supersampled renderings to within 0.1% of the intensity range.

6.2.3 Estimation of lens blur and affine intensity transformation

The mean of the 1000 images from the first camera position in the second slider dataset was taken as a template image. Simulated images using Gaussian blurs with a range of standard deviations were rendered. For each simulated image, numerical optimisation was used to find the affine intensity transformation $aI+b$ that minimised the root-mean-square (RMS) error between the checkerboards in the template image and transformed render. The results are shown in Figure 6.5.

6.3 RESULTS AND DISCUSSION

The real image datasets were used to analyse the intensity noise distribution. An experiment was performed to determine how much supersampling is required to render the checkerboard edges accurately. The results of these were used to generate the simulated dataset. Finally, the real image datasets and the simulated dataset were used to analyse the corner detection error distribution.

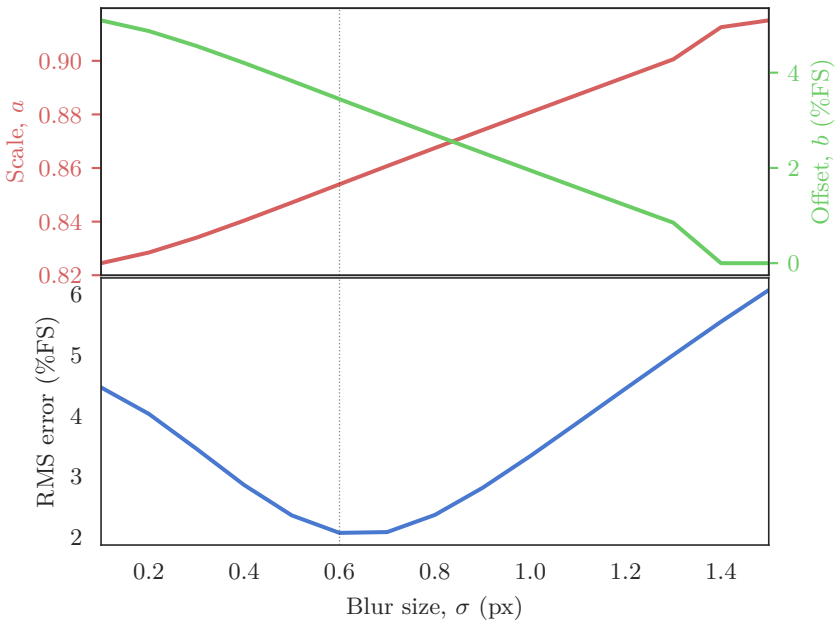


Figure 6.5 An illustration of the blur/intensity error surface for the second slider dataset, showing the minimum at 0.6 px blur, 3.44% offset and 0.85× scale. The upper plot shows the optimal intensity transformation for each blur value, and the lower plot shows the corresponding RMS error.

6.3.1 Intensity noise distribution

The real images are contaminated by noise, the statistical distribution of which is needed for the simulation. Noise from a digital image sensor comes from a variety of sources (shot noise, thermal noise, flicker noise, etc.), and can be generally divided into a Poissonian component (which depends on the intensity) and a Gaussian component (which does not) [Foi et al. 2008]. In practice, this noise can be modelled as a zero-mean Gaussian random variable with standard deviation $\sigma = \sqrt{aI + b}$, where I is the intensity and a and b are parameters. The image noise was calculated as the difference between each image and the mean for the 1000 images from the first camera position in the first slider dataset.[2] To confirm that a Gaussian distribution is appropriate, the pixels were grouped by intensity into 256 bins, a Gaussian distribution was fitted to the intensity noise of the pixels in each bin, and percentile-percentile plots [Thode 2002] for the resulting 256 distributions were produced (see Figure 6.6). Aside from a few outliers due to over-exposure, the distributions fit the data.

To investigate the relationship between intensity and the standard deviation of the noise, the sample standard deviation of the noise for each pixel was plotted against the mean intensity for that pixel (see Figure 6.7). The resulting curve is not a good fit for a simple Poissonian-Gaussian model (it resembles the Poissonian-Gaussian model with

[2] Only the first camera position was used, due to memory constraints: the full dataset is $70 \times 1000 \times 1280 \times 1024 \times 2$ bytes (about 171 GB). Repeating the procedure for other single camera positions gave similar results and the calculations for the percentile-percentile plot are inconvenient to perform incrementally, so it was concluded that using the whole dataset was unnecessary.

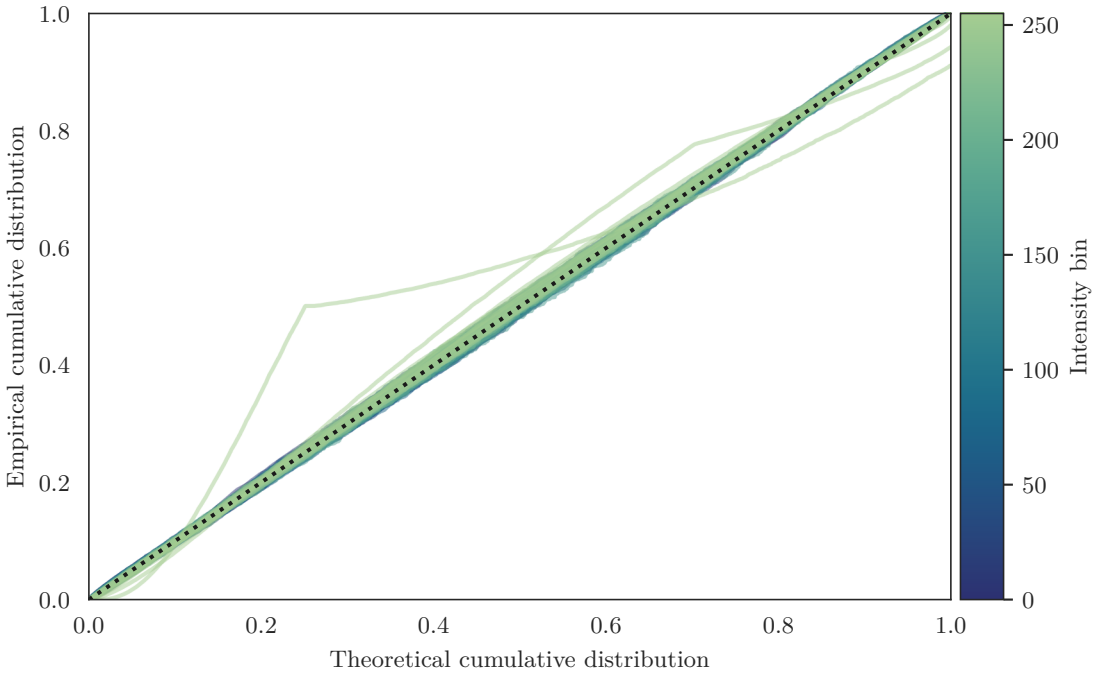


Figure 6.6 Percentile-percentile plots (one for each intensity bin) of the distribution of intensity noise in the 1000 images from the first camera position in the first slider dataset. These show that the noise distribution for nearly all intensity bins is normally distributed. The outliers are the top few bins, which are affected by overexposure. Other subsets of the dataset give similar results; the whole dataset was not used due to memory constraints.

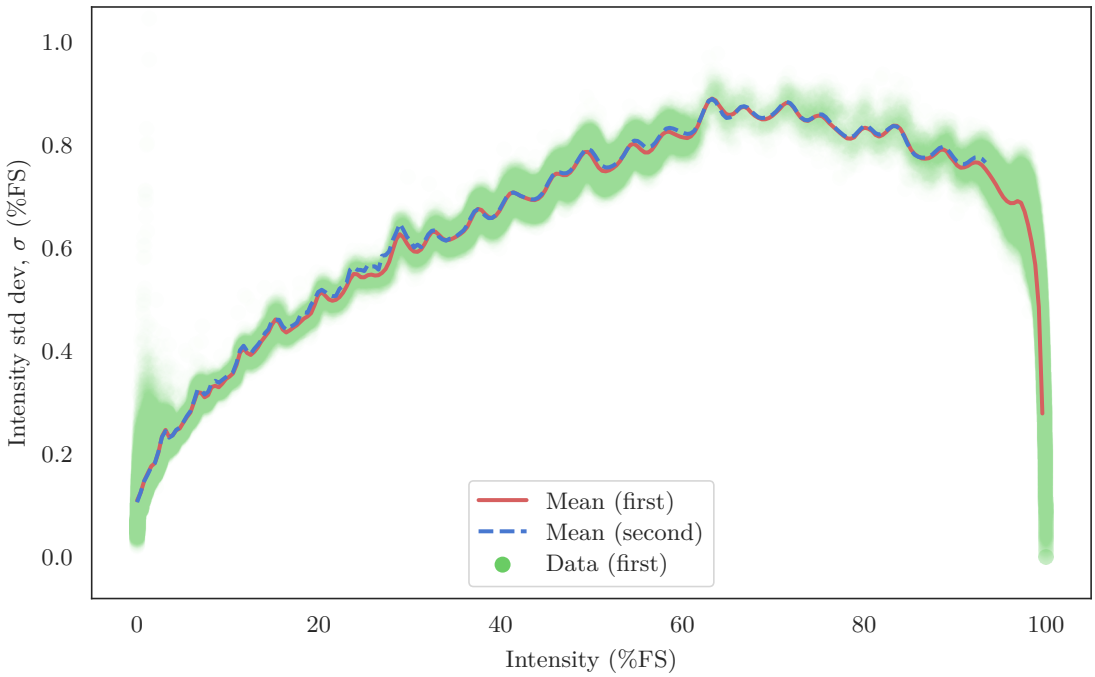


Figure 6.7 The sample standard deviation of the intensity noise for each pixel plotted against the mean intensity of that pixel for the first slider dataset, along with the mean of the standard deviations for the pixels in each intensity bin for both slider datasets. Note that the images are affected by overexposure, as indicated by the steep drop in intensity noise near 100% intensity.

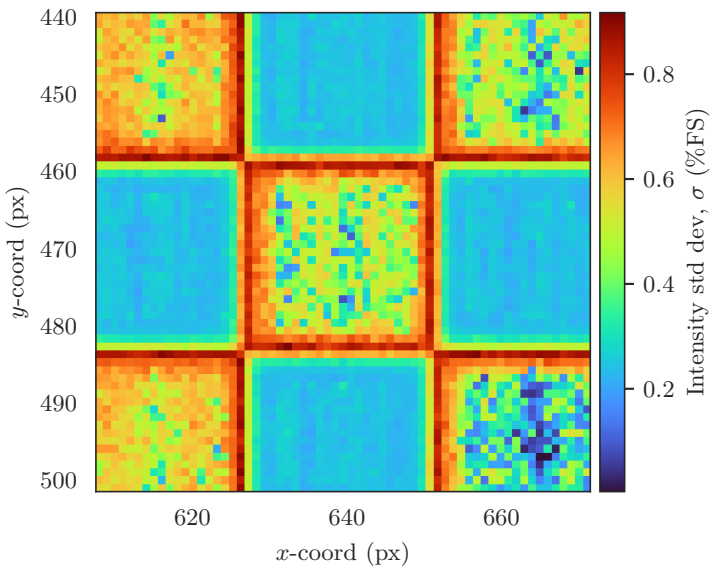


Figure 6.8 The standard deviation of the intensity noise in the region around one checkerboard square in the first slider dataset. At the edges, it is approximately 0.9% of the intensity range, and in the black squares, it is approximately 0.2%.

clipped observations presented by Foi et al. [2008] but with additional variation),[3] so a lookup table of the mean standard deviation of the noise for the pixels in each intensity bin was created. Images from every camera position were used to create the lookup table (not just the first), ensuring that even the least frequent intensities (the mid-greys that only show up in the edges of checkerboard squares) had at least 10,000 samples. This lookup table was used to generate noise in the simulated dataset.

Note that the standard deviation of the intensity noise varies by a factor of approximately four in the region around a corner (as shown in Figure 6.8); modelling image noise as an identically distributed Gaussian random variable may give poor results!

6.3.2 Supersampling

An important consideration for computer graphics rendering is anti-aliasing. For this simulation, adaptive supersampling was used; pixels containing edges were split into a grid of subpixels and a sample was taken from the centre of each. To determine an appropriate supersampling level, a simple image was rendered with edge pixels supersampled at 20 levels between 1× (i.e., no supersampling) and 1000× inclusive, and the difference between each rendering and the 1000×-supersampled rendering was calculated (see Figures 6.9 and 6.10). The results showed that relatively high levels

[3] The source of the high-frequency component of the curve in Figure 6.7 is not clear. It is consistent between camera positions and lighting conditions. Ideally, the intensity noise distribution would be characterised using images of a flat matte-white wall at a range of light levels. Although this particular camera's datasheet does not mention it, other Flir cameras use proprietary calibration procedures in order to achieve a desired intensity response, which could produce peaks and troughs in the intensity noise distribution.

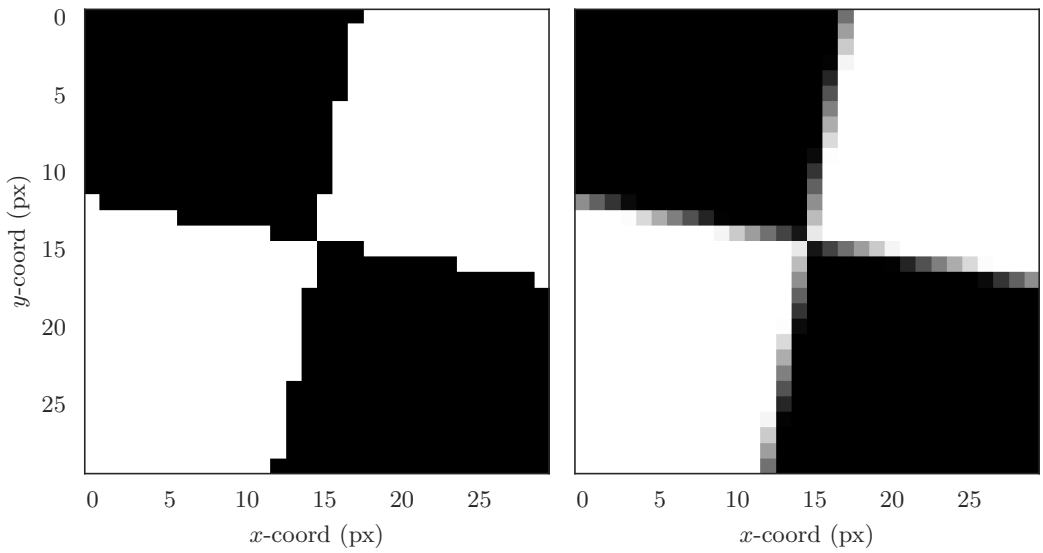


Figure 6.9 Two close-up renderings used to evaluate the effect of supersampling, one with no supersampling (left) and one with 1000× supersampling (right), showing the difference in intensity for edge pixels.

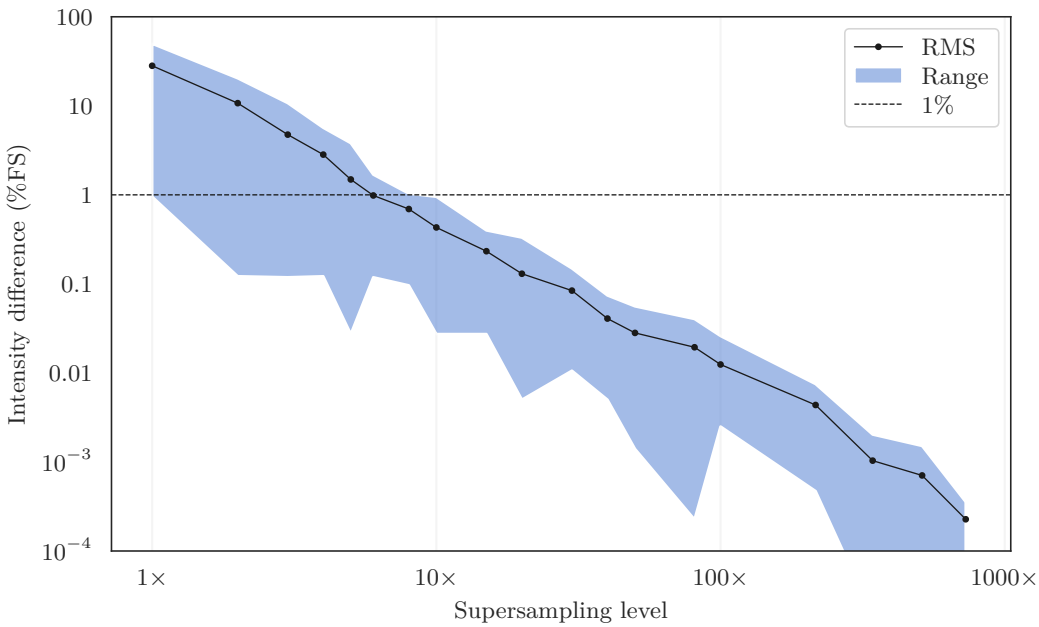


Figure 6.10 The difference (RMS and range) in edge pixel intensities between the 1000×-supersampled rendering and renderings with a range of supersampling levels between 1× (no supersampling) and 1000×.

of supersampling are required to accurately render simulated images in this context; the maximum error reaches 1% of the intensity range at around 8× supersampling (as shown in Figure 6.10).

6.3.3 Corner detection error distribution

In Chapter 5 it was demonstrated that checkerboard corner detection error is normally distributed and uncorrelated. Those simulations showed that the mean and standard deviation of corner detection error is dependent on the sub-pixel location of the corner,

$$\mathbf{x}^* = (x - \lfloor x \rceil, y - \lfloor y \rceil), \tag{6.1}$$

where $\lfloor x \rceil$ is x rounded to the nearest integer. Physically, this is the location within an individual sensor element in the camera to which the corner's position projects.

The datasets collected in Section 6.2.1 have camera motion with sufficient precision to demonstrate this phenomenon. Figures 6.11 to 6.13 show the standard deviations of the x- and y-coordinate estimates (i.e., the estimation noise power) as a function of sub-pixel corner location. These figures clearly show the same shape as in Figures 5.13 and 5.14: the standard deviation of the error in each coordinate is lowest at the corresponding edge of a pixel (i.e., the vertical edge for the x-coordinate and the horizontal edge for the y-coordinate) and highest when the edge is at the centre of a pixel. The standard deviation varies by a factor of approximately 1.8× depending on the sub-pixel position. The influence of the y-coordinate on the x-coordinate standard deviation is visible in the simulated data, but in the real data, it is hidden by noise.

The mean of the corner detection error (i.e., the estimation bias) is also dependent on the sub-pixel location of the corner. This can be shown in simulation by using the camera calibration and the checkerboard pose estimate to reproject the corners and then plotting the reprojection error in terms of sub-pixel location (6.1), as shown in Figure 6.14. The figure shows a clear (although small) bias, which varies in magnitude and direction depending on the sub-pixel corner location. Producing the equivalent plot using the real images (see Figure 6.15) shows a combination of the estimation bias and the effect of any lens distortion un-modelled by the distortion model used in camera calibration. To show a result as clear as that of the simulation, the camera calibration process would need to fit a lens distortion model with reprojection error on the order of 0.01 px; otherwise, the bias would be hidden in the rectification noise. The RMS bias magnitude in Figure 6.14 is 0.026 px and the RMS reprojection error from the camera calibration (across a much broader set of corners) is 0.45 px, so it is not surprising that Figure 6.14 and Figure 6.15 do not match. Using a generic (rather than parametric) camera calibration method such as the one presented by Schöps et al. [2020] could reduce the residual lens distortion enough to show this bias using real images. It is also

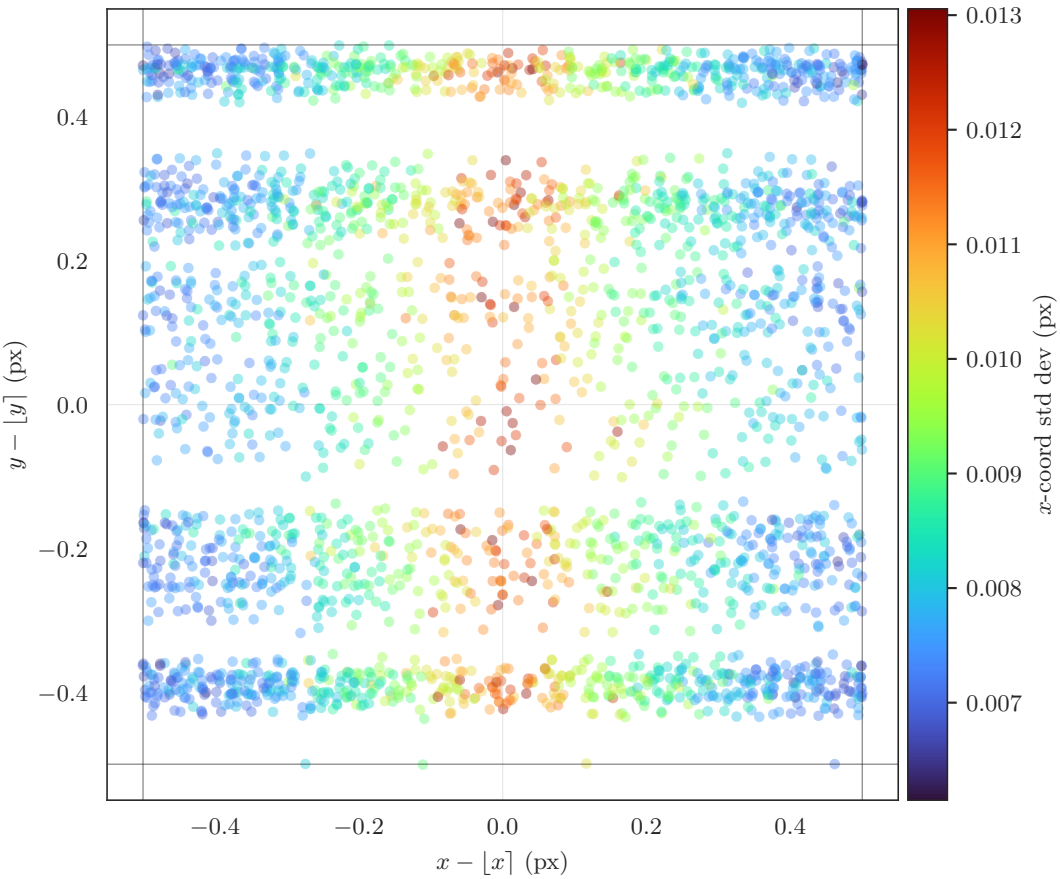
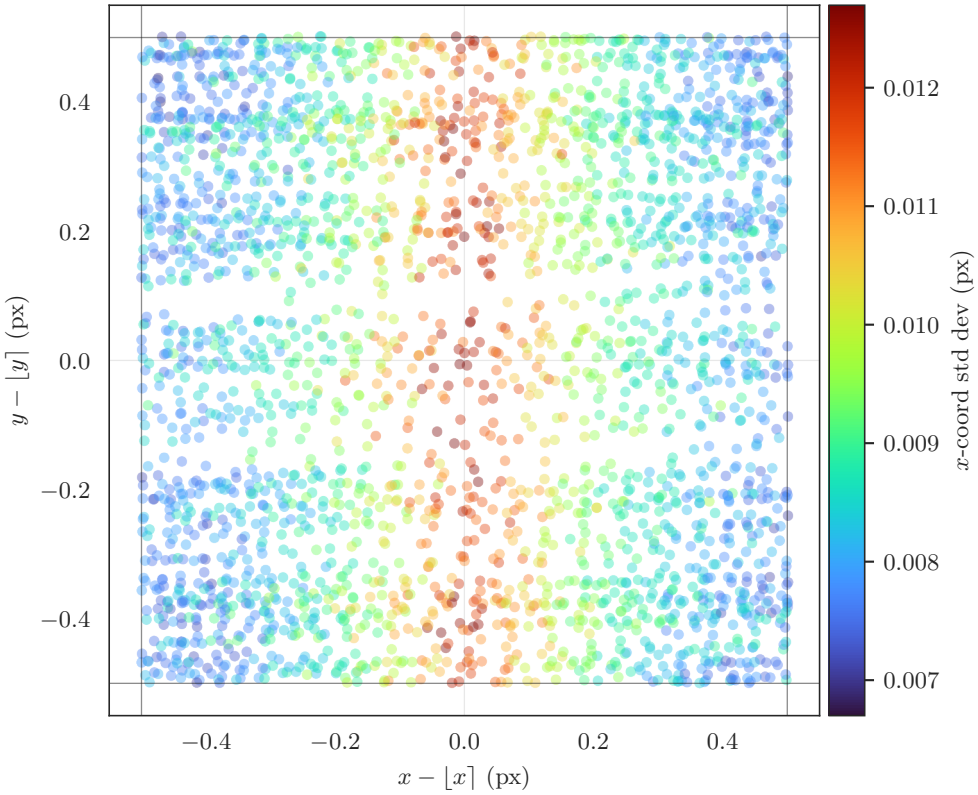
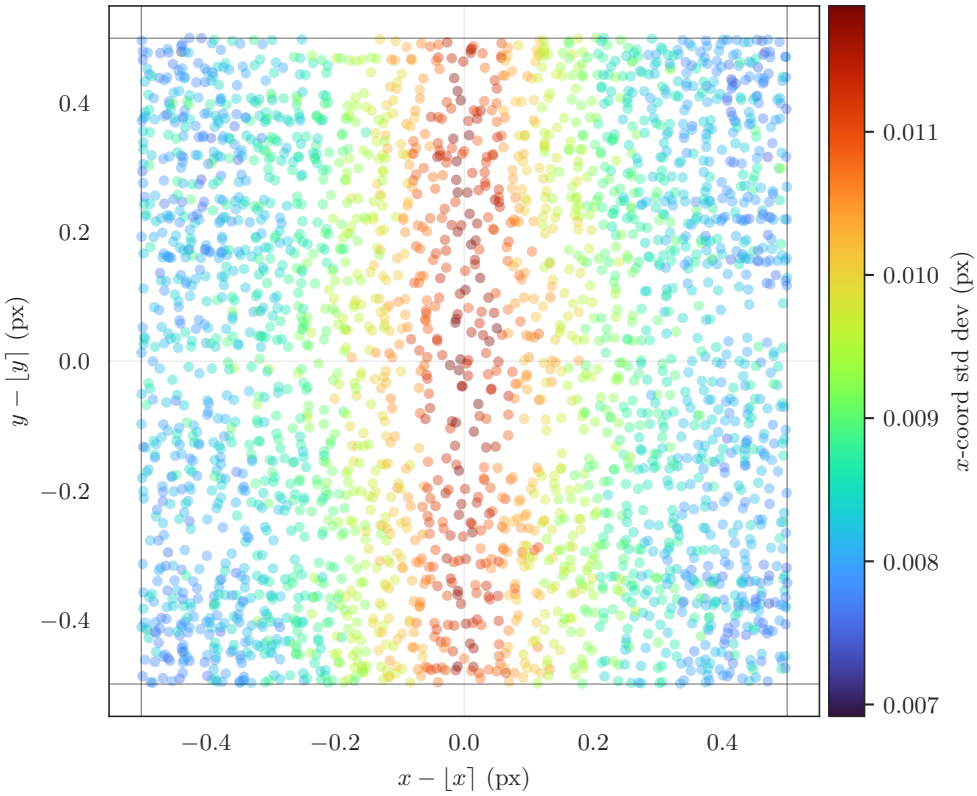


Figure 6.11 A scatter plot of estimated sub-pixel corner locations for the first slider dataset, coloured by their x-coordinate standard deviations. The standard deviations range from around 0.013 px at the pixel centre to 0.0072 px at the edges, a difference of around 1.8×. The gaps are due to the precision with which the checkerboard was aligned with the camera: each band is the path of one row of corners as they move from left to right across the image, and the corresponding vertical motion is only a fraction of a pixel. In Figure 6.12a, where the checkerboard was purposefully rotated by a few degrees in the image plane, the corner y-coordinates are more evenly distributed.

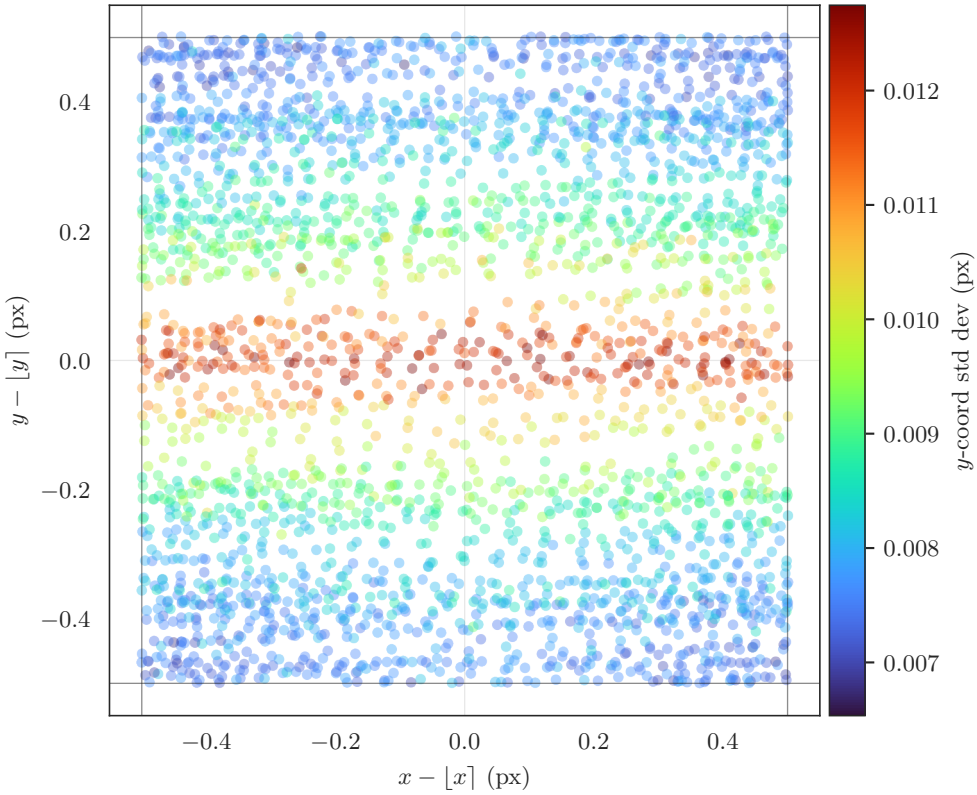


(a) For the second slider dataset.

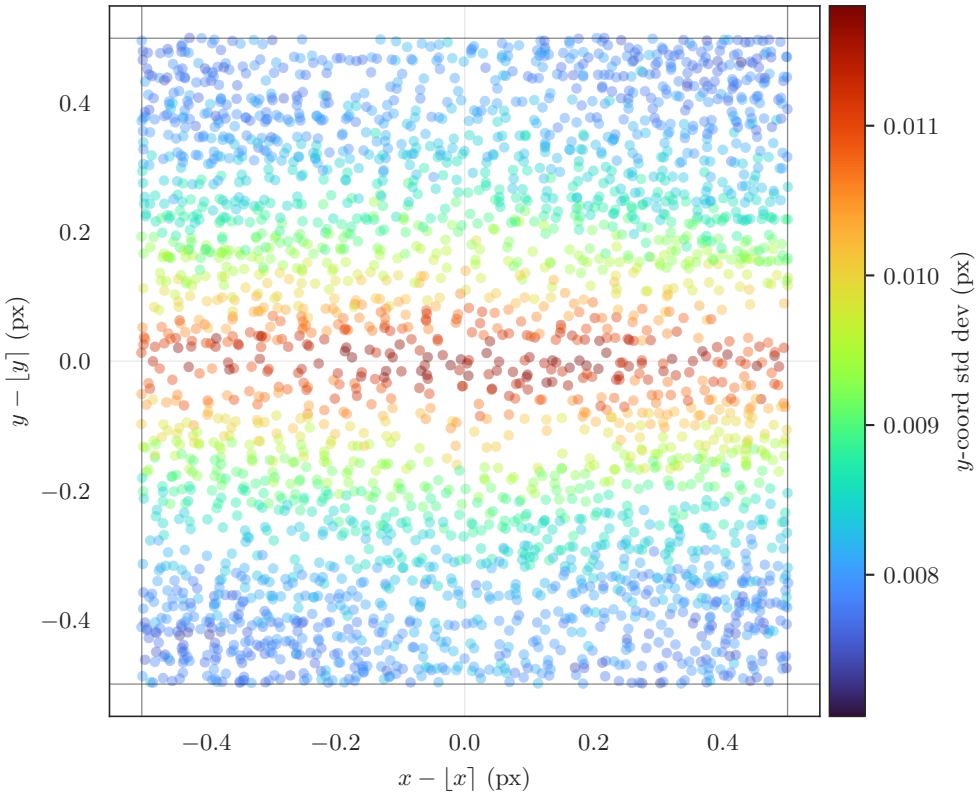


(b) For the simulated dataset.

Figure 6.12 Scatter plots of estimated sub-pixel corner locations for the second slider dataset and the simulated dataset, coloured by their x-coordinate standard deviations. The standard deviations in the real data range from around 0.013 px at the pixel centre to 0.0072 px at the edges, a difference of around 1.8×.



(a) For the second slider dataset.



(b) For the simulated dataset.

Figure 6.13 A scatter plot of estimated sub-pixel corner locations coloured by their y-coordinate standard deviations.

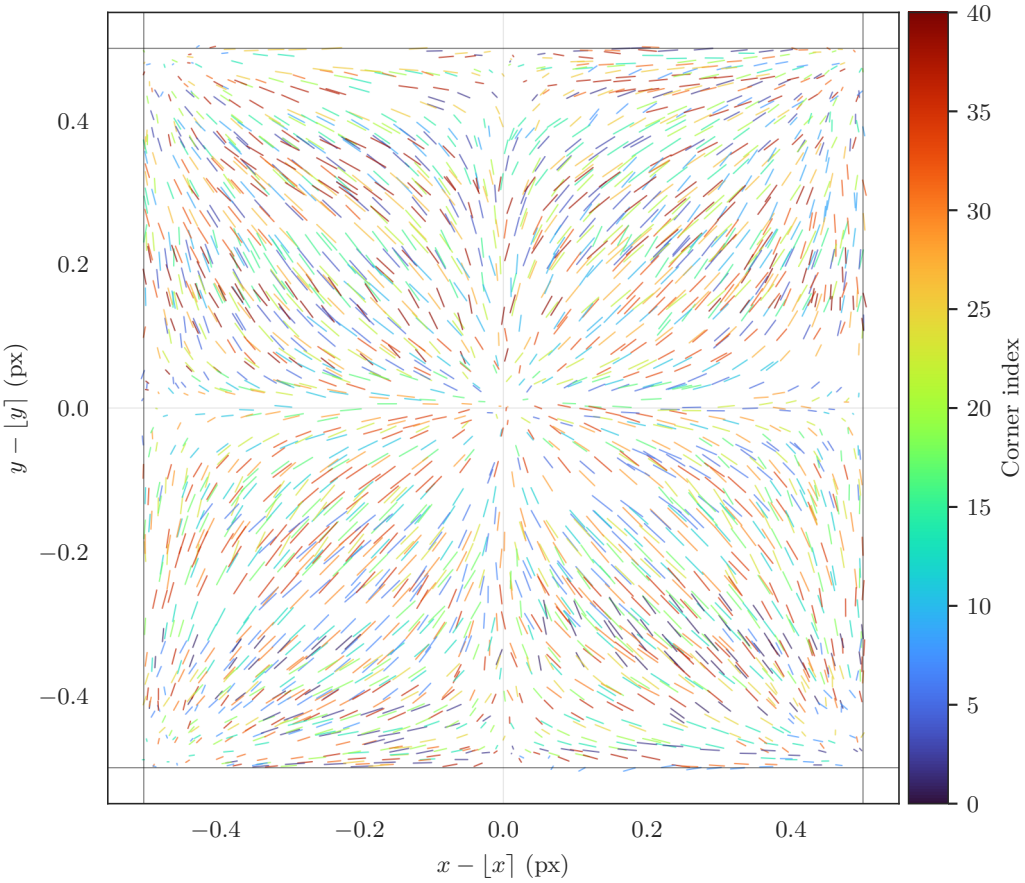


Figure 6.14 A plot showing estimation bias in the simulated dataset. The bias has a maximum magnitude of 0.043 px and an RMS magnitude of 0.026 px. Each line is from the mean estimated location to the projected location for each corner, coloured by the number of the corner it represents (numbered row-wise from top left). The projected corner locations for each frame were calculated by estimating all the corner locations, using them to estimate the checkerboard pose, then using the camera intrinsics (including lens distortion) and the pose to project the (known) 3D corner positions back into pixel coordinates. This has the effect of averaging out the biases of all the individual corner location estimates.

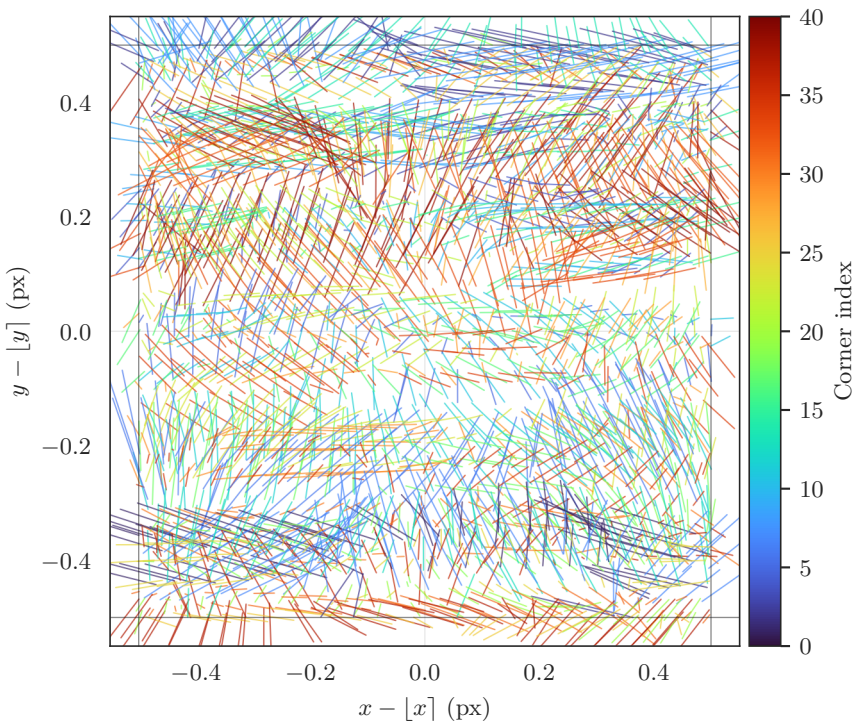


Figure 6.15 A plot showing a combination of estimation bias and residual lens distortion in the second slider dataset. Each line is from the mean estimated corner location to the projected corner location, coloured by the number of the corner it represents (numbered row-wise from top left). This plot does not show the same trend as Figure 6.14 due to insufficiently accurate camera calibration. The bias and residual combined have a maximum magnitude of 0.2 px and an RMS magnitude of 0.088 px.

possible to calibrate out the estimation bias, although achieving this with a real camera would be challenging.[4]

Another way of showing the estimation bias in the real images is to compare the estimated location for each corner at each camera position to the location predicted using the camera position ground truth from the slider. If the checkerboard were aligned perfectly with the slider, then this prediction would be

$$\hat{x}_i = \bar{x}_i + \frac{f_x}{z}\left(c_i - \bar{c}\right), \tag{6.2}$$

where x_i is the x-coordinate of a corner's location (in pixels) at camera position i, $\bar{x}$ denotes the mean, f_x is the camera's x-axis focal length, z is the distance from the camera to the checkerboard (in metres), and c_i is the camera's displacement down the slider at camera position i (in metres). Plotting the prediction residual $\hat{x}_i - x_i$ against sub-pixel location $x_i - \lfloor x_i \rceil$ (as shown in Figure 6.16) then shows a combination of

[4] The magnitude of the bias depends on the sharpness of the image, which for most cameras varies depending on the distance to the object. The bias function may also depend on other parameters, such as the number of corner refinement iterations. Verifying the improvement from estimation bias calibration would require an extremely well-calibrated camera. In most scenarios, a better way of reducing corner detection bias would be to use a more modern corner detection algorithm, as discussed in Section 5.1.

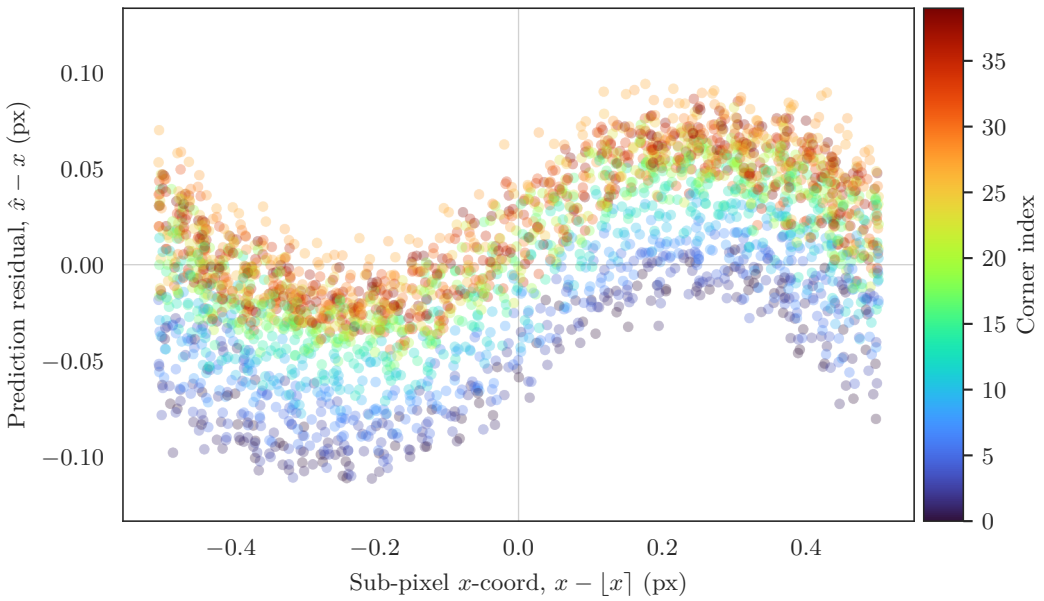


Figure 6.16 A scatter plot showing x-coordinate bias in the second slider dataset (calculated using camera position ground truth) as a function of sub-pixel location, coloured by corner number. The error due to misalignment shows as spread and offset, leaving the bias clearly visible. Note the resemblance to Figure 5.18, and to the biases in particle image velocimetry [e.g., Michaelis et al. 2016, fig. 6].

estimation bias and prediction error due to misalignment. The error due to misalignment shows as spread and offset, leaving the bias clearly visible.

In practice, for pose estimation with sufficiently randomly distributed poses, corner detection error could reasonably be modelled as unbiased. The bias is only noticeable in specific situations, such as the one in Chapter 5 where the corners all move by the same distance in image space each frame.

The goal of this chapter was to validate the results in Chapter 5 using real images, and this goal was achieved, but a limitation of the analysis is the limited set of checkerboard poses used. The approach used here could be repeated with a wider range of checkerboard poses to investigate the variance and bias effects for corners subjected to rotation, perspective distortion, and lens distortion. It could also be extended to replace the robot arm method in Chapter 4, which would result in much more precise camera positioning but require much more effort to set up than an ideal robot arm would.

6.4 CONCLUSION

OpenCV's sub-pixel corner refinement algorithm was found to introduce significant bias and noise which is dependent on the sub-pixel corner location. In real images, the standard deviation of the noise ranged from around 0.013 px at the pixel centre to 0.0072 px at the edges, a difference of around 1.8×. The bias could not be determined from the real images due to residual lens distortion, so the images were reproduced in

simulation, where the bias had a maximum magnitude of 0.043 px. Further research is required to determine the effect this has on pose estimation.

Intensity noise in digital images should not be modelled with identically distributed Gaussian random variables, especially when considering black-and-white targets. For the camera used in this research, the standard deviation of intensity noise was found to vary by a factor of approximately four within the region around a checkerboard corner.

Supersampling should be used when producing simulated imagery if sub-pixel accuracy is required; 8× supersampling was found to have around 1% intensity error when rendering a checkerboard.

Chapter 7

CONCLUSION

A new approach to pose estimation using fiducial markers was proposed that is more precise than existing algorithms. Rather than using only corners or edges for pose estimation, the marker has a radial sinusoid pattern that has a predictable appearance under perspective projection. The marker pose is initially estimated using traditional methods, then refined using a novel optimisation method in which a rendering of the marker is compared with the image. On average, pose estimation precision was increased by 27 % compared to traditional fiducial markers. This approach is a promising avenue for future research.

An analytic model was derived for the CRLB of pose estimation using a checkerboard (or fiducial marker) pose estimator. The model gives a lower bound for the variance (and covariance) of the estimate, which effectively predicts the precision of the most accurate pose estimate from a given set of data. Both a Monte Carlo simulation and real data validate this model. The model can both predict estimator precision before data is available and evaluate the performance of a real estimator on real data compared to the theoretical precision limit. One significant finding is that generalising measurements of pose estimation precision is difficult: results from different cameras are not directly comparable, and the performance for a marker in one pose does not trivially predict the performance for another pose. This approach for predicting and evaluating checkerboard pose estimation precision has not been considered in prior research.

A series of experiments were performed to investigate the error distribution for the OpenCV checkerboard corner detection algorithm. Modelling it using i.i.d. Gaussian random variables was found to be a good first approximation, with the following caveats:

- Corner detection error variance depends on the amount of lens blur in the image, so images that are not equally sharp throughout (for example, images with checkerboards at different depths or images which have been rectified) can have different error variance in different image regions.

- Corner detection in blurry images can produce results with outliers or multiple clusters, in which case a Gaussian distribution is not appropriate.

- Corner detection in images with any amount of motion blur can produce correlated results.

In particular, OpenCV's sub-pixel corner refinement algorithm was found to introduce significant bias and noise which is dependent on the sub-pixel corner location. This is the dominant component of the error in images with little lens blur (less than approximately one pixel). In real images, the standard deviation of the noise ranged from around 0.013 px at the pixel centre to 0.0072 px at the edges, a difference of around 1.8×. The bias could not be determined from the real images due to residual lens distortion, so the images were reproduced in simulation, where the bias had a maximum magnitude of 0.043 px.

Additionally, intensity noise in digital images should not be modelled with identically distributed Gaussian random variables, especially when considering black-and-white targets. For the camera used in this research, the standard deviation of intensity noise was found to vary by a factor of approximately four within the region around a checkerboard corner. Supersampling should be used when producing simulated imagery if sub-pixel accuracy is required; 8× supersampling was found to have around 1% intensity error when rendering a checkerboard.

In summary, the results presented in this thesis are relevant in many areas of computer vision. Section 7.1 gives recommendations for specific aspects which could be explored further.

7.1 IDEAS FOR FUTURE WORK

The work presented here opens many avenues for future research.

Fiducial markers for ground truth

The original problem that spawned the work in this thesis—collecting ground truth for VO in difficult environments—is still open for solutions. The fiducial marker presented in Chapter 3 showed promise; recent papers achieved good results in camera calibration using similar matching-based refinement techniques, albeit with different patterns and more effective algorithms [Hannemose et al. 2019, Schöps et al. 2020]. The results from Pfrommer et al. [2017] suggest that using a network of fiducial markers to achieve greater pose estimation precision than recent VO algorithms requires marker pose estimation precision to increase by at least an order of magnitude. The results from Chapter 4 suggest that although it is possible to further increase precision over traditional fiducial markers, this increase alone is unlikely to make pose estimation using a single fiducial marker sufficient for VO ground truth without requiring the marker to be so large as to dominate the scene. One solution could be to precisely measure

the relative poses of two or more markers and use these measurements as known values to refine the camera pose estimate for an image containing multiple markers; this could give precision more like a single very large marker (albeit with a more complex setup procedure). Another solution worth investigating is markers augmented with lenses or gratings to produce Moiré patterns (as discussed in Chapter 4), as they show significantly increased precision.

Evaluation of state-of-the-art fiducial marker performance

Many of the existing fiducial markers discussed in Chapter 3 claim to improve over one or more of the others in specific ways. The results in Chapter 4, which showed that pose estimation precision is heavily dependent on the marker pose and the camera, lens, and lighting used, suggest that even when publications use the same methodologies in their evaluations (which they generally do not), it is likely the results are not directly comparable. As such, a thorough review comparing the performance of the latest fiducial markers would be a valuable contribution to the field. The evaluation of pose estimation precision could involve extending the precise camera positioning methods used in Chapters 5 and 6, or simply imaging markers with a wide enough variety of poses to sufficiently sample the space of possible poses. Evaluating pose estimation accuracy is more difficult. Relative accuracy could be measured with a robot arm or slider; for absolute accuracy, capturing ground-truthed images using an image-assisted total station could work.

Implications of corner estimation bias

Chapters 5 and 6 found that one common checkerboard corner estimation method exhibits bias and noise that is dependent on sub-pixel corner location. It is likely, although not investigated here, that other methods [e.g., Duda and Frese 2018] have similar issues, especially in light of recent work in generic models for bias-free camera calibration [Schöps et al. 2020]. This may also extend to corner detectors that are not specific to checkerboard corners, such as those used in VO [e.g., Rublee et al. 2011].

One unexplored implication of this is the effect, if any, on pose estimation using the biased corner estimates; investigating this would be a useful next step.

Sub-pixel effects and noise distribution in simulated datasets

In the process of reproducing the experiment from Chapter 6 in simulation, two side experiments were performed: one to characterise the intensity noise distribution of the camera, and another to determine the level of supersampling required to accurately reproduce edges in the images. It would be useful to perform a simulation like the one in Section 6.2.2 with identically distributed intensity noise and then intensity-dependent

noise and compare the results in order to determine whether using a realistic intensity noise distribution is important. This could be extended to include more general scenes, which have less significant intensity changes. It would also be useful to evaluate the level of supersampling required to reproduce more general scenes. Synthetic datasets for VO (and machine learning) are becoming increasingly common, so the effect could be quantified in terms of estimation performance.

REFERENCES

Acuna, R. and Willert, V. (2018), 'Insights into the robustness of control point configurations for homography and planar pose estimation', `arXiv:1803.03025v2 [cs.CV]`.

Albarelli, A., Rodolà, E. and Torsello, A. (2010), 'Robust Camera Calibration using Inaccurate Targets', In *British Machine Vision Conference (BMVC)*, pp. 16.1–16.10.

Armstrong, B.S., Verron, T., Heppe, L.A., Karonde, R.M., Reynolds, J. and Schmidt, K. (2007), 'RGR-6D: Low-cost, High-accuracy Measurement of 6-DOF Pose from a Single Image', *Milwaukee (WI): Manuscript, University of Wisconsin.*

Bailey, D.G., Gilman, A. and Browne, R. (2005), 'Bias Characteristics of Bilinear Interpolation Based Registration', In *TENCON 2005 - 2005 IEEE Region 10 Conference*, Melbourne, Qld., Australia, pp. 1–6.

Banks, S., Green, R. and Junghyun, J. (2019), 'Use of Moiré Patterns in Camera Position Estimation', In *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, Dunedin, New Zealand, pp. 1–7.

Banks, S.J.J. (2020), *Long Range Moiré Patterns*, Master's thesis, University of Canterbury.

Benligiray, B., Topal, C. and Akinlar, C. (2019), 'STag: A stable fiducial marker system', *Image and Vision Computing*, Vol. 89, pp. 158–169.

Bergamasco, F., Albarelli, A., Rodolà, E. and Torsello, A. (2011), 'RUNE-Tag: a High Accuracy Fiducial Marker with Strong Occlusion Resilience', In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 113–120.

Bouguet, J.Y. (1996), 'Camera Calibration Toolbox for Matlab', `http://web.archive.org/web/20020208231430/www.vision.caltech.edu/bouguetj/calib_doc/` [Accessed: 8 February 2002].

Bradski, G. (2000), 'The OpenCV Library', *Dr. Dobb's Journal of Software Tools.*

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M.W. and Siegwart, R. (2016), 'The EuRoC micro aerial vehicle datasets', *International Journal of Robotics Research*, Vol. 35, No. 10, pp. 1157–1163.

Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M.M. and Tardós, J.D. (2020), 'ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM', `arXiv:2007.11898 [cs.RO]`.

Chen, D. and Zhang, G. (2005), 'A New Sub-pixel Detector for X-corners in Camera Calibration Targets', *13th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG 2005)*, pp. 97–100.

Chen, P. and Suter, D. (2009), 'Error analysis in homography estimation by first order approximation tools: A general technique', *Journal of Mathematical Imaging and Vision*, Vol. 33, pp. 281–295.

Cvisic, I. and Petrovic, I. (2015), 'Stereo odometry based on careful feature selection and tracking', In *2015 European Conference on Mobile Robots (ECMR)*, pp. 1–6.

Cvišić, I., Ćesić, J., Marković, I. and Petrović, I. (2018), 'SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles', *Journal of Field Robotics*, Vol. 35, No. 4, pp. 578–595.

Davis, L., Clarkson, E. and Rolland, J.P. (2003), 'Predicting accuracy in pose estimation for marker-based tracking', In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 28–35.

Duda, A. and Frese, U. (2018), 'Accurate detection and localization of checkerboard corners for calibration', In *British Machine Vision Conference*, p. 126.

Edwards, M.J., Hayes, M.P. and Green, R.D. (2016), 'High-accuracy fiducial markers for ground truth', In *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Palmerston North, New Zealand, pp. 1–6.

Edwards, M.J., Hayes, M.P. and Green, R.D. (2017), 'Statistical Lower Bound for Variance of Checkerboard Pose Estimate', In *2017 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Christchurch, New Zealand, pp. 1–6.

Edwards, M.J., Hayes, M.P. and Green, R.D. (2018), 'Error Distribution of Estimated Checkerboard Corner Location', In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Auckland, New Zealand, pp. 1–6.

Edwards, M.J., Hayes, M.P. and Green, R.D. (2020), 'Experimental Validation of Bias in Checkerboard Corner Detection', In *2020 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, Wellington, New Zealand, pp. 1–6.

Engel, J., Sturm, J. and Cremers, D. (2012), 'Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing', In *Proceedings of the of the Workshop on Visual Control of Mobile Robots (ViCoMoR) at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, Vilamoura, Portugal.

Fiala, M. (2005), 'ARTag Revision 1. A Fiducial Marker System Using Digital Techniques', In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2, pp. 590 – 596.

Fischler, M.A. and Bolles, R.C. (1981), 'Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography', *Communications of the ACM*, Vol. 24, No. 6, pp. 381–395.

Foi, A., Trimeche, M., Katkovnik, V. and Egiazarian, K. (2008), 'Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data', *IEEE Transactions on Image Processing*, Vol. 17, No. 10, pp. 1737–1754.

Forne, C. (2007), *3-D Scene Reconstruction from Multiple Photometric Images*, PhD thesis, University of Canterbury.

Förstner, W. and Gülch, E. (1987), 'A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features', In *ISPRS Intercommission Workshop*, Interlaken, Switzerland, pp. 281–305.

Fraundorfer, F., Heng, L., Honegger, D., Lee, G.H., Meier, L., Tanskanen, P. and Pollefeys, M. (2012), 'Vision-based autonomous mapping and exploration using a quadrotor MAV', In *IEEE International Conference on Intelligent Robots and Systems*, pp. 4557–4564.

Furgale, P., Rehder, J. and Siegwart, R. (2013), 'Unified temporal and spatial calibration for multi-sensor systems', In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Tokyo, pp. 1280–1286.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. and Marín-Jiménez, M. (2014), 'Automatic generation and detection of highly reliable fiducial markers under occlusion', *Pattern Recognition*, Vol. 47, No. 6, pp. 2280 – 2292.

Geiger, A., Lenz, P., Stiller, C. and Urtasun, R. (2012a), 'KITTI Visual Odometry / SLAM Evaluation 2012', `http://www.cvlibs.net/datasets/kitti/eval_odometry.php` [Accessed: 3 March 2021].

Geiger, A., Lenz, P. and Urtasun, R. (2012b), 'Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite', In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361.

Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A. and Rubin, D.B. (2013), *Bayesian Data Analysis*, Chapman and Hall/CRC, New York, 3rd ed.

Geneva, P., Eckenhoff, K., Lee, W., Yang, Y. and Huang, G. (2019), 'OpenVINS: A Research Platform for Visual-Inertial Estimation', *IROS 2019 Workshop on Visual-Inertial Navigation: Challenges and Applications*.

Gonzales, R.C. and Fittes, B.A. (1977), 'Gray-level transformations for interactive image enhancement', *Mechanism and Machine Theory*, Vol. 12, No. 1, pp. 111–122.

Grishin, V.A. (2010), 'Accuracy of Measuring Camera Position by Marker Observation', *Journal of Software Engineering and Applications*, Vol. 03, No. 10, pp. 906–913.

Ha, H., Perdoch, M., Alismail, H., Kweon, I.S. and Sheikh, Y. (2017), 'Deltille Grids for Geometric Camera Calibration', In *IEEE International Conference on Computer Vision*, Venice, Italy, pp. 5354–5362.

Hannemose, M., Wilm, J. and Frisvad, J.R. (2019), 'Superaccurate camera calibration via inverse rendering', In B. Bodermann, K. Frenner and R.M. Silver (editors), *Modeling Aspects in Optical Metrology VII*, Vol. 11057, SPIE, pp. 252–260.

Harris, C. and Stephens, M. (1988), 'A Combined Corner and Edge Detector', *Procedings of the Alvey Vision Conference*, pp. 147–151.

Hartley, R.I. and Zisserman, A. (2004), *Multiple View Geometry in Computer Vision*, Cambridge University Press, New York, NY, USA, 2nd ed.

Huang, T.S. and Netravali, A.N. (1994), 'Motion and Structure from Feature Correspondences: A Review', *Proceedings of the IEEE*, Vol. 82, No. 2, pp. 252–268.

Intel Corporation (2001), 'Open Source Computer Vision Library Contributors', `https://web.archive.org/web/20010822162745/http://www.intel.com/research/mrl/research/opencv/contributors.htm` [Accessed: 22 August 2001].

Jähne, B. (2004), 'Interpolation', In *Practical Handbook on Image Processing for Scientific and Technical Applications*, chap. 8, pp. 274–285, CRC Press, 2nd ed.

Kaartinen, H., Hyyppä, J., Vastaranta, M., Kukko, A., Jaakkola, A., Yu, X., Pyörälä, J., Liang, X., Liu, J., Wang, Y., Kaijaluoto, R., Melkas, T., Holopainen, M. and Hyyppä, H. (2015), 'Accuracy of Kinematic Positioning Using Global Satellite Navigation Systems under Forest Canopies', *Forests*, Vol. 6, No. 9, pp. 3218–3236.

Kanatani, K. and Ohta, N. (2001), 'Accuracy bounds and optimal computation of robot localization', *Machine Vision and Applications*, Vol. 13, No. 2, pp. 51–60.

Kato, H. and Billinghurst, M. (1999), 'Marker tracking and HMD calibration for a video-based augmented reality conferencing system', In *Proceedings of the IEEE and ACM International Workshop on Augmented Reality*, pp. 85–94.

Kay, S.M. (1993), *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1st ed.

Ke, T. and Roumeliotis, S.I. (2017), 'An efficient algebraic solution to the perspective-three-point problem', In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4618–4626.

Kircher, A. (1646), *Ars Magna Lucis Et Umbra.*

Klette, R. (2014), *Concise Computer Vision: An Introduction into Theory and Algorithms*, Springer Publishing Company, Incorporated.

Kneip, L., Chli, M. and Siegwart, R.Y. (2011a), 'Robust Real-Time Visual Odometry with a Single Camera and an IMU', In *Proceedings of the British Machine Vision Conference*, BMVA Press, pp. 16.1–16.11.

Kneip, L., Scaramuzza, D. and Siegwart, R. (2011b), 'A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation', In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2969–2976.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W. (2011), 'G2o: A general framework for graph optimization', In *IEEE International Conference on Robotics and Automation*, IEEE, Shanghai, pp. 3607–3613.

Lepetit, V., Moreno-Noguer, F. and Fua, P. (2009), 'EPnP: An accurate O(n) solution to the PnP problem', *International Journal of Computer Vision*, Vol. 81, No. 2, pp. 155–166.

Levenberg, K. (1944), 'A Method for the Solution of Certain Non-linear Problems in Least Squares', *Quarterly of Applied Mathematics*, Vol. 2, No. 2, pp. 164–168.

Lightbody, P., Krajník, T. and Hanheide, M. (2017), 'An efficient visual fiducial localisation system', *ACM SIGAPP Applied Computing Review*, Vol. 17, No. 3, pp. 28–37.

Lu, C.P., Hager, G.D. and Mjolsness, E. (2000), 'Fast and globally convergent pose estimation from video images', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 6, pp. 610–622.

Lucchese, L. and Mitra, S.K. (2002), 'Using saddle points for subpixel feature detection in camera calibration targets', *IEEE Asia-Pacific Conference on Circuits and Systems, Proceedings, APCCAS*, Vol. 2, pp. 191–195.

Lynch, K.M. and Park, F.C. (2017), *Modern Robotics: Mechanics, Planning, and Control*, Cambridge University Press.

Mallon, J. and Whelan, P.F. (2007), 'Which pattern? Biasing aspects of planar calibration patterns and detection methods', *Pattern Recognition Letters*, Vol. 28, No. 8, pp. 921–930.

Marquardt, D.W. (1963), 'An Algorithm for the Least-Squares Estimation of Nonlinear Parameters', *SIAM Journal of Applied Mathematics*, Vol. 11, No. 2, pp. 431–441.

Matsunaga, C. and Kanatani, K. (2000), 'Calibration of a Moving Camera Using a Planar Pattern: Optimal Computation, Reliability Evaluation, and Stabilization by Model Selection', In D. Vernon (editor), *Proceedings of the 6th European Conference on Computer Vision*, Springer, Berlin, Heidelberg, Dublin, Ireland, pp. 595–609.

Meurer, A., Smith, C.P., Paprocki, M., Čertík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., Rathnayake, T., Vig, S., Granger, B.E., Muller, R.P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M.J., Terrel, A.R., Roučka, Š., Saboo, A., Fernando, I., Kulal, S., Cimrman, R. and Scopatz, A. (2017), 'SymPy: symbolic computing in Python', *PeerJ Computer Science*, Vol. 3, p. 103.

Michaelis, D., Neal, D.R. and Wieneke, B. (2016), 'Peak-locking reduction for particle image velocimetry', *Measurement Science and Technology*, Vol. 27, No. 10, p. 104005.

Mihalyi, R.G., Pathak, K., Vaskevicius, N. and Birk, A. (2013), 'Uncertainty estimation of AR-marker poses for graph-SLAM optimization in 3D object model generation with RGBD data', In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, IEEE, Tokyo, pp. 1807–1813.

Moravec, H.P. (1980), 'Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover', Tech. rep., Carnegie Mellon University, Pittsburgh, PA.

Nelder, J.A. and Mead, R. (1965), 'A Simplex Method for Function Minimization', *The Computer Journal*, Vol. 7, No. 4, pp. 308–313.

Olson, E. (2011), 'AprilTag: A robust and flexible visual fiducial system', In *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, pp. 3400 – 3407.

Pfrommer, B., Sanket, N., Daniilidis, K. and Cleveland, J. (2017), 'PennCOSYVIO: A challenging Visual Inertial Odometry benchmark', In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Singapore, pp. 3847–3854.

Qin, T., Li, P. and Shen, S. (2018), 'VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator', *IEEE Transactions on Robotics*, Vol. 34, No. 4, pp. 1004–1020.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. and Ng, A. (2009), 'ROS: an open-source Robot Operating System', In *ICRA Workshop on Open Source Software*, IEEE, Kobe, Japan.

Raffel, M., Willert, C.E., Scarano, F., Kähler, C.J., Wereley, S.T. and Kompenhans, J. (2018), 'Peak Locking', In *Particle Image Velocimetry: A Practical Guide*, chap. 6, pp. 219–224, Springer International Publishing.

Rao, P.R. (2013), *Communication Systems*, Tata McGraw-Hill Education.

Rife, D. and Boorstyn, R. (1974), 'Single tone parameter estimation from discrete-time observations', *IEEE Transactions on Information Theory*, Vol. 20, No. 5, pp. 591–598.

Rihaczek, A.W. (1996), *Principles of high-resolution radar*, Artech House, Norwood, MA.

Roberts, A., Browne, W.N. and Hollitt, C. (2015), 'Accurate Marker Based Distance Measurement with Single Camera', In *Proceedings of the 30th International Conference on Image and Vision Computing New Zealand*, IEEE.

Robinson, D. and Milanfar, P. (2004), 'Fundamental Performance Limits in Image Registration', *IEEE Transactions on Image Processing*, Vol. 13, No. 9, pp. 1185–1199.

Rohde, J., Stellet, J.E., Mielenz, H. and Zollner, J.M. (2016), 'Localization accuracy estimation with application to perception design', In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4777–4783.

Rosinol, A., Abate, M., Chang, Y. and Carlone, L. (2020), 'Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping', *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1689–1696.

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G. (2011), 'ORB: An efficient alternative to SIFT or SURF', In *IEEE International Conference on Computer Vision*, IEEE, Barcelona, pp. 2564–2571.

Scaramuzza, D. and Fraundorfer, F. (2011), 'Visual Odometry', *IEEE Robotics and Automation*, Vol. 18, No. 4.

Schöps, T., Schönberger, J.L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M. and Geiger, A. (2017), 'A Multi-view Stereo Benchmark with High-Resolution Images and Multi-camera Videos', In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Honolulu, HI, USA, pp. 2538–2547.

Schöps, T., Sattler, T. and Pollefeys, M. (2019), 'BAD SLAM: Bundle Adjusted Direct RGB-D SLAM', In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 134–144.

Schöps, T., Larsson, V., Pollefeys, M. and Sattler, T. (2020), 'Why Having 10,000 Parameters in Your Camera Model Is Better Than Twelve', In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2532–2541.

Schubert, D., Goll, T., Demmel, N., Usenko, V., Stückler, J. and Cremers, D. (2018), 'The TUM VI Benchmark for Evaluating Visual-Inertial Odometry', In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1680–1687.

Shen, S., Mulgaonkar, Y., Michael, N. and Kumar, V. (2013), 'Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor', *Robotics: Science and Systems.*

Shen, S., Michael, N. and Kumar, V. (2015), 'Tightly-Coupled Monocular Visual-Inertial Fusion for Autonomous Flight of Rotorcraft MAVs', In *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Seattle, Washington, pp. 5303–5310.

Shi, J. and Tomasi, C. (1994), 'Good features to track', In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600.

Siebert, J.P. and Bogusław, C. (2009), *An Introduction to 3D Computer Vision Techniques and Algorithms*, John Wiley & Sons.

Strobl, K.H. and Hirzinger, G. (2008), 'More accurate camera and hand-eye calibrations with unknown grid pattern dimensions', In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1398–1405.

Suzuki, S. and Abe, K. (1985), 'Topological structural analysis of digitized binary images by border following', *Computer Vision, Graphics, and Image Processing*, Vol. 30, No. 1, pp. 32–46.

Swapna, P., Krouglicof, N. and Gosine, R. (2009), 'The question of accuracy with geometric camera calibration', In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pp. 541–546.

Szeliski, R. (2011), *Computer Vision: Algorithms and Applications*, Springer, London.

Tanaka, H., Sumi, Y. and Matsumoto, Y. (2012), 'A high-accuracy visual marker based on a microlens array', In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4192–4197.

Tanaka, H., Kajitani, I., Homma, K., Wakita, Y. and Matsumoto, Y. (2015), 'A Motion Tracker Using High-Accuracy AR Markers for On-site Motion Analysis', In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 1427–1432.

Tanaka, H., Ogata, K. and Matsumoto, Y. (2017), 'Solving pose ambiguity of planar visual marker by wavelike two-tone patterns', In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 568–573.

Terzakis, G. and Lourakis, M. (2020), 'A Consistently Fast and Globally Optimal Solution to the Perspective-n-Point Problem', In A. Vedaldi, H. Bischof, T. Brox and J.M. Frahm (editors), *Computer Vision – ECCV 2020. Lecture Notes in Computer Science*, Vol. 12346, Springer, Cham, pp. 478–494.

Terzakis, G., Lourakis, M. and Ait-Boudaoud, D. (2018), 'Modified Rodrigues Parameters: An Efficient Representation of Orientation in 3D Vision and Graphics', *Journal of Mathematical Imaging and Vision*, Vol. 60, No. 3, pp. 422–442.

Thode, H.C. (2002), 'Percent-percent plots', In *Testing for Normality*, chap. 2, pp. 23–24, CRC Press.

Uss, M.L., Vozel, B., Dushepa, V.A., Komjak, V.A. and Chehdi, K. (2014), 'A precise lower bound on image subpixel registration accuracy', *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 52, No. 6, pp. 3333–3345.

Wagner, D. and Schmalstieg, D. (2007), 'ARToolKitPlus for Pose Tracking on Mobile Devices', In *Proceedings of 12th Computer Vision Winter Workshop CVWW07*, pp. 139–146.

Wang, D., Jiang, Y., Wang, W. and Wang, Y. (2016), 'Bias reduction in sub-pixel image registration based on the anti-symmetric feature', *Measurement Science and Technology*, Vol. 27, No. 3, p. 35206.

Watanabe, S. (2009), *Algebraic Geometry and Statistical Learning Theory*, Cambridge University Press, New York.

Wilm, J. (2018), 'Calibration Best Practices', `https://calib.io/blogs/knowledge-base/calibration-best-practices` [Accessed: 14 April 2021].

Xu, A. and Dudek, G. (2011), 'Fourier Tag: A Smoothly Degradable Fiducial Marker System with Configurable Payload Capacity', In *2011 Canadian Conference on Computer and Robot Vision*, pp. 40–47.

Zhang, J. and Singh, S. (2015), 'Visual-lidar Odometry and Mapping: Low-drift, Robust, and Fast', In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, pp. 2174–2181.

Zhang, Z. (2000), 'A flexible new technique for camera calibration', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 11, pp. 1330–1334.

Zhang, Z. and Scaramuzza, D. (2018), 'Perception-aware Receding Horizon Navigation for MAVs', In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 2534–2541.

Zhang, Z. and Scaramuzza, D. (2019), 'Beyond Point Clouds: Fisher Information Field for Active Visual Localization', In *2019 International Conference on Robotics and Automation (ICRA)*, Vol. 2019-May, IEEE, pp. 5986–5992.

Appendix A

THE FÖRSTNER OPERATOR

The only available version of Förstner and Gülch's [1987] manuscript is the original typewritten one, so the following lightly edited excerpts are provided for context.

A.1 MATHEMATICAL MODEL

The task can be written as a least squares problem in the form of a Gauss-Markov model for the n observed values contained in the vector $\mathbf{x}$ and the u unknowns contained in the vector $\mathbf{y}$:

$$\mathbf{x} + \mathbf{e} = \mathbf{A}\mathbf{y}, \quad \mathbf{D}(\mathbf{x}) = \mathbf{C} = \sigma^2 \mathbf{W}^{-1}, \tag{A.1}$$

with normal equations for the estimates, $\mathbf{y}$,

$$\mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{A}\hat{\mathbf{y}} = \mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{x}, \tag{A.2}$$

and the estimate for the variance factor,

$$\sigma_0^2 = \mathbf{e}^{\mathrm{T}}\mathbf{W}\mathbf{e}/r, \tag{A.3}$$

derived from the residuals, $\mathbf{e}$, where $r = n - u$ is the redundancy of the system. The $n \times u$ design matrix, $\mathbf{A}$, is assumed to have full rank. The weight matrix, $\mathbf{W}$, is assumed to be known. It may be derived from the variances, c_{ii}, in the covariance matrix, $\mathbf{C}$, assuming an arbitrary variance factor, σ_0^2.

The only unknowns are the row, r_0, and the column, c_0, of a point, thus $\mathbf{y}^{\mathrm{T}} = (r_0, c_0)$. Each pixel, (r, c), within a small window, $g(r, c)$, say between 5×5 and 16×16 pixels, contributes to the solution in the same manner.

A.1.1 Intersection of edge elements, corners

Let the edge element (*edgel*) at each pixel be defined as a straight line passing through the centre of the pixel with an orientation derived from the gradient, $\mathbf{v}_g^{\mathrm{T}}(r, c) = (g_r(r, c), g_c(r, c))$, using any appropriate operator for determining the partial derivatives of g (see Figure A.1). A corner, $C(r_0, c_0)$, can then be estimated from the intersection of all edgels. The straight line can be represented by

$$r \cos \Phi + c \sin \Phi - l = 0, \tag{A.4}$$

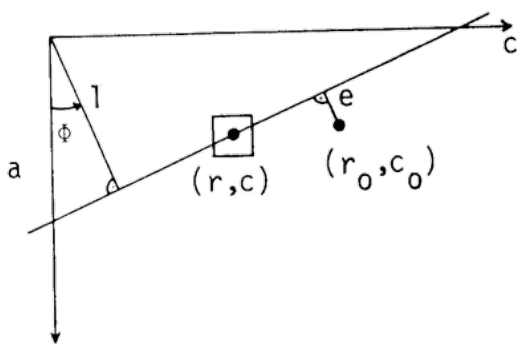


Figure A.1 Edge element (edgel) at position (r, c) for determining the intersection point (r_o, c_o) [Förstner and Gülch 1987].

where l is the distance of the origin from the line and Φ is the angle of this direction. Note that $\mathbf{v}_g^{\mathrm{T}} = |\mathbf{v}_g| \cdot (\cos\Phi, \sin\Phi)$. The linear model for this intersection point can then be written as

$$l(r, c) + e_l(r, c) = r_0 \cos\Phi(r, c) + c_0 \sin\Phi(r, c). \tag{A.5}$$

The weight of the edgel is intuitively proportional to the absolute gradient square,

$$w_l(r, c) = |\mathbf{v}_g|^2. \tag{A.6}$$

This can be proven by again assuming the variance of the gray-level noise to be σ_n^2, thus constant, and observing that $|v_g| = |\mathrm{d}g/\,\mathrm{d}l|$ thus $\sigma_l = \sigma_n/|v_g|$.

A.1.2 Weighted centre of gravity

Let each pixel in a window contribute to the centre of gravity of that window by using the gradient as weight. We immediately obtain the linear model

$$\begin{aligned} r + e_r &= r_0 \\ c + e_c &= c_0. \end{aligned} \tag{A.7}$$

The weight of each coordinate, r and c, depends on the direction of the local gradient $\mathbf{v_g}(r, c)$. By rotating the vector $(|\mathbf{v}_g|, 0)$ into $\mathbf{v}_g(r, c)$ using the rotation matrix $\mathbf{R}_\Phi$, one obtains the weight matrix for the pixel (r, c),

$$\begin{aligned} \mathbf{W}_{rc}(r, c) &= |\mathbf{v}_g|^2 \begin{bmatrix} \cos^2\Phi & \cos\Phi\sin\Phi \\ \cos\Phi\sin\Phi & \sin^2\Phi \end{bmatrix} \\ &= \mathbf{v}_g \cdot \mathbf{v}_g^{\mathrm{T}} \\ &= \begin{bmatrix} g_r^2(r, c) & g_r(r, c)g_c(r, c) \\ g_r(r, c)g_c(r, c) & g_c^2(r, c) \end{bmatrix}. \end{aligned} \tag{A.8}$$

If, for example, the edge is horizontal, then $\Phi = 0$ and only the row coordinate contributes to the centre of gravity.

The derivation of the singular weight matrix uses the propagation of weight matrices. If $\mathbf{x}$ has covariance matrix $\mathbf{C}_{xx}$, the covariance of $\mathbf{y} = \mathbf{A}\mathbf{x}$ is $\mathbf{C}_{yy} = \mathbf{A}\mathbf{C}_{xx}\mathbf{A}^{\mathrm{T}}$. Thus we

can use $\mathbf{W}_{yy} = \mathbf{A}^{-\mathrm{T}}\,\mathbf{W}_{xx}\,\mathbf{A}^{-1}$, if an inverse of $\mathbf{A}$ exists. The gradient $\mathbf{v}_g$ results from rotation of $\mathbf{e} = (|\mathbf{v}_g|, 0)$ by

$$\mathbf{R}_\Phi = \begin{bmatrix} \cos\Phi & -\sin\Phi \\ \sin\Phi & \cos\Phi \end{bmatrix}, \tag{A.9}$$

thus $\mathbf{v}_g = \mathbf{R}_\Phi \cdot (|\mathbf{v}_g|, 0)$. If now the component of $\mathbf{e}$ in the row direction has weight $|\mathbf{v}_g|^2$ and the component in the column direction has weight zero then

$$\mathbf{W}_{ee} = \begin{bmatrix} |\mathbf{v}_g|^2 & 0 \\ 0 & 0 \end{bmatrix}. \tag{A.10}$$

With $\mathbf{A} = \mathbf{R}_\Phi$, this finally leads to $\mathbf{W}_{rc} = (\mathbf{R}_\Phi)^{-\mathrm{T}}\ \mathbf{W}_{ee}\ (\mathbf{R}_\Phi)^{-1}$ in (A.8).

A.1.3 Normal equations

The normal equation systems for the intersection of all edgels and the weighted centre of gravity are the same:

$$\begin{bmatrix} \sum g_r^2 & \sum g_r g_c \\ \sum g_r g_c & \sum g_c^2 \end{bmatrix} \begin{bmatrix} \hat{r}_0 \\ \hat{c}_0 \end{bmatrix} = \begin{bmatrix} \sum g_r^2\, r + \sum g_r g_c\, c \\ \sum g_r g_c\, r + \sum g_c^2\, c \end{bmatrix}. \tag{A.11}$$

The sums have to be taken over all pixels within the window.

A.2 DISCUSSION

The weighted centre of gravity is identical to the intersection of the edgels. Both interpretations have their advantage. The intersection point is geometrically intuitive. The formulation as weighted centre of gravity is more simple, as the design matrix, A, consists of only 2×2 unit matrices.

The intersection of edgels can also be interpreted as linear regression in Hough space. Each edgel at (r, c) corresponds to a point, $(\tan\Phi(r,c),\ l(r,c)/\cos\Phi(r,c))$, in Hough space. The edgels of one edge form a cluster in Hough space. If several edges intersect, the corresponding clusters lie on a straight line. The model used here is to take the slope, $\tan\Phi$, of the edgel as fixed and the intercept, $a = l/\cos\Phi$, as observed value (see Figure A.1), with a standard deviation $\sigma_a = \sigma_l/\cos\Phi = 1/|g_r|$. Then the linear model for the fitting line in Hough space can be written as

$$a(r,c) + e_a(r,c) = r_0 + c_0 \tan\Phi(r,c) \tag{A.12}$$

with weights

$$w_a(r,c) = g_r^2(r,c), \tag{A.13}$$

which leads to the same normal equation system as (A.5) and (A.6).