YIN-HAN CHUNG

MODELLING THE XBOX 360 KINECT FOR VISUAL SERVO CONTROL APPLICATIONS

MODELLING THE XBOX 360 KINECT FOR VISUAL SERVO CONTROL APPLICATIONS

YIN-HAN CHUNG

A research report submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, August 2016

Yin-Han Chung: *Modelling The Xbox 360 Kinect for Visual Servo Control Applications,* M.Sc. Research Report, August 2016

SUPERVISORS:
Jacques Arno Naudé
Michaël Antonie van Wyk

DEPARTMENT:
School of Electrical and Information Engineering

FACULTY:
Engineering and the Built Environment

INSTITUTION:
University of the Witwatersrand, Johannesburg

DECLARATION

I declare that this research report is my own unaided work, other than where specifically acknowledged. It is being submitted for the degree of Master of Science in Engineering to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other university.

. . . . *day of* . . . . *year* . . . .

Yin-Han Chung

ABSTRACT

There has been much interest in using the Microsoft Xbox 360 Kinect cameras for visual servo control applications. It is a relatively cheap device with expected shortcomings. This work contributes to the practical considerations of using the Kinect for visual servo control applications. A comprehensive characterisation of the Kinect is synthesised from existing literature and results from a nonlinear calibration procedure. The Kinect reduces computational overhead on image processing stages, such as pose estimation or depth estimation. It is limited by its $0.8\,\mathrm{m}$ to $3.5\,\mathrm{m}$ practical depth range and quadratic depth resolution of $1.8\,\mathrm{mm}$ to $35\,\mathrm{mm}$, respectively. Since the Kinect uses an infra-red (IR) projector, a class one laser, it should not be used outdoors, due to IR saturation, and objects belonging to classes of non-IR-friendly surfaces should be avoided, due to IR refraction, absorption, or specular reflection. Problems of task stability due to invalid depth measurements in Kinect depth maps and practical depth range limitations can be reduced by using depth map preprocessing and activating classical visual servoing techniques when Kinect-based approaches are near task failure.

To my parents,
for their love and support.

ACKNOWLEDGMENTS

I wish to thank my supervisors, Jacques Naudé and Antonie van Wyk, for their invaluable support and guidance.

My research has been, many a time, set on a one-way flight back to nowhere, which was averted with Jacques' paranoid optimism. You have my most sincere gratitude.

Additionally, I wish to acknowledge:

- Mark Goosens of the Wits Genmin Laboratory, for machining the wooden calibration board.
- All the folks who spent time in the coffee room, for their thoughtful conversations.
- André Miede, for the ClassicThesis LaTeX template on which this document is based.[1]
- The National Research Foundation (NRF) of South Africa, for the funding of this research.[2]

1 http://www.miede.de
2 http://www.nrf.ac.za/

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

α_0, α_1	depth distortion α coefficients
$\mathbf{a}$	camera characteristics
c_0, c_1	Kinect disparity coefficients
$\{C\}$	coordinate frame
δ_r	additive radial lens distortion
Δ_r	multiplicative radial lens distortion
δ_s	small error due to finite bandwidth
δ_t	tangential lens distortion
$\mathbf{D}_\delta(u, v)$	depth distortion β coefficients
d	Kinect disparity [kdu]
d_k	undistorted Kinect disparity [kdu]
$\mathbf{e}(t)$	task function error
f	focal length [mm]
f	focal point
H	image sensor vertical pixel count [px]
$\mathbf{I}$	identity matrix
k_1, k_2, k_3	radial lens distortion coefficients
$\mathbf{K}$	camera matrix
$\mathbf{K}_p$	projection matrix
λ	controller gain
$\mathbf{L}$	image Jacobian, or interaction matrix
$\mathbf{L}^+$	pseudo-inverse of the interaction matrix
$\mathbf{m}(\mathbf{t})$	set of image measurements
$\boldsymbol{\nu}$	camera linear velocity
$\boldsymbol{\omega}$	camera angular velocity
Ω_a	available bandwidth
Ω_c	corner frequency of the desired sensitivity function
p	pole of a transfer function
p_1, p_2	tangential lens distortion coefficients
$\mathbf{p}$	two-dimensional image point
$\bar{\mathbf{p}}$	augmented image point
$\mathbf{p}_n$	normalised image point

$\mathbf{p}_{dn}$	distorted normalised image point
$\mathbf{P}$	three-dimensional world point
$\bar{\mathbf{P}}$	augmented world point
P_h	vertical camera spatial resolution [$m/px/m$]
P_w	horizontal camera spatial resolution [$m/px/m$]
ρ_h	vertical pixel pitch [m/px]
ρ_w	horizontal pixel pitch [m/px]
$\mathbf{R}$	rotation matrix
$\mathbf{s}$	set of visual features
$S(j\omega)$	sensitivity function
S_{min}	smallest sensitivity penalty
$\theta\mathbf{u}$	Euler axis/angle
θ_h	horizontal field of view
θ_v	vertical field of view
θ	angle of rotation
t	scalar translation
$[\mathbf{t}]_\times$	cross product matrix of a vector **t**
$\mathbf{T}$	homogeneous transformation, or pose, matrix
${}^{A}_{B}\mathbf{T}$	relative pose of frame {B} with respect to {A}
$\mathbf{X}$	translation matrix
$\boldsymbol{\upsilon}$	camera spatial velocity
u_0	horizontal principle point [px]
v_0	vertical principle point [px]
W	image sensor horizontal pixel count [px]
ξ	relative pose
$z(d)$	Kinect depth [m]
$z_k(d)$	undistorted Kinect depth [m]
z_w	distance between world point and lens [mm]
z_i	distance between lens and image point [mm]

ACRONYMS

DOF	Degree of Freedom
EKF	Extended Kalman Filter
fps	frames per second
GPU	Graphics Processing Unit
IBVS	Image-Based Visual Servoing
ICP	Iterative Closest Point
IR	infra-red
KCT	Kinect Calibration Toolbox
kdu	Kinect disparity units
NUI	Natural User Interface
OpenCV	Open Source Computer Vision
PBVS	Position-Based Visual Servoing
PCL	Point Cloud Library
PGM	Portable Grey Map
PPM	Portable Pix Map
px	pixel
RGB	Red Green Blue, primary colour model
ROS	Robot Operating System
SDK	Software Developer Kit
SURF	Speeded Up Robust Features
URL	Uniform Resource Locator
USB	Universal Serial Bus

1 INTRODUCTION

Robotics is the intelligent connection of perception to action.

— Michael Brady [1]

The problem of guiding a robot's actions using visual feedback from a camera arose at least four decades ago [2]. The modern term for this is *visual servoing*. Visual servoing requires, at minimum, a single camera, but, with the availability of RGB-D cameras, classical visual servoing techniques have been reinvestigated and new methods and applications have been developed.

The first industrial robots were envisioned to make manufacturing more economical than cam control systems, by being adaptable to new tasks [3]. As tasks grew in complexity, so did the fixtures that were required to fix objects to precise locations for the robot to interact with, negating the robot's general-purpose or flexibility [4].

A robot without sensory feedback of its environment requires its environment to be structured. The environment must be co-engineered to facilitate the specific robot [5] and the robot must know, a priori, the location of objects it interacts with. Visual feedback can reduce the required tolerance on the expected location of objects [6], allowing the robot to work in a more unstructured environment without explicit reprogramming of existing tasks.

There has been much interest in using the Microsoft Xbox 360 Kinect camera for visual servoing applications [7–14]. Of these applications, some do not explicitly state if they consider camera calibration and depth distortion modelling [7, 8, 10, 13, 14], and some do not consider the Kinect's limitations, such as the depth camera's sensitivity to sunlight [9] or its near range limitation [13, 14].

Since the Kinect is a relatively cheap device, it is expected to have its shortcomings. This research models the Kinect's RGB and depth cameras, conducts joint camera calibration using the Kinect Calibration Toolbox (KCT) by Herrera [15], characterises the devices limitations, and investigates some of the consequences of using the Kinect camera for visual servo control applications.

1.1 Research Questions

This research addresses the questions:

1. How are the Kinect RGB and depth cameras modelled?
2. What practical considerations does the Kinect impose on visual servo control applications?

1.2 Methodology

The research questions are addressed by following these steps:

- Model the Kinect RGB camera using the pinhole camera model.
- Model the Kinect depth camera using supporting literature.
- Calibrate the Kinect cameras using the KCT by Herrera [15].
- Characterise the limitations of the Kinect device.
- Given the calibrated models and limitations, discuss some of the consequences of using the Kinect for visual servo control applications.

1.3 Structure of the Research Report

Chapter 2 covers relevant literature on visual servoing, its limitations, and Kinect camera modelling and calibration.

Chapter 3 covers briefly the topics of: pose, image formation, image distortion, image sensors, and the intrinsic and extrinsic camera parameters. Readers familiar with these topics can skip this chapter.

Chapter 4 shows the Kinect characterisation, discusses and conducts the calibration procedure, and analyses the results.

Chapter 5 covers basic visual servoing techniques, and discusses some of the consequences of using the Kinect for visual servo control applications.

Chapter 6 addresses the research questions and concludes this research.

Chapter 7 provides recommendations for future research.

2 LITERATURE REVIEW

A literature survey on the basic techniques, advanced techniques, and problems of visual servoing is presented. Additionally, visual servoing in the context of the Kinect is presented. Kinect topics are then discussed separately, in particular, its sensor design, characterisation, calibration, depth distortion models, interference, and noise.

2.1 Visual Servoing

The earliest found *IEEE* visual servoing publication was released in 1985 (Figure 2.1), but *visual feedback* techniques date past four decades. Shirai and Inoue's 1973 publication [16] is generally regarded as the first to show end-effector positioning using visual feedback.

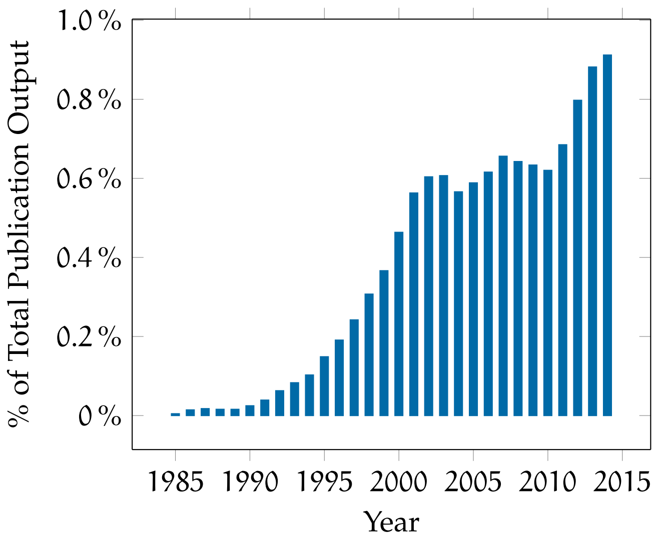


Figure 2.1: Ratio of *visual servoing* publications to total publication output, based on the IEEE Xplore database.

To the author's knowledge, Agin [17] first used the term *visual servoing* in 1977 at SRI International, for coarse positioning of an industrial manipulator mounted with a camera for a bolt insertion task.

Most of the basic visual servoing fundamentals were established by the early 1990s, which followed with the first visual servoing tutorial by Hutchinson et al. [18] in 1996.

Early advanced techniques were developed at the turn of the millennium, which followed with updated tutorials by Chaumette and Hutchinson [19, 20] in 2006 and 2007 respectively.

Visual servoing schemes can be classified in a number of ways. The main classifications are based on: use of dynamic, closed-loop visual feedback control; use of the robot's closed-loop joint control; use of a three-dimensional target object model; and camera configuration.

Early visual feedback systems, *static look-and-move*, cycled a closed-loop joint controlled robot through a sequence of three mutually exclusive steps to complete a task [21]: look, plan, and move. While the robot was busy moving, it could not look or plan the next incremental position to move. While these systems had elements of visual sensing, they lacked dynamic control, hence they are no longer used.

Later visual servo systems, *dynamic look-and-move*, executed every step in parallel [21]. These systems used a closed-loop visual feedback controller to output velocity commands to the joint controller, making the system more responsive, but dynamic performance and stability must be evaluated for acceptable transient response.

An *indirect* system is synonymous with a dynamic look-and-move system [22]. In contrast, a *direct* system uses the vision-based controller to directly stabilise a robot with no joint controller [21].

Weiss et al. [21] originally proposed the distinction between visual servo (direct) systems and dynamic look-and-move systems. The term, dynamic look-and-move, is depreciated, and now visual servo systems generally refer to indirect systems [4]. Indirect systems are preferred because a joint controller handles low sampling rates from the camera and robot dynamics and kinematic singularities are hidden from the vision-based controller [18]

In *model-based* visual servoing, the three-dimensional model of the target object is available [22]. The target object model and camera intrinsic parameters are required for pose (position and orientation) estimation algorithms. Otherwise, the method is *model-free*.

An *eye-in-hand*, or *end-point closed-loop*, system mounts the camera by the robot's end-effector. Camera motion is coupled to the robot's motion. An *eye-to-hand*, or *end-point open-loop*, system has the camera fixed in the workspace observing the target object and robot [4].[1]

For visual servoing task functions, the desired end-effector reference is specified using either predefined Cartesian references or taught image references [23].

1 These end-point definitions differ to Hutchinson et al. [18].

Predefined Cartesian references are naturally specified using pose, usually from a known model, but cause the steady state end-effector pose accuracy, or sensitivity, to suffer in the presence of camera calibration errors, robot calibration errors, and target object modelling errors [24].

Taught image references are obtained via an offline method, teach-by-showing [25], where the robot's end-effector is brought to the desired end-effector pose and a corresponding image is captured, or taught. Calibration and modelling errors are partially circumvented [24].

2.1.1 Basic Techniques

There are two basic approaches to visual servoing: Position-Based Visual Servoing (PBVS), or three-dimensional visual servoing; and Image-Based Visual Servoing (IBVS), or two-dimensional visual servoing.[2] These control schemes are identified by their feedback signal. The strengths and weaknesses of each method is discussed.

PBVS control schemes regulate end-effector pose in Cartesian space. Pose estimation is used to determine the relative pose error from the desired end-effector pose. This method bears resemblance to regulating a robot along a geodesic Cartesian trajectory.

There are few publications on PBVS. Westmore and Wilson [26] investigated the feasibility and performance of using the Extended Kalman Filter (EKF) for real-time PBVS. Their static object experiment was successful, but it was limited to two-dimensional position control. The three-dimensional control case and complete PBVS design methodology was later demonstrated by Wilson et al. [27].

Since PBVS regulates pose, and not the image, the resulting Cartesian trajectory does not guarantee that the image features of a target object stays within the camera's field of view (a visibility constraint), which causes pose estimation to fail. Such a problem can be avoided by using trajectory planning [4] or certain PBVS control laws [23].

IBVS control schemes regulate image features directly in image space. The position of a set of image features implicitly defines a pose. By extension, the current image and reference image implicitly define the current pose and reference pose, respectively. Hence, regulating the current image to look like the reference image should move the end-effector towards the desired pose.

Sanderson and Weiss [28] proposed the first direct IBVS system. Simulations of direct IBVS using adaptive control techniques were demonstrated by Weiss et al. [25], but were limited to three or less Degree of

2 Position-based and pose-based visual servoing are used interchangeably.

Freedom (DOF) manipulators due to the low sampling rate and time delay of their image processing system. An indirect 4-DOF IBVS system was later shown by Feddema and Mitchell [29], who overcame the vision sampling rate problems using an asynchronous feature-based trajectory generator. An indirect 6-DOF IBVS system was demonstrated by Rives et al. [30] and Chaumette et al. [31], who formalised visual servoing analysis using the task function approach.

In contrast to PBVS losing sight of the target object, IBVS control schemes have no control over their Cartesian trajectories. Given the position of an image point in the current image and its position in a reference image, the IBVS control law pushes this image point from its current position in a straight line towards its reference position. This trajectory, or the image feature trajectory, requires the robot's joints to do unnatural movements. If large rotations about the camera's optical axis are required, the robot may exceed its joint limits [23].

IBVS requires computation of the image Jacobian, or interaction matrix, which relates camera spatial velocity to image feature velocity. The image Jacobian's form depends on the choice of image features, with point features being the most common form. Rives et al. [30] solved the image Jacobian for geometric primitives, such as points, lines, and circles. Other image features include ellipses [32] and image moments [33, 34].[3]

Most image Jacobians are a function of feature depth. Depth can be measured or estimated. Online estimation solutions to this problem include: partial three-dimensional reconstruction [29], constant depth [32], adaptive control [35], and depth from image and robot motion [36]. A depth-independent image Jacobian approach is presented here [37], but only for eye-to-hand (fixed camera) systems.

2.1.2 Stability, Robustness, and Sensitivity

IBVS with taught image references is known to be tolerant towards image quantisation errors, measurement noise [38] and camera calibration errors [39] - that is, systematic errors. Hence the errors from the reference and measurement cancel out. Despite this, a case of IBVS failure due to poor camera calibration and noise has been reported in [40], and, even with accurate camera calibration, care must be taken with regards to depth estimation errors of the target object [41, 42].

In contrast, PBVS control is generally said to be unstable in the presence of sensor errors. This is an often cited PBVS disadvantage. Janabi-sharifi et al. [23] argued that PBVS instability is due to the use of predefined Cartesian references. They said that PBVS and IBVS performance

3 Rives et al. [30] used the term *feature sensitivity matrix* for the interaction matrix.

is expected to be the same when both schemes use the same reference type (predefined or taught references), and performance should be evaluated with a common framework.

Given a common framework, the performance issues lie with the camera, target object, and robot modelling errors. Under these errors [23]: image feature trajectories will warp when using taught image references; and the steady state error will increase when using predefined Cartesian references.[4]

Local asymptotic stability of IBVS was shown by Espiau et al. [32], making IBVS a local-only control method; if the initial camera displacement (relative to the target pose) is too large, the current pose may fail to converge to the desired pose.

At least three point features are required for IBVS control schemes, such that the image Jacobian matrix has a rank of six and the image Jacobian inverse exists. Then, the camera motion equations can be solved, but this solution is not unique. There are four solutions [43], or global minima, where both the camera spatial velocity and control error, or task function error, are zero. A unique solution is available if at least four point features are used, for which the image Jacobian pseudo-inverse has a least squares error solution [38].

When using four or more image point features, the dimension of the image Jacobian pseudo-inverse kernel is non-zero. This indicates that *sometimes* local minima exist [43], that is, camera spatial velocity is zero and the task function error is non-zero. If a IBVS control law with a constant image Jacobian pseudo-inverse is used, the unique global minimum can be reached. Nonetheless, this control law sometimes causes image point features to leave the camera's field of view, causing the image Jacobian to become rank deficient.

Tasks requiring a rotation around the (optical) z-axis follow unsatisfactory Cartesian trajectories, as IBVS enforces shortest image feature trajectory. Instead of rotating, a backwards camera translation happens first before moving towards the target object. This effect is called camera retreat, which is a result of motion control coupling [43].

A required 180° rotation about the z-axis is an extreme case which results in task singularity. The initial and desired image point features are reflections about the camera's principal point. Camera retreat occurs towards depth at infinity, where all image point features lie at the centre of the image, or principal point. Either the robot becomes joint-limited first, or the image Jacobian becomes rank deficient.

Also on 180° rotation problems, some IBVS control laws cause a forwards camera translation (camera advance) towards depth at zero,

4 Trajectory warping increases the task convergence time, since regulating along a non-straight line takes longer than along a straight line.

where hopefully all image point features are out of the camera's field of view before colliding with the target object.

These characteristic problems of IBVS, local minima and motion control coupling, do not exist for PBVS [23].[5] PBVS problems usually involve pose estimation algorithms and visibility constraints, which incur computational costs. Characteristic problems of IBVS are addressed by advanced visual servoing techniques.

2.1.3 Advanced Techniques

Several categories of (early) advanced visual servo schemes exist that mix PBVS and IBVS behaviour together to avoid their respective disadvantages. Major categories are covered, but only a handful of commonly encountered schemes are discussed.

Hybrid techniques, such as the Koichiro Deguchi (or KD) method [45] and 2-1/2-D visual servo [46], have been proposed. These techniques are homography-based, which find the plane-to-plane relationship, or homography matrix, between the current and reference images. The homography matrix is then decomposed into separate rotation and translation quantities for decoupled motion. Since motion control is decoupled, IBVS local minima can be avoided. Additionally, partial camera displacement estimation using the homography matrix does not require the three-dimensional object model.

With homography-based approaches, however, at least four and eight matching image feature points are required for coplanar objects and non-coplanar objects, respectively. The homography matrix is also susceptible to noise, which makes the performance of hybrid schemes degrade more than plain IBVS approaches [47].

Specifically, the KD method [45] is designed around the visibility constraint and translation motion is constrained to take the shortest path, but it does not address task singularity and stability issues. In contrast, 2-1/2-D visual servo [46] designs for robustness [48] and a task singularity free workspace, but adaptive control of gain and system parameters is required to enforce the visibility constraint, which inflicts a slow time-to-convergence due to gain adaptation.

For partitioned visual servo schemes, the XY/Z partitioned scheme by Corke and Hutchinson [44] acknowledged that motion control coupling is a z-axis translation phenomenon and task singularities are a z-axis rotation phenomenon. These two motion controls are isolated, which involves computing two computationally inexpensive image

5 The camera retreat problem is also called the Chaumette Conundrum [44].

features that are mutually motion decoupled. Still, this method is susceptible to image feature points leaving the camera's field of view.

A simple PBVS and IBVS switching controller was proposed by Gans and Hutchinson [49]. Each controller had an accompanying Lyapunov function with a chosen threshold. While the PBVS controller was active, the switch to IBVS control happened when the IBVS Lyapunov function exceeded its threshold and vice versa. PBVS flaws are mitigated, but characteristic problems of IBVS still exist.

More recently, authors have taken a different approach, using direct visual servoing methods. In contrast to geometric feature-based visual servoing methods, direct methods do not require any image feature extraction, matching, and tracking, and instead perform alignment of whole images.[6] Since whole images are used, there is high redundancy, making final positioning errors much smaller than feature-based methods and tasks more robust towards partial occlusions. This is very useful for tasks in complex scenes.

One example of direct methods is photometric visual servoing by Collewet and Marchand [51], where whole image luminance is used as a visual feature. Only the corresponding reference image is required to perform a task, but tasks may diverge due to illumination variations. Variation sensitivity depends on the illumination model. The authors proposed that complex illumination models may provide better insensitivity.

Of particular interest is the direct dense depth map-based visual servoing by Teulière and Marchand [13], where the control law uses whole depth maps from a Kinect sensor. Depth maps are insensitive to illumination variations, offering improved stability over photometric visual servoing. Nonetheless, this method's limitations lie with the Kinect, which has limited range and inability to sense outdoors due to infrared light from sunlight, as it is an active infrared sensor.

An extension to Teulière and Marchand [13] was presented by Hojaij et al. [14]. They designed a two phase controller which first performed coarse positioning using a IBVS controller with Speeded Up Robust Features (SURF), then switched to the direct controller for fine positioning. The fine positioning controller is activated when crossing a hardcoded depth threshold with respect to the target object.

Direct methods are essentially IBVS methods with redundant features, hence they suffer similar limitations. Their Cartesian trajectories are non-optimal. In addition, control laws of direct methods are more nonlinear, so this effect is more pronounced. Experiments by [52] using a hybrid method alleviate this problem.

6 Direct visual servoing is a subclass of nonmetric visual servoing [50].

With IBVS, tasks will fail with large initial displacements. Similarly, direct methods fail when there is insufficient overlap between current and reference images or wider occlusions, that is, they have limited domain of convergence. Improvements to domain of convergence have been proposed using hybrid methods [53] and different optimisation techniques [50].

2.2 Microsoft Kinect

The Kinect is Microsoft's vision and voice sensor, used by their Xbox gaming devices. It has a colour (RGB) camera, a depth camera, a multi-array microphone, and a tilt motor at its base which allows it to pitch [54]. Its depth camera is an IR camera and projector pair. Measurements are accumulated internally and either raw or processed data is transmitted externally via USB.

Microsoft's Kinect SDK can be used to infer information such as human body location and track body movement. For game designers, this kind of information is useful for designing transparent user interfaces with intuitive controls.[7]

2.2.1 Kinect-based Research

Figure 2.2: The Kinect for Xbox 360.

The Xbox 360 Kinect was advertised as a gaming interface (Figure 2.2), and, within three months of its November 2010 release, it sold over 10 million units [54]. Its low cost and off-the-shelf availability promoted

7 Otherwise known as a Natural User Interface (NUI).

its popularity with researchers. Microsoft termed the development of many creative Kinect applications the "*Kinect Effect*" [54].

Many open source software organisations developed computer vision library and robotics middleware support, allowing for fast prototyping of Kinect-based experiments.[8]

Kinect vision-based research and applications have been surveyed extensively by Han et al. [55] and Zhang [54]. In addition, some practical robotics examples include:

- *Quadrotor + Kinect* by Bouffard et al. [56], a project which received wide media coverage, and has a YouTube video with over one million views.
- *TurtleBot*, a personal robot kit from the Open Source Robotics Foundation, Inc. [57] (formerly Willow Garage) for researchers and hobbyists.
- Kinect-based visual servoing: PBVS with redundant joints [7], object handling [8–10], contour following [11], robust and fast object tracking [12], and direct and hybrid methods [13, 14].
- And more recently, *Future Robotic Interfaces* at NASA's Jet Propulsion Laboratory (JPL) [58], using the improved Kinect 2.

2.2.2 Software Drivers

Three drivers are available for communicating with the Kinect [55, 59]. Two common choices used for research are the `libfreenect` driver by OpenKinect and OpenNI driver by PrimeSense.[9] The third driver, bundled in the Microsoft Kinect Software Developer Kit (SDK), is proprietary and only runs on Windows 7 (or above) operating systems.

The OpenNI driver is known to enforce factory calibrated settings and give per-pixel depth values in millimetres [60]. Only the `libfreenect` driver is able to provide per-pixel disparity values in Kinect disparity units (kdu), which is a widely used metric for comparing depth accuracy measurements [60–63].

8 Organisations such as: Open Source Computer Vision (OpenCV), Point Cloud Library (PCL), and Robot Operating System (ROS)

9 The official PrimeSense sensor module GitHub repository does not contain any compiled binaries `https://github.com/PrimeSense/Sensor`. Cross-platform binaries can be found in the SensorKinect repository, maintained by a user known as avin2 `https://github.com/avin2/SensorKinect`. PrimeSense was acquired by Apple Inc. and *official* support for OpenNI has since dropped.

2.2.3 Depth Sensor Design

Since the Kinect's hardware is proprietary, its technology is undisclosed. Early efforts to understand the Kinect were published as informal online sources.[10] According to Zhang [54], Microsoft licensed their depth-sensing technology from PrimeSense. Since there is no direct link, Kinect characterisation authors [61, 62, 65–67] have speculated that the technology is explained in the patent by Freedman et al. [68], which was assigned to PrimeSense.

Freedman et al. [68] explains that PrimeSense's technology computes the three-dimensional coordinates of points using triangulation. Computing such a point requires the point to be visible in at least two images, each from a different view [4].[11] To generate the first image, a projector projects a fixed speckle pattern onto a region, which is captured by a camera. A stored reference pattern with known distance from the sensor is used as the second image. This reference pattern is captured during factory calibration.

The speckle and reference patterns are not stated to be the same, but it is highly likely that they are similar so that the correspondence problem (or image pair pixel matching) can be solved with less computation. A more in-depth discussion about this technique is given by Chow and Lichti [67]. Martinez and Stiefelhagen [69] suggest a possible architecture to emulate the Kinect's internal processing.

The IR projector's light source is not limited to IR, but the near-IR optical band is preferred for its availability of low-cost components [68]. Zhang [54] notes that the Kinect does indeed use an IR projector and camera pair for depth-sensing.[12]

Each Kinect depth map is a greyscale image. If a pixel is black, it is an invalid depth measurement [54], or *zero depth* [66], because the projected pattern cannot be seen in that region by the IR camera. This has known to occur when points are too far or too close the sensor, regions are shadowed due to occlusion, surfaces are not IR-friendly, or projected pattern regions are exposed to sunlight.

2.2.4 Kinect Disparity Models

The relationship between depth and disparity (the pixel difference of the same image point between two images) is linear, by way of sim-

10 Search ChipWorks or iFixit for *Kinect Teardown*. URLs are not provided. ChipWorks have changed theirs before, which made reference tracing difficult in [62, 64].
11 Using two cameras, or a single moving camera.
12 See `http://openkinect.org/wiki/Hardware_info` for more information about the IR projector and camera pair.

ilarity of triangles [62, 65]. The Kinect, however, returns normalised 11-bit integer disparity values (kdu), and not raw image disparity.

Denormalisation of raw image disparity is required, using denormalisation factors. Such models are suggested by Smisek [64, pg. 12], Khoshelham [70], Macknojia et al. [65], and Herrera et al. [60, 71].

Since the normalised disparity, or Kinect disparity, is quantised, the measurable depth range is also quantised. As a result, depth measurements are subject to depth uncertainty. This uncertainty is bounded by a depth value's adjacent quantised values.

2.2.5 Depth Resolution and Accuracy

The minimum measurable change in depth, or quantisation error, at different distances from the sensor is called depth resolution. This is found to change quadratically with distance [61, 62, 65, 72].

Smisek et al. [61] find depth resolution experimentally, by moving the Kinect away from a planar target and recording the quantisation error between two consecutive values around a depth map's centre.

Macknojia et al. [65] construct a depth resolution model and verify it empirically. Their quantisation error is constructed using two consecutive depth values expected by their Kinect disparity model.

2.2.6 Calibration

Microsoft's factory calibration of the Kinect is based on the technique by Zhang [54, 73]. Khoshelham and Elberink [62] do early work on improving depth accuracy, but perform RBG and IR camera calibration separately using proprietary software, Photomodeler. Smisek et al. [61], Smisek [64] calibrate the RGB and IR camera separately using the Camera Calibration Toolbox for Matlab by Bouguet [74]. In [64], the IR projector is blocked and the scene is illuminated using a halogen light for the IR camera to capture calibration images.

Herrera et al. [60] argue that Khoshelham and Elberink's and Smisek et al.'s approaches are incomplete, since minimising residuals (for a camera parameter) of the RGB and IR camera separately does not guarantee the minimum residuals when using the cameras jointly.

For this reason, joint calibration is investigated by Zhang and Zhang [75] and Herrera et al. [60]. Both show improvements to sensor accuracy over individual calibration. Herrera's calibration procedure is available publicly as the KCT [15].

Raposo et al. [63] make incremental performance improvements on Herrera et al.'s method by using different initial parameters and a smaller data set to shorten the calibration error minimisation runtime. This method, however, requires the dimensions of the calibration target. Chow and Lichti [67] further improve on depth accuracy by modelling the depth systematic errors using bundle adjustment.

Using two Kinect sensors, Herrera et al. [60] discovered the variability of camera parameters. Colour and depth reprojection errors increase considerably when using foreign calibration results. The exact parameters that are variable, however, are unmentioned. Hence, this is an opportunity for further investigation.

2.2.7 Depth Distortion

The Kinect exhibits depth (or disparity) distortion that is more pronounced on close range measurements. Smisek et al. [61] show that this error is highly correlated on depth maps captured between $0.7\,\mathrm{m}$ to $1.3\,\mathrm{m}$. It is corrected using a depth correction image that is superimposed on the captured depth map, reducing the standard deviation of depth reprojection errors. The depth correction image is created using the mean error of the captured close range images.

Alternatively, Herrera et al. [60] argue that the depth distortion is an exponentially decaying fixed pattern and correcting the Kinect disparity units directly yields more accurate results than Smisek et al.'s model. Both of these models are non-parametric and only account for IR camera distortions. Yamazoe et al. [76] propose a parametric model for depth distortion that models both the IR camera and projector. Their model parameters are estimated by minimising the plane-fitting errors to a plane with known dimensions.

2.2.8 Depth Sensor Interference and Noise

A survey of Kinect-equipped mobile robots [77] notes the poor performance of the Kinect for outdoor work. The projected pattern of the Kinect is inherently dim, so exposure to sunlight quickly saturates depth measurements. If outdoor work is absolutely necessary, pairing the Kinect with a stereo camera is proposed [78].

Fiedler and Heinrich [79] find that depth measurements are sensitive to device temperature changes, which increases focal lengths of the RGB and IR camera by as much as $2\,\mathrm{mm}/°\mathrm{C}$.

Experiments by Haggag et al. [72] show that depth measurements fluctuate due to thermal illumination from heated surfaces, making the Kinect unsuitable for use in high temperature environments.

Depth measurements also fluctuate on non-heated stationary planar surfaces [72], but less wildly than from thermal illumination. Mallick et al. [66] call this temporal noise and suggest a model, but acknowledge that it behaves like depth distortion. Hence, a tuned depth distortion model should sufficiently compensate for this noise.

Mallick et al. [66] define other noise classes. Specifically, shadow noise is the depth hole formed behind occlusions due to IR camera and projector displacement and lateral noise is the depth measurement fluctuation around object edges due to depth discontinuities. These noise types increase with the distance of background objects.

Multiple Kinects will interfere with each other when their projected patterns overlap. Workarounds include setting patterns at different angles or cycled shuttering using software, hardware, or mechanically. This topic discussed in detail here [66, 72, 80].

2.3 Summary

Basic and most advanced visual servoing techniques require some depth estimation or measurement. While depth errors are not problematic, using a depth camera can simplify control law computation. A candidate depth sensor is the Kinect, as it is low cost and has a large body of characterisation research. Based on the literature survey and the author's knowledge, there is no literature on Kinect characterisation specifically for visual servoing applications.

3 GEOMETRIC CAMERA MODELLING

A camera records visual information from the world as two-dimensional images. In practice, these images have observable errors due to physical phenomena or manufacturing defects. A camera model acknowledges these problems, such that they can be parametrised and corrected. The intrinsic, extrinsic, and lens distortion parameters for a geometric camera are derived in the following sections.[1]

3.1 Planar Transformations

Planar transformations map coordinates from one coordinate frame to another. They take the form

$$\bar{\mathbf{p}}' = \mathbf{M}\bar{\mathbf{p}} \tag{3.1}$$

where $\bar{\mathbf{p}}$ is an augmented coordinate, $\mathbf{M}$ is a homogeneous transformation matrix, and $\bar{\mathbf{p}}'$ is the transformation of $\bar{\mathbf{p}}$.

3.1.1 Translation

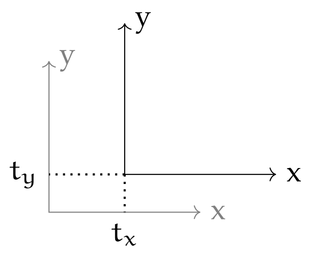


Figure 3.1: Two-dimensional translation.

Translation in one dimension is written $x' = x + t_x$ (Figure 3.1). A positive translation shifts a coordinate frame's position forward. The two-dimensional translation matrix is

$$\mathbf{X}(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

1 The notations used throughout follow Corke [4], Szeliski [81], and Bouguet [74].

3.1.2 Rotation

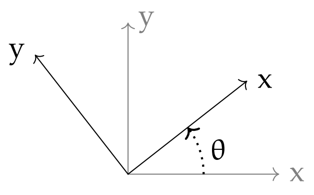


Figure 3.2: Two-dimensional rotation.

Counter-clockwise rotation about a coordinate system's origin is illustrated in Figure 3.2, where θ is its rotation angle, or orientation. The two-dimensional rotation matrix [4] is

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.3}$$

The three-dimensional rotation uses a sequence of three two-dimensional rotations about different axes.[2] Rotation about the z-axis is given in Equation 3.3. Rotations about the x and y axes [4] are

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{3.4}$$

and

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}. \tag{3.5}$$

Commonly used sequences are ZYZ and XYZ (roll-pitch-yaw), that is, $\mathbf{R}_{zyz} = \mathbf{R}_z(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi)$ and $\mathbf{R}_{xyz} = \mathbf{R}_x(\theta_r)\mathbf{R}_y(\theta_p)\mathbf{R}_z(\theta_y)$, respectively. A rotation is undone, or reversed, by using its transposed matrix, given that $\mathbf{R}^T = \mathbf{R}^{-1}$ is a property of orthonormal matrices.

3.1.3 Relative Pose

An object's position and orientation is called its pose. Pose is meaningless without stating a reference frame. Given two coordinate frames {A} and {B}, relative pose of frame {B} with respect to frame {A} is denoted ${}^A_B\xi$ (Figure 3.3).

2 These sequences, or Euler angles, are discussed by Corke [4].

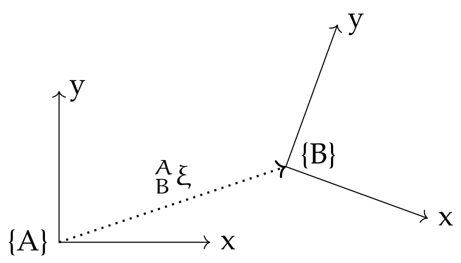


Figure 3.3: Relative pose between two coordinate frames.

Relative pose is interpreted in either one of two ways:

1. {B}'s location relative to {A}; each frame is a different object.
2. {A}'s motion to get to {B}; both frames are the same object.

One way to represent pose is the homogeneous transformation matrix, $\xi(\mathbf{t}, \theta) \sim \mathbf{T}(\mathbf{t}, \theta)$.[3] The three-dimensional pose matrix [4] chains together the translation and rotation matrices

$$\mathbf{T} = \mathbf{X}(\mathbf{t})\mathbf{R}(\theta) = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}_{4\times 4}, \tag{3.6}$$

which has an inverse

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.7}$$

Pose has an important property called compounding: given three coordinate frames {A}, {B}, and {C}, if poses $^A_B\xi$ and $^B_C\xi$ are known, then the two known poses can be compounded to find $^A_C\xi$.

As an example, consider a small room with a table at its centre. Atop the table sits two objects (Figure 3.4). Usually, the table's location is described with respect to the room. Similarly, the objects' location are described with respect to the table.

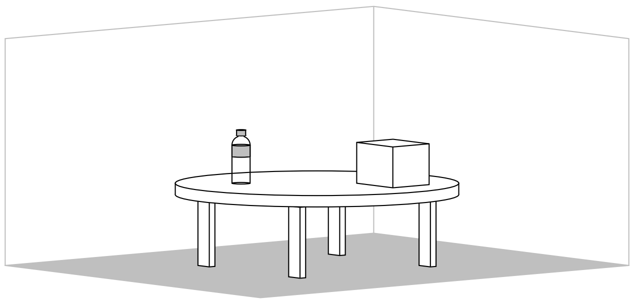


Figure 3.4: Objects in three-dimensional space.

3 Different types of pose representations are discussed by Corke [4].

With each object stripped to its coordinate frame (Figure 3.5), it is possible to follow, or compound, the pose between the room and table and the pose between the table and its objects, to find the pose between the room and the objects.

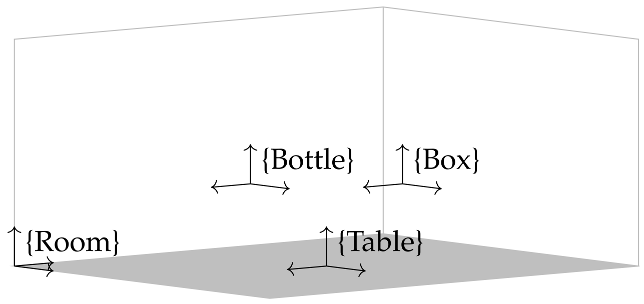


Figure 3.5: Coordinate frames in three-dimensional space.

Using $\mathbf{T} \sim \xi$ for pose, composition is matrix multiplication

$$\begin{aligned} {}^{A}_{C}\mathbf{T} &= {}^{A}_{B}\mathbf{T}\,{}^{B}_{C}\mathbf{T} \\ &= \begin{bmatrix} {}^{A}_{B}\mathbf{R} & {}^{A}_{B}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^{B}_{C}\mathbf{R} & {}^{B}_{C}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} {}^{A}_{B}\mathbf{R}\,{}^{B}_{C}\mathbf{R} & {}^{A}_{B}\mathbf{R}\,{}^{B}_{C}\mathbf{t} + {}^{A}_{B}\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \end{aligned} \tag{3.8}$$

Essentially, $\mathbf{T}$ has the planar transformation form (Equation 3.1), ${}^{A}\bar{\mathbf{p}} = {}^{A}_{C}\mathbf{T}\;{}^{C}\bar{\mathbf{p}}$, and has the inverse ${}^{C}\bar{\mathbf{p}} = {}^{A}_{C}\mathbf{T}^{-1}\;{}^{A}\bar{\mathbf{p}} = {}^{C}_{A}\mathbf{T}\;{}^{A}\bar{\mathbf{p}}$.

3.2 Image Formation

Photographic images are formed from structured light rays.[4] This structure holds geometric information, while the light rays carry spectral information, which depend on ambient light and the reflection (or absorption) of different surfaces (Figure 3.6).

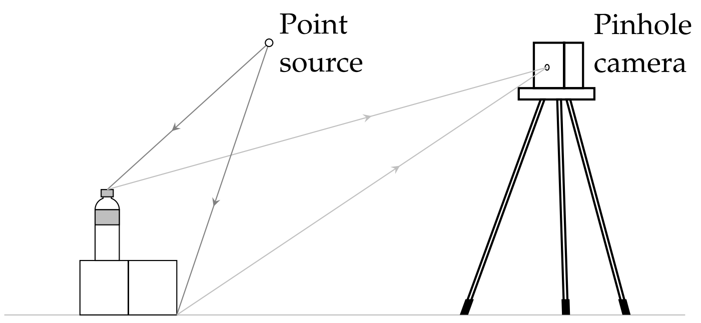


Figure 3.6: Ray tracing from a point source to a pinhole camera.

4 Light can be interpreted as either rays or waves [82].

The most basic camera is the pinhole camera. A pinhole has limited control over the light rays it captures. This allows it to capture either, sharp images with low brightness, or blurred images with high brightness. Both cases are not very useful. Nonetheless, pinholes demonstrate the fundamentals of image formation.

An ideal pinhole allows one world point to project to image point. The formed images have low brightness and they are inverted, due to how pinholes allow light rays to go through (Figure 3.7).

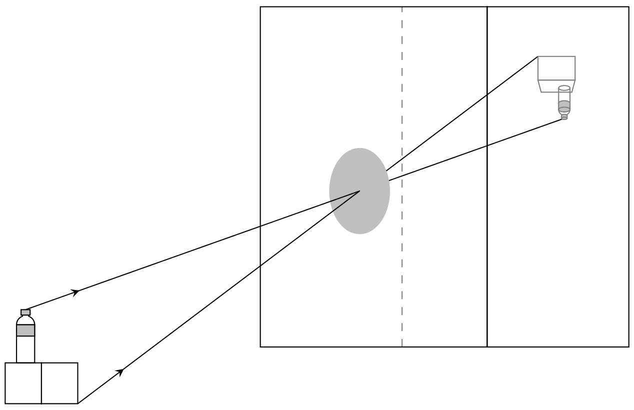


Figure 3.7: Ideal pinhole image formation.

Naturally, image brightness increases by capturing more light, that is, by increasing a pinhole's radius. This causes, however, one world point to project to multiple image points, because more light rays are allowed to pass through the pinhole. Multiple images form that shift and overlap, causing the result to blur (Figure 3.8).

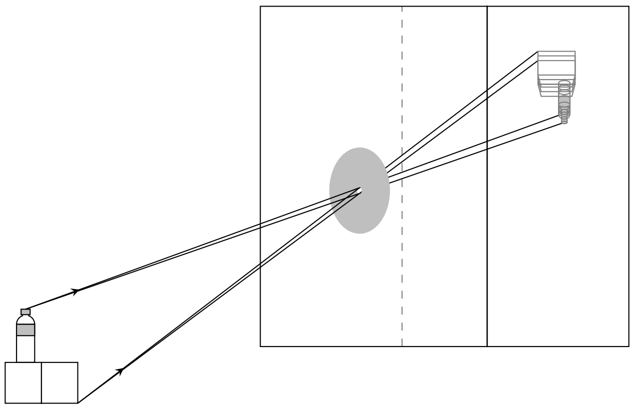


Figure 3.8: Blurred pinhole image formation.

Cameras can only use light rays within their line of sight. This increases along with a pinhole's radius, but it is self-defeating without control over light rays. To control these, pinholes must use a lens.

3.2.1 Thin Lens Optics and Focus

A lens converges parallel light rays, that are perpendicular to it, to its (back) focal point f. This point is at a length away from the lens called the focal length f. The focal length of a lens measures how strongly it bends light rays (Figure 3.9).

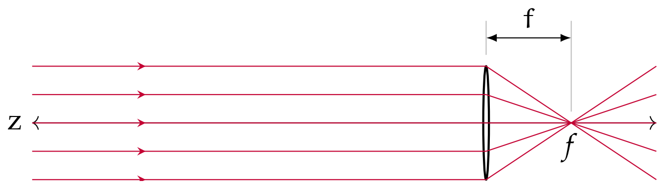


Figure 3.9: A lens converges collimated light to its focal point.

By convention, the z-axis is central and normal to the lens, pointing away from the camera and commonly called the optical axis. The optical centre, or $z = 0$, is the lens' equivalent pinhole (Figure 3.10).

The plane cutting the focal point, perpendicular to the optical axis, is called the (back) focal plane. Parallel to it is the (back) image plane. The image plane intersects incoming light rays, which captures a 2-dimensional image on it (Figure 3.10).

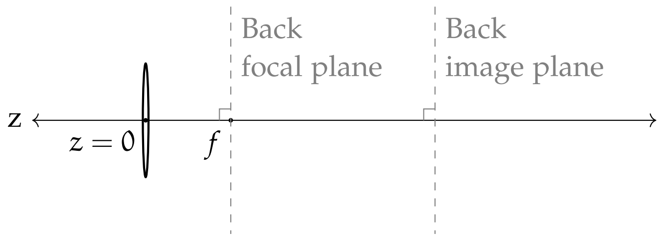


Figure 3.10: Optical centre, focal plane, and image plane.

Light ray optics can be shown basically using the thin lens. The thin lens equation [4] is

$$\frac{1}{\mathrm{f}} = \frac{1}{z_w} + \frac{1}{z_i}, \tag{3.9}$$

where:

f	focal length [mm]
z_w	distance between world point and lens [mm]
z_i	distance between lens and image point [mm]

that is, given a thin lens with focal length f, a world point at z_w projects to its sharpest image point at z_i (Figure 3.11, adapted [81]).[5]

5 The thin lens equation is parallel addition, essentially.

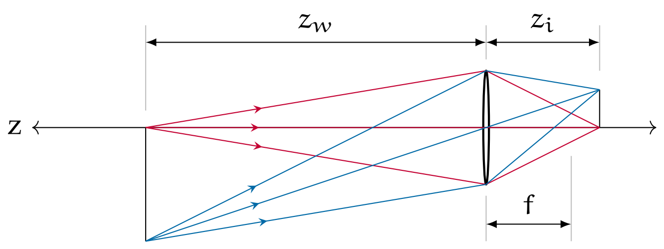


Figure 3.11: Thin lens optics.

Here, images captured on the image plane, placed at z_i, are in focus, because all converged image points lie on the image plane.

Using the thin lens, shifting the optical centre along the optical axis scales the image's size. This action is called zoom. Additionally, there are two other actions, pan and tilt, which are rotations about a lens' x and y axes, respectively (Figure 3.12).

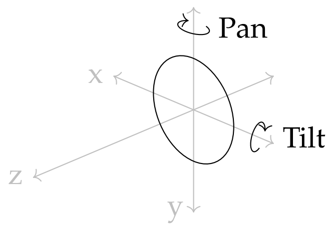


Figure 3.12: Pan and tilt.

In most situations, a camera views objects from an angle, and projected images are no longer parallel to the lens (Figure 3.13).

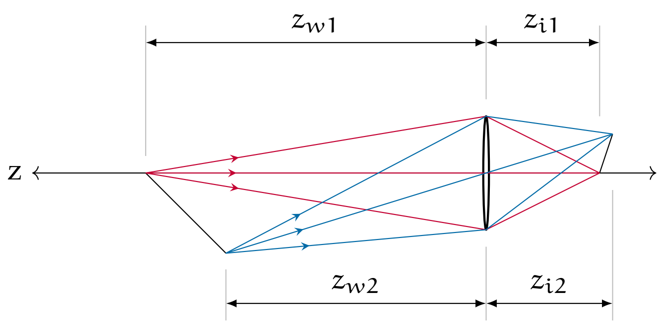


Figure 3.13: Viewing objects from an angle.

Here, the image plane can be placed at z_{i1}, z_{i2}, or in-between. Converged image points that touch the image plane are in focus, while surrounding image points are slightly out of focus; an image is only in focus within a region, and image focus becomes ambiguous.

The length between the nearest and furthest out of focus image points, which are acceptably sharp, is called the depth of field. This is a physical constraint, but it is less noticeable for objects further away.[6]

Equation 3.9 can be arranged as a translated hyperbola

$$z_i = f\frac{z_w}{z_w - f}. \tag{3.10}$$

It is obvious that $\infty > z_w > f \Rightarrow f > z_i > 2f$, or rather z_i lies between f and $2f$, if z_w lies between ∞ and $2f$.[7]

If z_w is sufficiently greater than f, then z_i will be sufficiently close to f (Figure 3.14). Acceptable sufficiency depends on the application. As an example, if $f = 18\,\text{mm}$ and $z_w = 1.8\,\text{m}$, then $z_i = 18.18\,\text{mm}$; that is, $z_w = 100f \Rightarrow z_i = 1.01f$.

This behaviour is reminiscent of Figure 3.9, which illustrates how a lens' focal length is measured. Indeed, light rays from world points that are far enough will appear to be collimated.

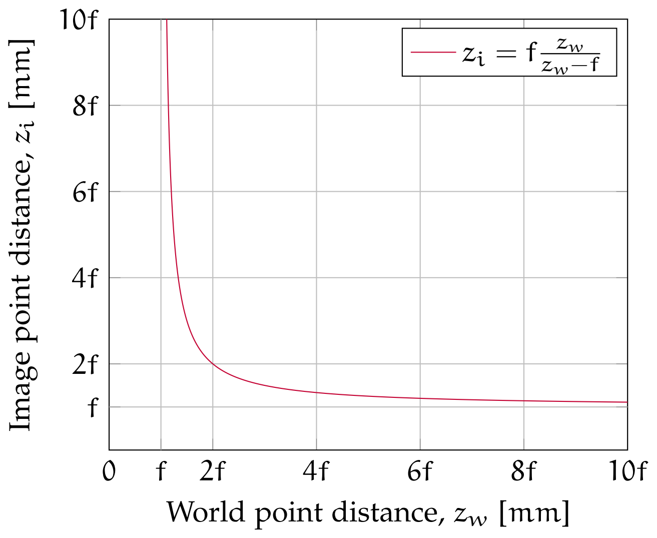


Figure 3.14: Thin lens behaviour using focal length multiples.

If $z_w \gg f \Rightarrow z_i \approx f$, then the whole image is nominally in focus on the focal plane; image focus is well defined. Also, the image plane is easily located if it coincides with the focal plane, such that it is at a distance, f, away from the lens.

Now, a lens focuses multiple light rays from one world point to one image point. When multiple images form, they overlap in the same position, increasing the image brightness, while keeping image sharpness (Figure 3.15).

6 A lens' aperture size and camera's exposure time can be adjusted to increase depth of field, but the added latency may be unacceptable for real-time applications.
7 z_w and z_i are interchangeable.

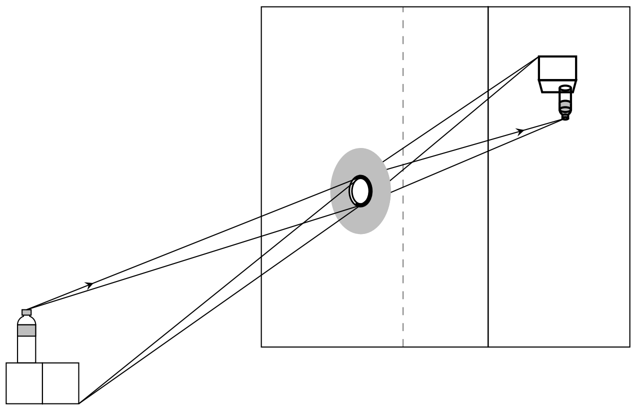


Figure 3.15: Focused image formation using a lens.

In practice, lenses are cascaded for finer adjustment over f. Hence, f usually represents a lens system's effective focal length.

3.2.2 Perspective Transform

The relationship between a three-dimensional point ${}^c\mathbf{P} = (X, Y, Z)$ and its image projection ${}^c\mathbf{p} = (x, y)$ is modelled by the perspective transform (Figure 3.16).

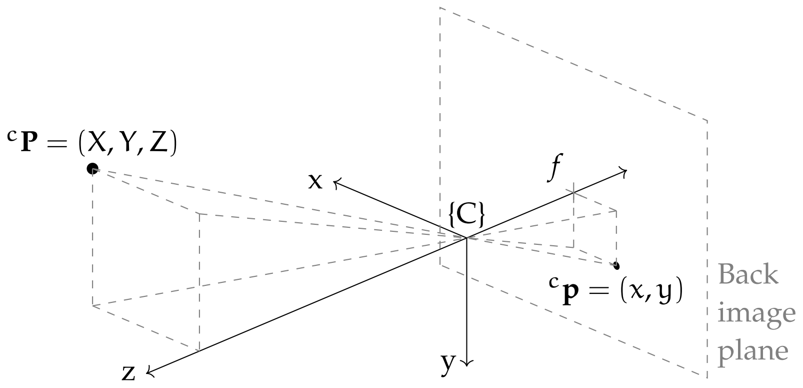


Figure 3.16: three-dimensional point projection through an equivalent pinhole.

The lens' frame is attached with the camera's frame, {C}. {C}'s origin coincides with the lens' optical centre, or equivalent pinhole. It is obvious that $\mathbf{P}$ and $\mathbf{p}$ are related via similarity of triangles,

$$\frac{x}{f} = \frac{X}{Z}, \tag{3.11}$$

$$\frac{y}{f} = \frac{Y}{Z}. \tag{3.12}$$

It is more *convenient* to use the central projection model, which instead uses the front image plane. This plane is the reflection of the back image plane about the optical centre (Figure 3.17).

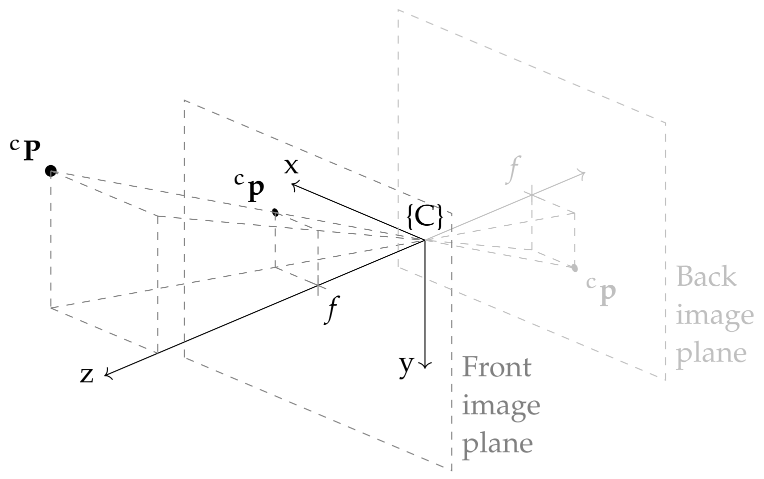


Figure 3.17: Central projection model.

A non-inverted image is formed on the front image plane. This plane, from the optical centre's point of view, is consistent with an observers' point of view, such that ${}^{c}\mathbf{P}$ and ${}^{c}\mathbf{p}$ move in the same direction.

3.2.3 Normalised Image Plane

A common intermediate step between ${}^{c}\mathbf{P}$ and ${}^{c}\mathbf{p}$ is to use normalised image coordinates [60, 74, 83], that is, image coordinates projected onto the plane at $z = 1$ (Figure 3.18).

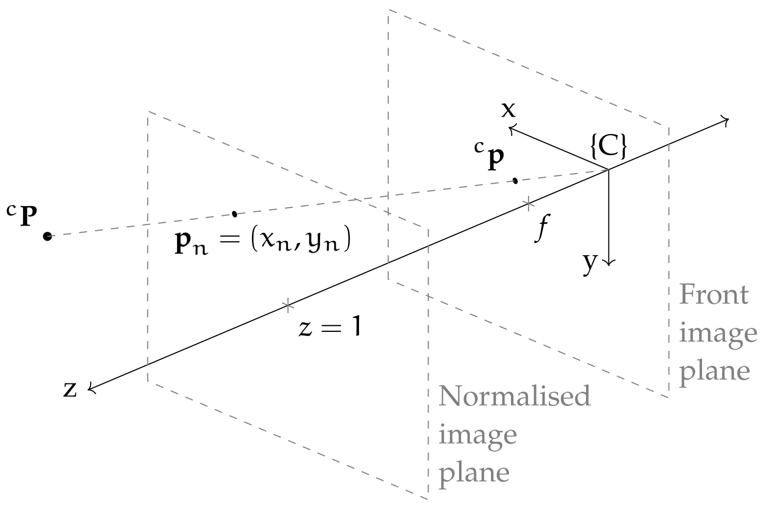


Figure 3.18: Normalised image plane and coordinates.

The normalised three-dimensional point of ${}^{c}\mathbf{P}$ is

$$\bar{\mathbf{p}}_n = \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \\ Z/Z \end{bmatrix} = {}^{C}\mathbf{P}_n. \tag{3.13}$$

It follows that $\mathbf{p}_n$ is scaled by f to obtain ${}^c\mathbf{p}$

$$ {}^c\bar{\mathbf{p}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{p}}_n. \tag{3.14}$$

The normalised image coordinate $\mathbf{p}_n$ is an unscaled intermediary for the theoretical image projection ${}^c\mathbf{p}$ which is *unobservable* in the presence of image distortions.

3.2.4 Geometric Lens Distortion

All lenses exhibit some sort of distortion that obfuscates the information on a projected image. The most prominent distortions are radial and tangential distortions (Figure 3.19, adapted [84]).

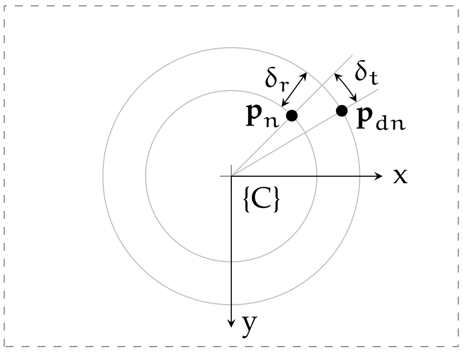


Figure 3.19: Radial and tangential lens distortions on the normalised plane.

Geometric distortions are systematic errors, which are easily corrected in software once parametrised. Let $r_n^2 := \|\mathbf{p}_n\|^2 = x_n^2 + y_n^2$, then the distorted normalised image coordinate is

$$\begin{aligned}
\mathbf{p}_{dn} = \begin{bmatrix} x_{dn} \\ y_{dn} \end{bmatrix} &= \mathbf{p}_n + \delta_r(x_n, y_n) + \delta_t(x_n, y_n) && (3.15)\\
&= \mathbf{p}_n + \underbrace{\left(k_1 r_n^2 + k_2 r_n^4 + k_3 r_n^6\right) \mathbf{p}_n}_{\delta_r(x_n, y_n)} \\
&\quad + \underbrace{\begin{bmatrix} 2p_1 x_n y_n + p_2(r_n^2 + 2x_n^2) \\ p_1(r_n^2 + 2y_n^2) + 2p_2 x_n y_n \end{bmatrix}}_{\delta_t(x_n, y_n)} \\
&= \left(1 + k_1 r_n^2 + k_2 r_n^4 + k_3 r_n^6\right) \mathbf{p}_n \\
&\quad + \begin{bmatrix} 2p_1 x_n y_n + p_2(r_n^2 + 2x_n^2) \\ p_1(r_n^2 + 2y_n^2) + 2p_2 x_n y_n \end{bmatrix} \\
&= \Delta_r(x_n, y_n)\mathbf{p}_n + \delta_t(x_n, y_n). && (3.16)
\end{aligned}$$

This is the Brown-Conrady distortion model [85],[8] parametrised by the low-order (normalised) radial and tangential distortion coefficients $(k_1, k_2, k_3, p_1, p_2)$. There are higher-order distortion coefficients, but low-order coefficients are enough for most applications [86, 87].

Once the distortion coefficients are found, the inverse distortion can be computed to correct the distorted image [86, 87], and the distorted normalised coordinate $\bar{\mathbf{p}}_{dn}$ replaces the normalised image coordinate $\bar{\mathbf{p}}_n$ in Equation 3.14, such that

$$^c\bar{\mathbf{p}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{p}}_{dn}. \tag{3.17}$$

3.3 Pixels, Image Sensors and Digital Images

The term *pixel (px)*, or picture element, is a blanket term that is ambiguous without context. A pixel refers to, among others: a single element of an image sensor; a point sample of a digital image.

Specifically, single elements of an image sensor are differentiated using the terms: photosite, sensor element, or *sensel*. A sensel is responsible for converting incoming photons, from the lens, into electrons. The quantity of electrons found, over an exposure time, relates to the intensity reported by a sensel.[9] Then each sensel's full colour value is demosaiced (or interpolated from adjacent sensels), based on the image sensor's colour filter array.

A camera chip's image sensor has a rectangular array, or pixel count, of $W \times H$ sensels. The distance between each sensel, or a sensel's width or height, is called the pixel pitch, where ρ_w and ρ_h are the horizontal and vertical pitches, respectively.[10] These are usually specified in $\mu m/px$ [m/px]. Hence, the image sensor has a width and height of $W\rho_w$ and $H\rho_h$, respectively.

Unsurprisingly, the image sensor is modelled as a plane, which coincides with the back image plane. The sensor's frame is called the pixel coordinate frame $\{X\}$, with u and v axes that are parallel to a camera frame's x and y axes. Coordinates belonging to the pixel frame, or pixel coordinates, are measured in pixels.

On the front image plane, from the optical centre's point of view, the pixel frame sits towards the upper left direction (Figure 3.20).

8 Typically organised as Equation 3.15 (additive) or Equation 3.16 (multiplicative) in literature.

9 Reported intensity depends factors such as: quantum efficiency, lens aperture, shutter speed, fill factor, and analogue gain [4, 81, 88].

10 Image sensor performance metrics are discussed in [88].

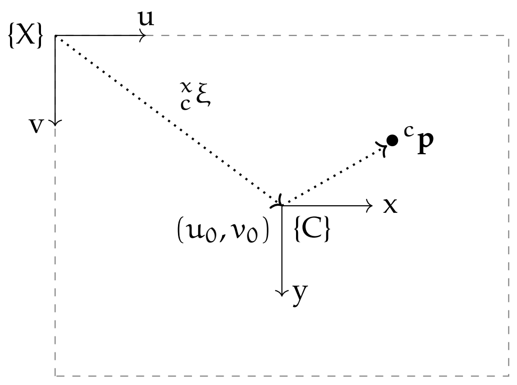


Figure 3.20: Pixel coordinate frame on the front image plane.

The image plane intersection with the optical axis is called its principle point.[11] This point, with respect to {X}, is (u_0, v_0) and it is specified in pixel coordinates. Pixel coordinates ${}^{x}\mathbf{p} = (u, v)$ are found by quantising image coordinates ${}^{c}\mathbf{p} = (x, y)$. In general,

$$
{}^{x}\bar{\mathbf{p}} = \begin{bmatrix} \frac{1}{\rho_w} & 0 & u_0 \\ 0 & \frac{1}{\rho_h} & v_0 \\ 0 & 0 & 1 \end{bmatrix} {}^{c}\bar{\mathbf{p}}. \tag{3.18}
$$

In practice, $u, v \notin \mathbb{Z}_{\geqslant 0}$ and must be rounded to a corresponding pixel, such that $u, v \in \mathbb{Z}_{\geqslant 0}$. Furthermore, pixel coordinates are stored digitally as point samples [89],[12] which are commonly rendered on digital displays as squares (Figure 3.21).

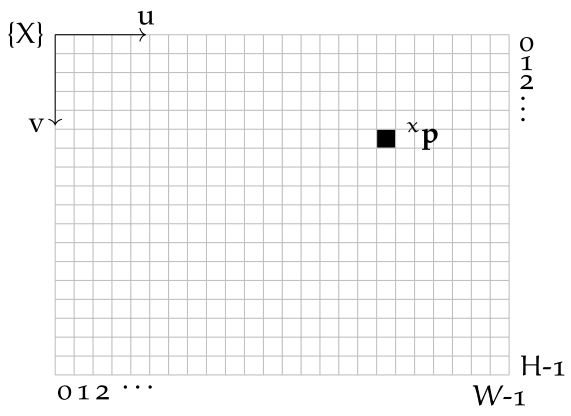


Figure 3.21: Square rendering of pixels.

By convention, a digital image's first and last addressable pixel coordinates are usually $(0, 0)$ and $(W - 1, H - 1)$, respectively.

11 The principle point is not necessarily the image sensor's geometric centre, due to manufacturing misalignment [84].

12 Image sensing pipeline and digital post-processing are discussed by Szeliski [81].

3.3.1 Field of View

A camera only captures light rays in its field of view. The field of view is parametrised by its horizontal and vertical angles, subtended by the width and height of an image sensor, respectively (Figure 3.22).

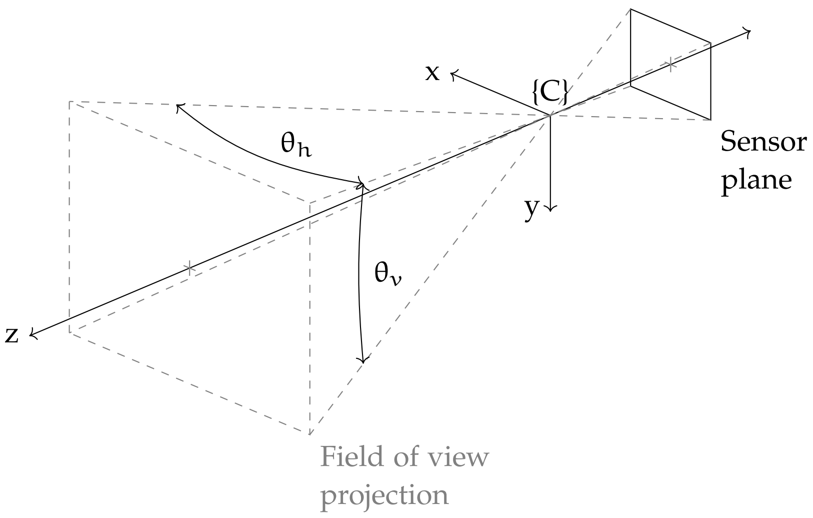


Figure 3.22: Vertical field of view.

The horizontal field of view [4] is

$$\tan\frac{\theta_h}{2} = \frac{\frac{W}{2}\rho_w}{f}$$

$$\theta_h = 2\tan^{-1}\frac{\frac{W}{2}\rho_w}{f}, \qquad (3.19)$$

and similarly, the vertical field of view

$$\theta_v = 2\tan^{-1}\frac{\frac{H}{2}\rho_h}{f}. \qquad (3.20)$$

The angles, θ_h and θ_v, decrease with increasing f, that is, a camera's field of view will narrow due to magnification.

3.3.2 Sensor Spatial Resolution

Visual detail can only be captured reliably, or resolved, if it lies within a sensel's field of view. The projection of a sensel's size $\rho_w \rho_h$ subtended by its field of view at $z = 1$ is called the sensel's (normalised) spatial resolutions. These are

$$\frac{P_w}{1} = \frac{\rho_w}{f}, \qquad (3.21)$$

$$\frac{P_h}{1} = \frac{\rho_h}{f}, \qquad (3.22)$$

with units of $(m/px)/m$. The spatial x-y resolutions decrease, or improve, by increasing f and losing field of view.[13]

If, instead, field of view is specified, then, from Equation 3.19,[14]

$$P_w = \frac{2}{W} \tan \frac{\theta_h}{2}, \tag{3.23}$$

$$P_h = \frac{2}{H} \tan \frac{\theta_v}{2}. \tag{3.24}$$

3.4 Intrinsic Parameters

Given Equation 3.17 and Equation 3.18, a complete camera model can be formed

$$\begin{aligned} {}^{x}\bar{\mathbf{p}} &= \begin{bmatrix} \frac{1}{\rho_w} & 0 & u_0 \\ 0 & \frac{1}{\rho_h} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{p}}_{dn} \\ &= \begin{bmatrix} \frac{f}{\rho_w} & 0 & u_0 \\ 0 & \frac{f}{\rho_h} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{p}}_{dn} \\ &= \mathbf{K}\bar{\mathbf{p}}_{dn}, \end{aligned} \tag{3.25}$$

where:

${}^{x}\bar{\mathbf{p}}$	augmented pixel coordinate
$\mathbf{K}$	camera matrix
$\bar{\mathbf{p}}_{dn}$	augmented distorted normalised image coordinate.

The camera matrix $\mathbf{K}$ and distortion coefficients are the intrinsic parameters of a camera. These parameters are determined by a camera calibration procedure.

The camera parameters f/ρ_w and f/ρ_h, or the pixel focal lengths [4], are lumped values, that is, f cannot be determined independently, unless ρ_w or ρ_h is known.

3.5 Extrinsic Parameters

A camera's extrinsic parameters represents its relative pose with respect to a world coordinate frame, be it another camera, or the base of a robot arm on which it is mounted [18]. Once the relative pose is found, world coordinates can be referenced by the camera's frame.

13 Overall spatial resolution depends on the cascaded sensor and lens spatial resolution. See http://photo.blogoverflow.com/2012/06/the-realities-of-resolution/.

14 This form is closer to Mallick et al. [66], called spatial x(y) resolution.

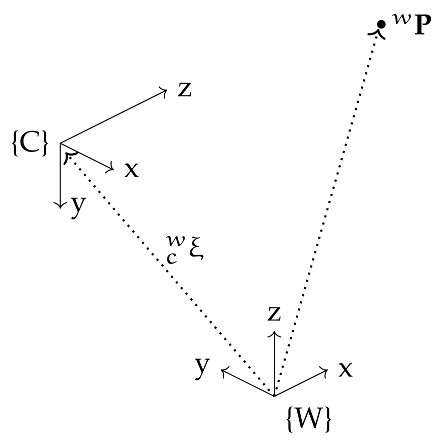


Figure 3.23: World coordinate frame.

In general, this is

$$^{c}\bar{\mathbf{P}} = {}^{c}_{w}\mathbf{T}\ {}^{w}\bar{\mathbf{P}} = {}^{w}_{c}\mathbf{T}^{-1}\ {}^{w}\bar{\mathbf{P}} \tag{3.26}$$

where:

$^{c}\bar{\mathbf{P}}$	augmented camera coordinate
$^{w}_{c}\mathbf{T}$	pose of the camera with respect to its world
$^{w}\bar{\mathbf{P}}$	augmented world coordinate.

The pose matrix $^{w}_{c}\mathbf{T}$ is the extrinsic parameter of a camera. Along with intrinsic parameters, extrinsic parameters can be determined by a camera calibration procedure.

3.6 Summary

A geometric camera model based on the pinhole camera has been presented. The result is a camera matrix which captures most of the important parameters. These are found using a suitable camera calibration procedure, after which image errors can be corrected.

4 KINECT CAMERA CHARACTERISATION

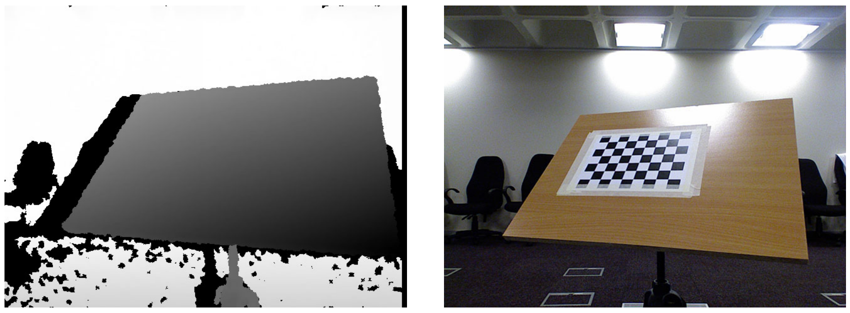

(a) Depth camera. (b) RGB camera.

Figure 4.1: Kinect point of view.

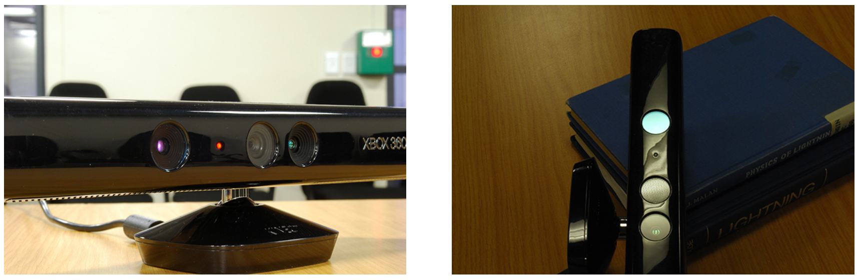
The Kinect has a RGB and depth camera, an IR camera and projector pair, which outputs uncalibrated images shown in Figure 4.1. RGB camera models and calibration procedures are well known [74], but characterising the depth camera requires more attention.

Due to the Kinect's proprietary nature, black box testing is necessary to characterise its performance and limitations. Disassembly is avoided, to prevent unintentional changes to its characteristics. Known characteristics about the RGB and depth camera, from various authors, are compared and discussed.

A corner-based calibration procedure is conducted using an assembled calibration rig and Herrera's Kinect Calibration Toolbox (KCT), which runs joint nonlinear minimisation [60] and corrects for depth distortion. The procedure is followed by a comparison and discussion of results between various data sets and authors.

4.1 Components

A LED indicates the Kinect's operational status (Figure 4.2a). The IR projector stands alone, emitting red light, followed by the RGB and IR cameras, respectively. The RGB or IR camera can be identified, by physically covering either one's view and observing a change on its data stream or captured image.



(a) Kinect operational. (b) IR projector filter.

Figure 4.2: Kinect cameras and projector.

A blue lid covers the IR camera's lens, possibly an IR bandpass filter, and the IR projector is shielded by a blue-tinted disk that becomes opaque from a viewing angle onwards (Figure 4.2b).

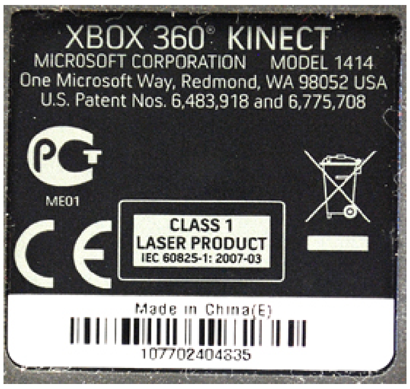

XBOX 360® KINECT
MICROSOFT CORPORATION MODEL 1414
One Microsoft Way, Redmond, WA 98052 USA
U.S. Patent Nos. 6,483,918 and 6,775,708
ME01
CLASS 1 LASER PRODUCT IEC 60825-1: 2007-03
Made in China(E)
107702404335

Figure 4.3: Kinect product information.

The Kinect's base reveals a model number, `MODEL 1414` (Figure 4.3).[1] Patents `US6483918` and `US6775708` are assigned to Microsoft Corporation and implement techniques for robust data transmission.[2] The IR projector is a class one laser and conforms to the eye safety standard, `IEC60825-1:2007-03`.

4.2 Specifications

Microsoft published only the Kinect for Windows specifications [90]. Useful metrics are unspecified, such as pixel count, frame rate, depth resolution, and depth accuracy. Han et al. [55], OpenKinect's protocol documentation [91], and OpenKinect's imaging information [92] have provided some specifications. These are combined into Table 4.1.

1 Model numbers are unmentioned by any known author.
2 Patent search is available at `https://www.google.com/patents`.

Table 4.1: Kinect documented specifications.

SPECIFICATION	DESCRIPTION
RGB-full stream [55, 91]	1280×1024 px, 10 fps
RGB-standard stream [55, 91]	640×480 px, 30 fps
RGB data format [55]	8 bits/px/channel
Depth stream [55, 91]	640×480 px, 30 fps
Depth data format [91]	16 bits/px, 11 bits used
Field of view [55]	57° horizontal 43° vertical
Depth spatial resolution (at 2 m) [92]	3 mm
Practical depth range [55, 92]	0.8 m to 3.5 m
Depth resolution (at 2 m) [92]	10 mm

Since spatial resolution scales linearly, the expected depth spatial resolution at 1.0 m is 1.5 mm.

The Kinect is capable of extending beyond the 3.5 m practical depth range, but at the loss of depth resolution. The actual usable depth range is limited by the Kinect's maximum accessible depth and an application's depth error tolerance.

The expected uncompressed data sizes and rates for the three streams are calculated in Table 4.2, signalling the required throughput.

Table 4.2: Kinect image data sizes and data rates.

TYPE	DATA SIZE (MB)	DATA RATE (MB/s)
RGB-full	3.932	39.32
RGB-standard	0.922	27.65
Depth	0.614	18.43

4.3 Depth Camera Characterisation

4.3.1 IR Speckle Pattern

Along with RGB and disparity images, the Kinect outputs IR images. These record the IR projector's speckle pattern, which act as intermediaries in the depth processing pipeline, and do nothing more. Nonetheless, the IR images explain the some of the depth sensor's limitations, which are not directly observable on disparity images.

An IR image, captured with the Kinect facing a white plastered wall, is shown in Figure 4.4a. The IR speckle pattern is superimposed with a visible distortion. This distortion is not seen when captured by an external IR camera [93], hence it is on-board the IR camera.

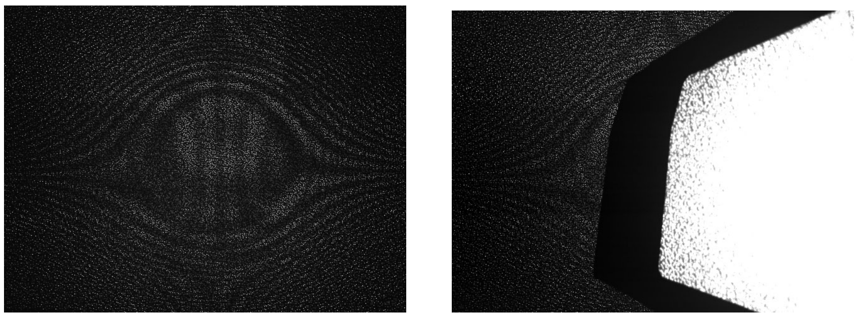

(a) Distorted speckle pattern. (b) Near range and occlusion.

Figure 4.4: Kinect IR images.

When objects approach the IR camera, the speckle pattern's dots clump together (Figure 4.4b) and they cannot be uniquely identified. Given one or more unidentifiable dots, depth cannot be measured properly, demonstrating the Kinect's near range limitation.

Since the speckle pattern is assembled with dots, the space between dots should have interpolated depth values. This is an issue for large depth discontinuities, such as an object against a far background, which have fluctuating depth values even when the Kinect and its scene are stationary.[3] By design, the Kinect excels at real-time skeletal tracking [54]. Therefore, planar features should be considered first.

There is a relative pose between the IR camera and projector. The speckle pattern's dots will be occluded by the visible part of objects. As a result, objects in IR or disparity images will always cast a shadow behind them (Figure 4.4b).[4] These are compensated for, by using a suitable depth hole filtering technique [55].

4.3.2 Disparity

The relative shift between captured and reference speckle patterns is used to generate the (greyscale) disparity image, where each pixel represents Kinect disparity, measured in kdu. The relationship between Kinect disparity and depth [60] is

$$z(d) = \frac{1}{c_1 d + c_0}, \tag{4.1}$$

3 Mallick et al. [66] call this *lateral noise*.
4 Mallick et al. [66] call this *shadow noise*.

where z is depth [m], d is Kinect disparity [kdu], c_0 and c_1 are the disparity coefficients. The coefficient c_1 is always negative [60].

Kinect disparity is an 11-bit number, that is, $d \in [0, 2047] \cap \mathbb{Z}_{\geqslant 0}$. The usable interval for this disparity model is much smaller, since it is useful only when $z(d) \geqslant 0$, or rather $z(d)$ has a singularity at

$$d = \frac{c_0}{-c_1} := d_s.$$

This is the disparity model's limitation. It does not represent the IR camera's ability to resolve speckle pattern dots at far range, which is far more limiting than the model's limitation, but acts as a sanity test for using Equation 4.1. As an example, the KCT's documentation posts $c_0 = 3.12$ and $c_1 = -0.002855$. Then $\lfloor d_s \rfloor = 1092$ and $z(d) = 427$.

4.3.3 Spatial Disparity Distortion

The disparity distortion model used by Herrera et al. [60] is

$$d_k = d + \mathbf{D}_\delta(u, v) \exp(\alpha_0 - \alpha_1 d), \tag{4.2}$$

where d_k is the undistorted disparity, d is Kinect disparity [kdu], α_0 and α_1 are the distortion alpha coefficients, $\mathbf{D}_\delta(u, v)$ is a $640 \times 480\,\text{px}$ sized image of distortion beta coefficients [kdu].

The undistorted depth is then

$$z_k(d) = \frac{1}{c_1 d_k + c_0}, \tag{4.3}$$

which has a singularity at

$$d_k = \frac{c_0}{-c_1}.$$

This can be solved using the same method that [60] uses to solve for their forward disparity distortion model, that is, using the Lambert $\mathcal{W}$ function.[5] Hence, d_s with disparity correction becomes

$$d_{ks} = d_s + \frac{\mathcal{W}(-\alpha_1 \mathbf{D}_\delta(u, v) \exp(\alpha_0 - \alpha_1 d_s))}{\alpha_1}. \tag{4.4}$$

The disparity and distortion coefficients are subject to uncertainty in the form $c_0 = \bar{c}_0 \pm \sigma_{c_0}$, where $\bar{c}_0$ is the mean value of c_0, and σ_{c_0} is its standard deviation. In turn, Equation 4.3 is subject to uncertainty propagation. For expediency, the three point estimate is used

$$z_{k\,\min}(d) \leqslant \bar{z}_k(d) \leqslant z_{k\,\max}(d), \tag{4.5}$$

5 The KCT has an existing `lambertw_fast()` implementation.

where:

$$\begin{aligned} z_{k\,\max}(d) &= \frac{1}{(\bar{c}_1 - \sigma_{c_1})d_{k\,\max} + (\bar{c}_0 - \sigma_{c_0})}, \\ \bar{z}_k(d) &= \frac{1}{\bar{c}_1 d + \bar{c}_0}, \\ z_{k\,\min}(d) &= \frac{1}{(\bar{c}_1 + \sigma_{c_1})d_{k\,\min} + (\bar{c}_0 + \sigma_{c_0})}, \end{aligned}$$

and

$$\begin{aligned} d_{k\,\max} &= d + \mathbf{D}_\delta(u,v)\exp((\bar{\alpha}_0 + \sigma_{\alpha_0}) - (\bar{\alpha}_1 - \sigma_{\alpha_1})d), \\ d_{k\,\min} &= d + \mathbf{D}_\delta(u,v)\exp((\bar{\alpha}_0 - \sigma_{\alpha_0}) - (\bar{\alpha}_1 + \sigma_{\alpha_1})d). \end{aligned}$$

4.3.4 Depth Resolution

The minimum measurable change in depth, or quantisation error, between Kinect disparity values is called the depth resolution. Experimental [61] and theoretical [65] quantisation errors are modelled using univariate quadratic polynomials (Table 4.3, Figure 4.5).

Table 4.3: Kinect depth resolution.

AUTHOR	DEPTH RESOLUTION (mm), DEPTH (m)
Smisek et al. [61]	$q(z) = 2.73z^2 + 0.74z - 0.580$
Macknojia et al. [65]	$q(z) = 3.02z^2 - 0.56z + 0.307$

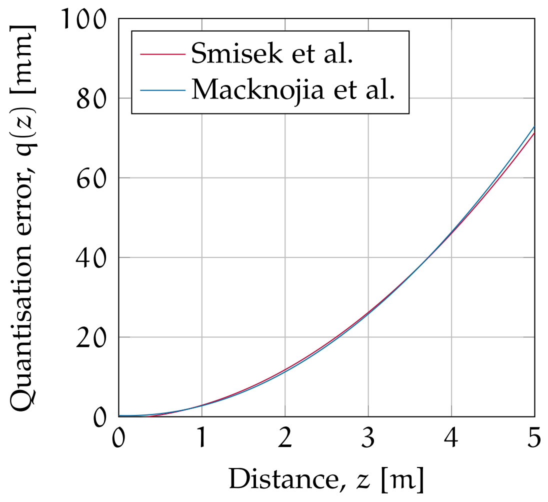


Figure 4.5: Depth resolution.

By definition, the quantisation error is

$$\Delta z(d) = z(d+1) - z(d), \quad d = 0, \dots, \lfloor d_s \rfloor - 1. \tag{4.6}$$

4.4 Calibration Experiment

Corner-based calibration uses images of a calibration target with detectable markers and known constraints to determine camera model parameters. A chessboard is a common calibration target, where the intersection between every four blocks, or corners, are assumed to be planar. This topic is well covered by Zhang [73] and Heikkilä [94].

The KCT runs within the MATLAB environment. Images are captured using the included image capture tool. A calibration rig is assembled with a checkerboard pattern for RGB camera calibration and planar surface for depth camera calibration.

4.4.1 Image Capturing Tool

The KCT (v2.2) is packaged with the source code for a capture tool, used to capture pairs of Portable Pix Map (PPM) and Portable Grey Map (PGM) images. Project files are generated for Visual Studio 12 using CMake 2.8.12.2 and built using Visual Studio 2013. The dependencies used to build the capture tool are listed below:

- `libfreenect (2014-04-03)`
- `opencv-2.2.0`
- `boost_1_55_0`
- `libusb-win32-bin-1.2.6.0`
- `pthreads-w32-2-9-1-release`
- `glui-2.36`
- `glut-3.7.6-bin`
- `glext.h (2014-03-19)`

4.4.2 Calibration Rig

A 9-by-7 squared checkerboard pattern is printed on A4 paper and taped flush against an A2-sized wooden board.[6] There are 48 squares, where each square has a side length of 27.3 mm. Then, the wooden board is secured to a camera tripod. The full setup is shown in Figure 4.6a. Additionally, a flat wall is used as a full planar surface to calibrate the depth camera (Figure 4.6b).

6 MATLAB's Image Processing Toolbox has a `checkerboard()` function. Otherwise, use the file from Bouguet's toolbox, or a checkerboard pattern generator.

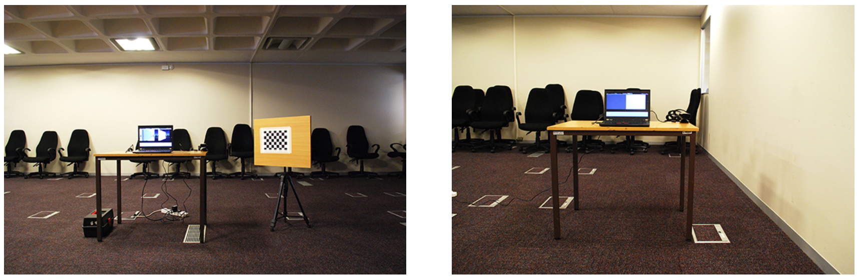

(a) Calibration rig setup. (b) Full planar surface.

Figure 4.6: Full calibration setup.

4.4.3 Procedure

Guidelines by Fiedler and Heinrich [79] are used to control measurement errors due to temperature. The Kinect is given a 60 minute warm-up period to reach stable temperatures and then calibrated in a room air-conditioned to 23 °C.

This procedure requires five types of camera shots at varying angles and distances. These are: the checkerboard pattern for each of three different plane rotations (Figure 4.7); full-viewed checkerboard pattern images (for lens distortions, only RGB images); and fronto-parallel wall images (for disparity distortion, only disparity images). At least five images for each camera shot type are recommended [15].[7]

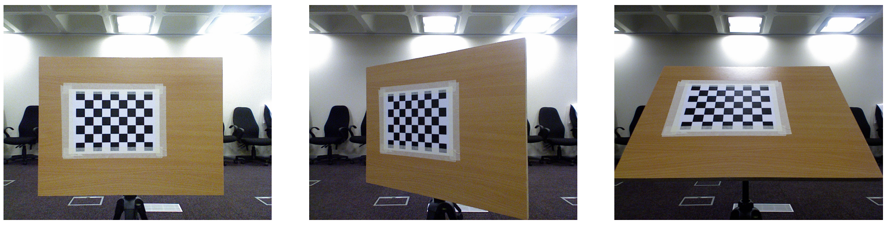
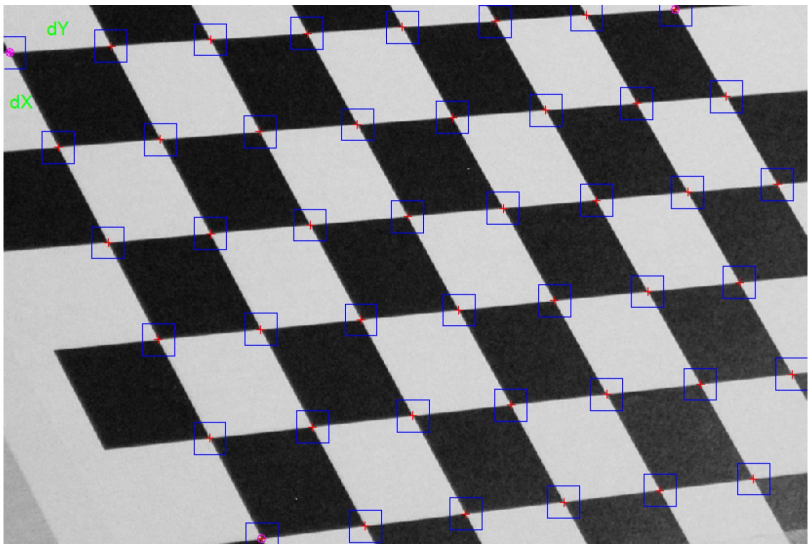

(a) Frontal plane. (b) y-axis rotation. (c) x-axis rotation.

Figure 4.7: Calibration target plane rotations.

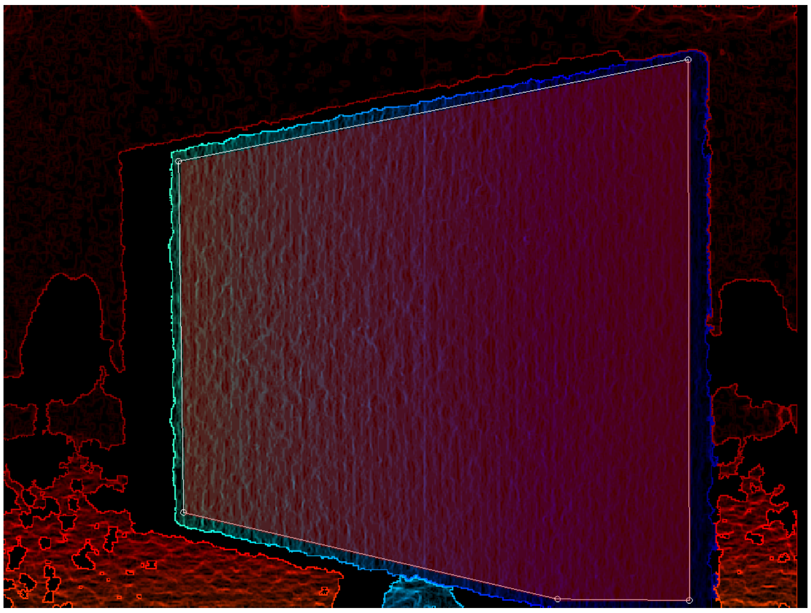
After image capturing, PPM images are compressed to JPEG format.[8] RGB images undergo corner extraction. Automatic corner detection is initiated by specifying the physical checkerboard square size, expected corner count, and pixel window size (Figure 4.8a).

Lastly, plane selection is done by manually selecting the usable, non-edge, planar region of the wooden board and full planar surface in each disparity image (Figure 4.8b).

7 If unsure, see the dataset included with KCT.
8 Any MATLAB readable format is fine. The KCT included dataset uses JPEG images. Compressed images are used to speed-up the calibration process.



(a) Automatic corner detection.



(b) Plane selection.

Figure 4.8: Corner extraction and plane selection of calibration rig.

4.5 Calibration Results

Calibration results for RGB-full and -standard data sets, one each, are tabulated over Table 4.4 and Table 4.5. The errors, or uncertainties, provided by the KCT are three times the standard deviation.

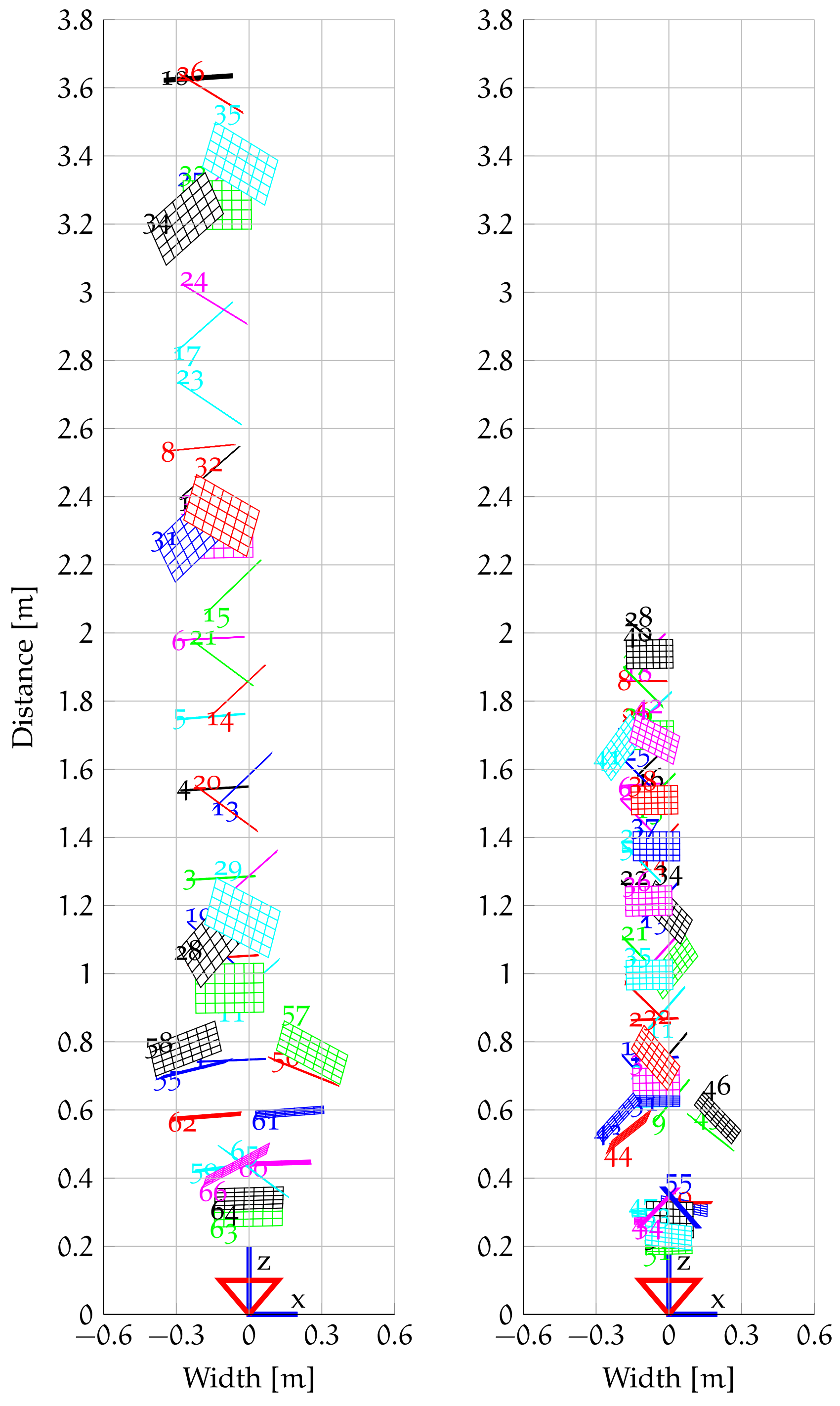
Table 4.4: RGB camera calibration results.

PARAMETER	SYMBOL	RGB-FULL	RGB-STANDARD
Pixel focal length (px)	f/ρ_w	1045.09 ± 0.38	522.27 ± 0.19
	f/ρ_h	1045.26 ± 0.38	522.35 ± 0.19
Principle point (px)	u_0	633.41 ± 0.29	315.75 ± 0.18
	v_0	517.31 ± 0.28	258.58 ± 0.17
Distortion coefficient	k_1	0.2296 ± 0.0012	0.2321 ± 0.0016
	k_2	-0.6401 ± 0.0049	-0.6463 ± 0.0067
	p_1	0.0001 ± 0.0001	0.0005 ± 0.0001
	p_2	0.0014 ± 0.0001	0.0012 ± 0.0001
	k_3	0.5645 ± 0.0062	0.5668 ± 0.0088
Reprojection error (px)	μ	0	0
	σ	0.2082	0.1405

Table 4.5: Depth camera calibration results.

PARAMETER	SYMBOL	RGB-FULL	RGB-STANDARD
Pixel focal length (px)	f/ρ_w	580.18 ± 0.18	584.62 ± 0.36
	f/ρ_h	587.81 ± 0.51	591.86 ± 0.54
Principle point (px)	u_0	310.47 ± 0.32	308.80 ± 0.36
	v_0	239.51 ± 0.27	237.65 ± 0.32
Disparity coefficient	c_0	3.16	3.15
	c_1	$-0.002\,899$	$-0.002\,883$
Distortion α coefficient	α_0	1.3716 ± 0.0332	1.6369 ± 0.0330
	α_1	0.0025 ± 0.0001	0.0025 ± 0.0001
Distortion β coefficients	$\mathbf{D}_\delta(u,v)$	$\{-10.2044, \ldots, 5.3239\}$	$\{-8.5699, \ldots, 5.1769\}$
Reprojection error (kdu)	μ	0.0120	0.0023
	σ	0.6971	0.6950

The RGB-full data set has 47 RGB and 40 disparity images, and the RGB-standard data set has 55 RGB and 52 disparity images. RGB images from both data sets are visualised in Figure 4.9.[9]



(a) RGB-full. (b) RGB-standard.

Figure 4.9: Kinect-centred, top-down view of plane rotation data sets.

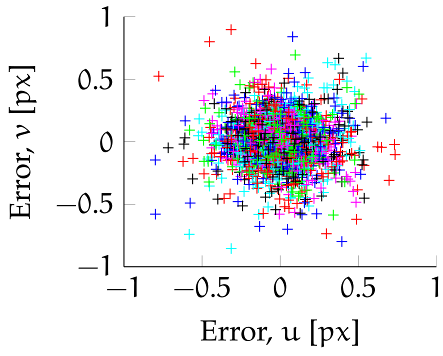
9 Using the *Show Extrinsic* function from Bouguet's toolbox.

4.5.1 RGB Reprojection Error

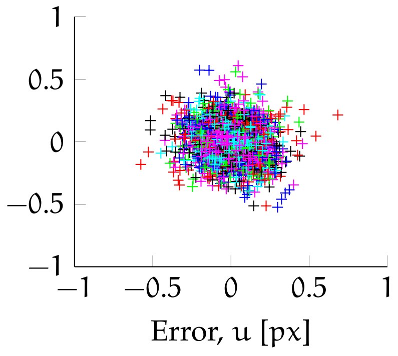
RGB corner reprojection errors for Herrera et al. [60] (calibration without the external camera) and data sets from KCT (external camera parameters removed) are compared in Table 4.6. Sub-pixel standard deviations indicate accurate calibration [60]. Corner reprojection errors for RGB-full and -standard data sets are plotted in Figure 4.10.[10]

Table 4.6: RGB reprojection error comparison between data sets.

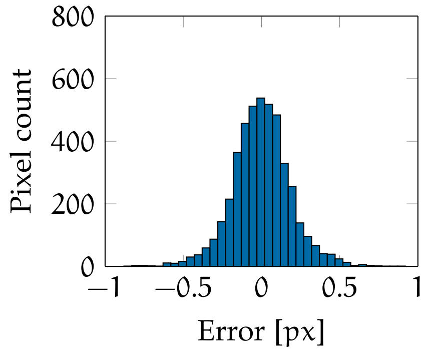
DATA SET	IMAGE COUNT	STD. DEV. (px)
RGB-full	47 RGB, 40 disparity	0.21
RGB-standard	55 RGB, 52 disparity	0.14
KCT [15] `full_set`	54 RGB, 55 disparity	0.21
KCT [15] `small_set`	11 RGB, 18 disparity	0.26
Herrera et al. [60]	60 pairs	0.26



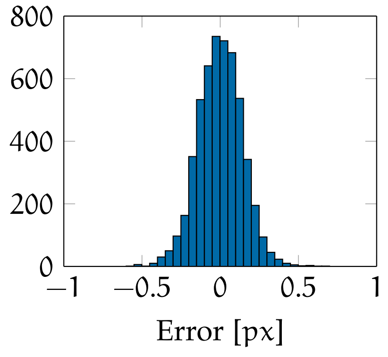
(a) RGB-full.



(b) RGB-standard.



(c) RGB-full.



(d) RGB-standard.

Figure 4.10: Plots and histograms of RGB corner reprojection errors.

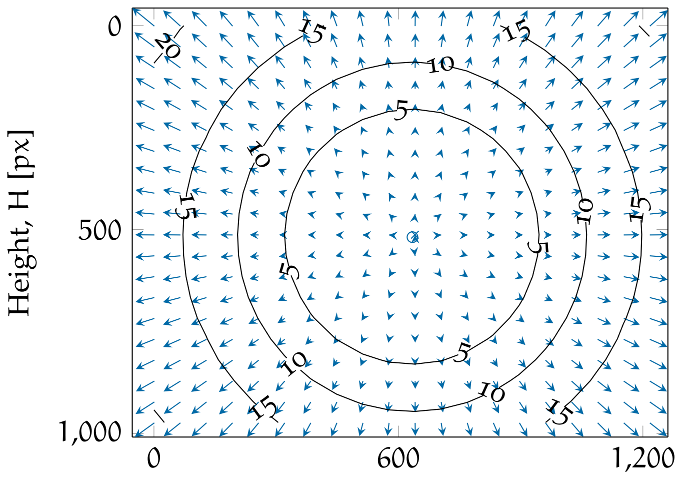
10 For an in-depth look at reprojection errors, see the *Main Calibration step* of `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html`.

Images with large reprojection errors are borderline field of view and full-viewed. This is expected, since distortion components are known to be stronger towards the edge and weak around the centre.

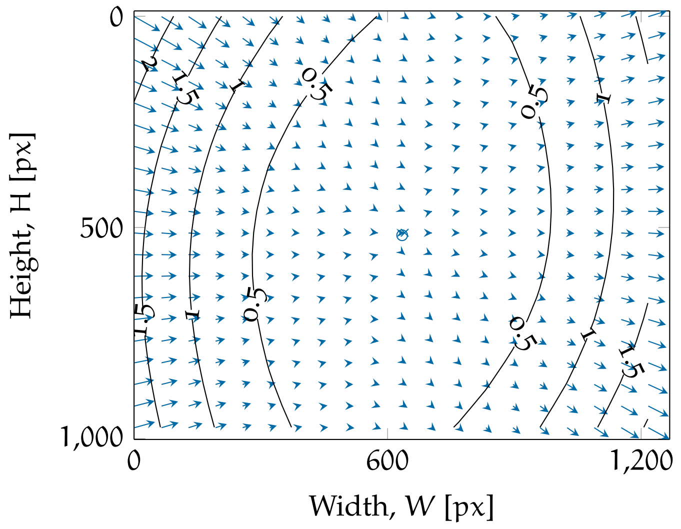
The RGB-standard data set has eight more images than the RGB-full data set, that is, it has 384 additional corner reprojection errors.

4.5.2 RGB Lens Distortion

Lens distortions for both RGB modes are very similar. Hence, only the distortions for RGB-full are visualised (Figure 4.11).[11] The tangential distortions are small, compared to their radial counterparts.



(a) Radial components.



(b) Tangential components.

Figure 4.11: RGB-full lens distortions.

11 Using the `visualize_distortions()` function from Bouguet's toolbox .

As an example, a plane rotation image is undistorted using the distortion model to verify the undistortion (Figure 4.12).[12]

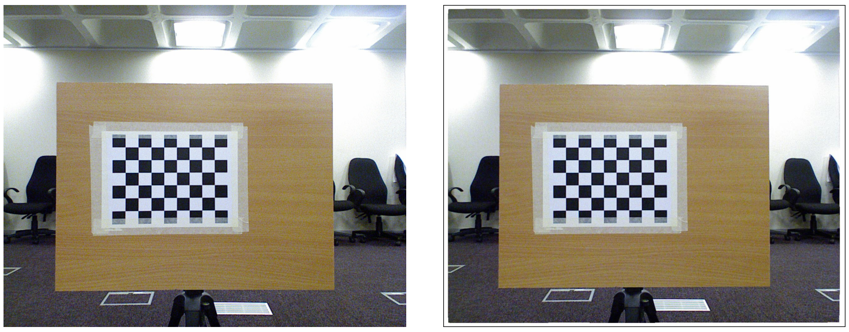

(a) Distorted image. (b) Undistorted image.

Figure 4.12: Undistorting an image using the distortion model.

The RGB-full and KCT `full_set` lens distortions are compared in Table 4.7. Since other RGB camera parameters are expected to be similar, Kinect RGB camera variability is largely due to lens distortions.

Table 4.7: RGB lens distortion coefficient comparison between data sets.

DATA SET	k_1	k_2	p_1	p_2	k_3
RGB-full	0.2296	−0.6401	0.0001	0.0014	0.5645
KCT [15] `full_set`	0.1841	−0.4726	−0.0030	0.0014	0.3636

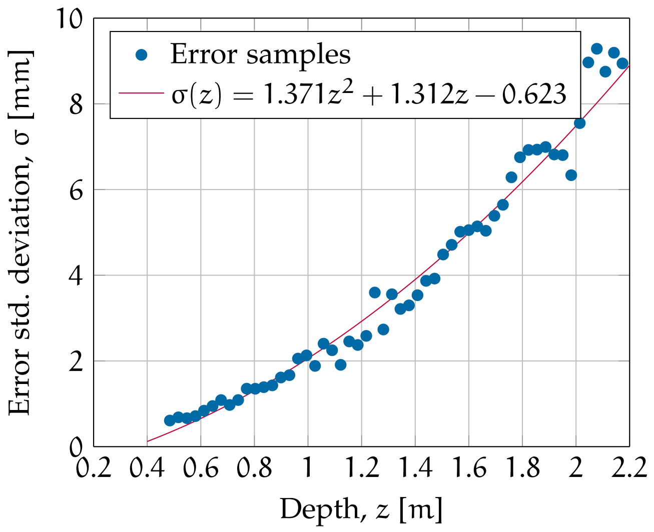
4.5.3 Depth Reprojection Error

Depth reprojection errors are compared in Table 4.6. Sub-disparity-unit standard deviations indicate accurate calibration [60]. Depth errors for both RGB data sets are plotted in Figure 4.13.

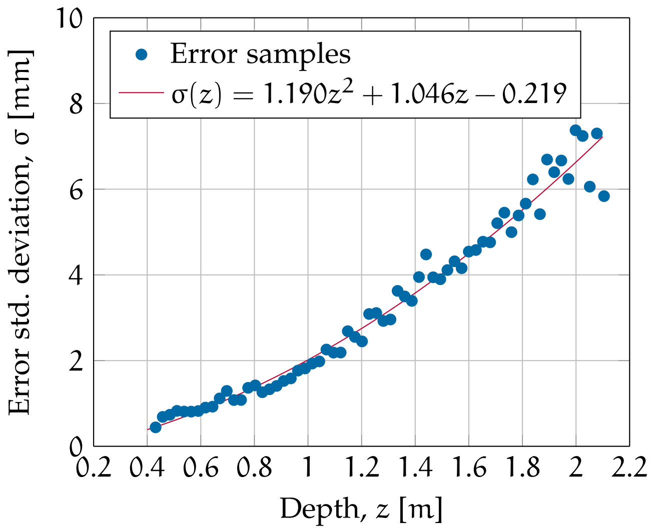
Table 4.8: Depth reprojection error comparison between data sets.

DATA SET	IMAGE COUNT	STD. DEV. (kdu)
RGB-full	47 RGB, 40 disparity	0.697
RGB-standard	55 RGB, 52 disparity	0.695
KCT [15] `full_set`	54 RGB, 55 disparity	0.772
KCT [15] `small_set`	11 RGB, 18 disparity	0.794
Herrera et al. [60]	60 pairs	0.765

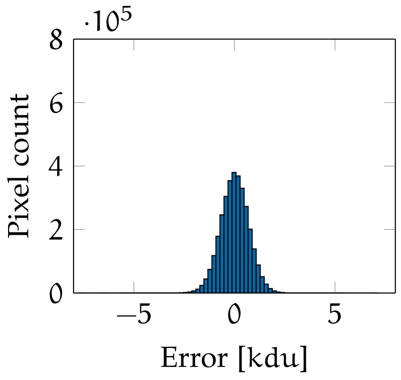
12 Using the *Undistort image* function from Bouguet's toolbox.



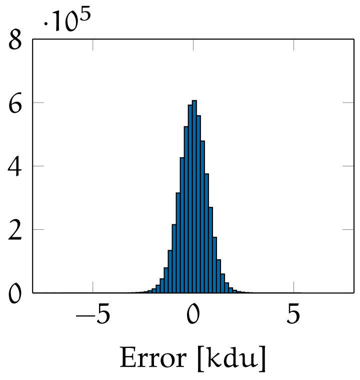
(a) RGB-full.



(b) RGB-standard.



(c) RGB-full.



(d) RGB-standard.

Figure 4.13: Plots and histograms of depth reprojection errors.

4.5.4 Spatial Disparity Distortion

The absolute value of the spatial disparity distortion image $\mathbf{D}_\delta(u, v)$ for RGB-full, RGB-standard, and the KCT `full_set` are shown in Figure 4.14. The RGB data set images have the same spatial distortion.

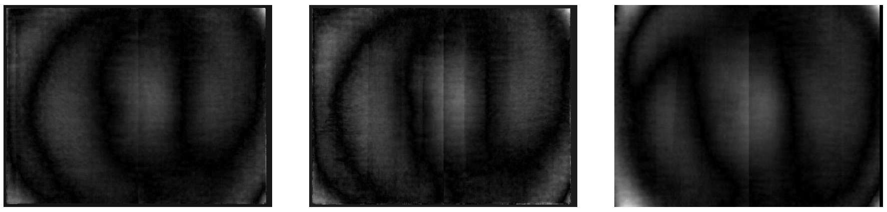

(a) RGB-full. (b) RGB-standard. (c) KCT [15] `full_set`.

Figure 4.14: Absolute valued spatial disparity distortion images $\mathbf{D}_\delta(u, v)$.

At a glance, the KCT `full_set`'s spatial distortion is different to the other two. Hence, distortion α and β coefficient variability will contribute to Kinect depth camera variability.

4.5.5 Disparity and Depth Resolution

The disparity coefficients, c_0 and c_1, from different data sets are compared in Table 4.9. These are similar across different Kinects, even though only the first five data sets use joint minimisation.

Table 4.9: Disparity coefficient comparison between data sets and authors.

DATA SET	COEFFICIENT c_0	COEFFICIENT c_1
RGB-full	3.16	−0.002 899
RGB-standard	3.15	−0.002 883
KCT [15] `full_set`	3.13	−0.002 859
KCT [15] `small_set`	3.12	−0.002 854
Herrera et al. [71]	3.11	−0.002 850
Smisek [64, pg. 37]	3.15	−0.002 845

The disparity model and depth resolution for RGB-full is plotted in Figure 4.15. The upper and lower bounds use $\max_{u,v} \mathbf{D}_\delta(u, v)$ and $\min_{u,v} \mathbf{D}_\delta(u, v)$ for $z_{k\,\max}(d)$ and $z_{k\,\min}(d)$, respectively, and 3σ for the uncertainties. Here, the quantisation error is plotted as a function of disparity. Since disparity is quantised, depth is also quantised, thus some depth values are unachievable. These results are comparable to the depth resolutions in Table 4.1 and Table 4.3.

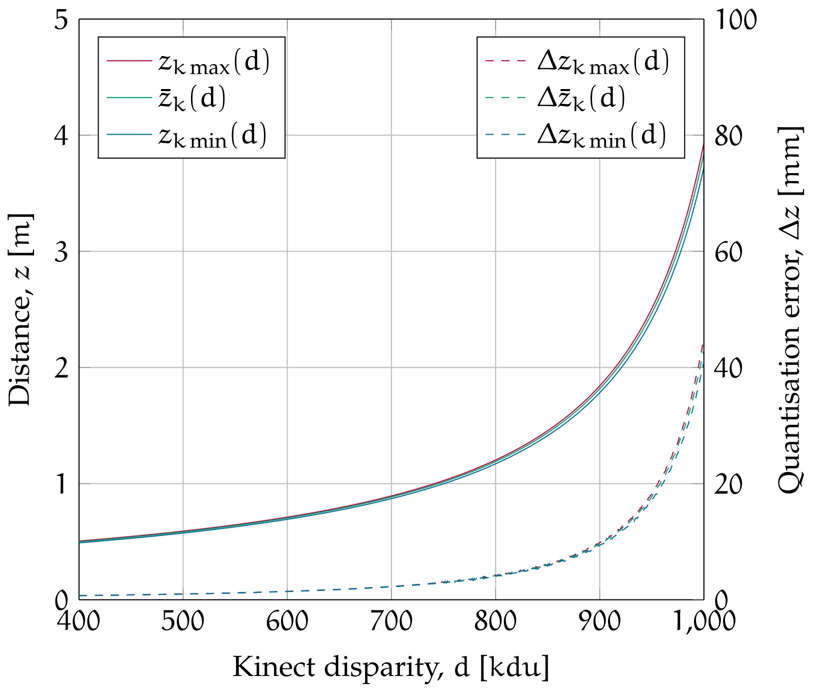


Figure 4.15: RGB-full disparity, depth and depth resolution.

Lastly, the disparity model singularities for RGB-full and -standard are calculated using $\max_{u,v} \mathbf{D}_\delta(u, v)$ and given in Table 4.10.

Table 4.10: Disparity model singularity for RGB-full and -standard.

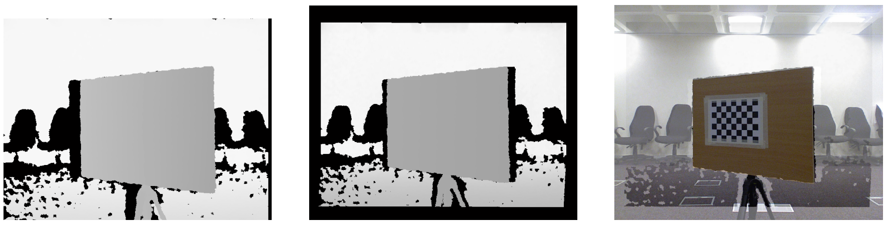
DATA SET	DISPARITY d_s (kdu)	DISPARITY d_{ks} (kdu)
RGB-full	1090.03	1088.65
RGB-standard	1092.61	1090.87

4.5.6 Extrinsic Parameters

The relative pose between the RGB and IR camera from the RGB-full calibration is

$$ {}^{\mathrm{RGB}}\mathbf{T}_{\mathrm{IR}} = \begin{bmatrix} 1.00000 & -0.00454 & -0.00260 & -0.02883 \\ 0.00454 & 1.00000 & 0.00190 & 0.00279 \\ 0.00259 & -0.00192 & 1.00000 & -0.00360 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

The relative pose is used to register the depth camera's coordinate frame into the RGB camera's frame, such that overlapping pixels from both images describe the same world point (Figure 4.16).



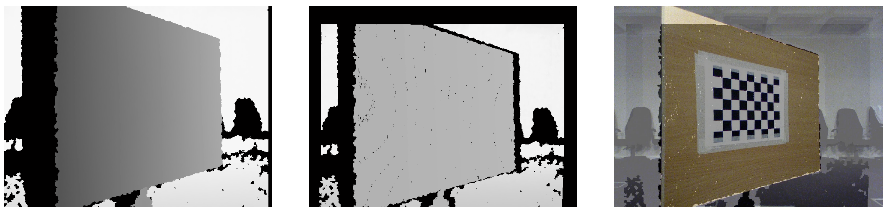
(a) Disparity image. (b) Registered. (c) Superimposed.

Figure 4.16: RGB-full image registration using extrinsic parameters.

Similarly, the relative pose for RGB-standard is

$$
{}^{\text{RGB}}\mathbf{T}_{\text{IR}} = \begin{bmatrix} 1.00000 & -0.00151 & 0.00031 & -0.03167 \\ 0.00151 & 1.00000 & 0.00034 & 0.00347 \\ -0.00031 & -0.00034 & 1.00000 & -0.00773 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

along with its disparity image registration in Figure 4.17.



(a) Disparity image. (b) Registered. (c) Superimposed.

Figure 4.17: RGB-standard image registration using extrinsic parameters.

4.5.7 Post-calibration Specifications

Table 4.11 and Table 4.12 present both RGB data sets' field of views and spatial x-y resolutions. These verify the specifications in Table 4.1.

Table 4.11: Kinect post-calibration RGB specifications.

SPECIFICATION	SYMBOL	RGB-FULL	RGB-STD
Field of view (deg)	θ_h	62.97 ± 0.02	62.99 ± 0.02
	θ_v	52.19 ± 0.02	49.35 ± 0.02
Spatial resolution (mm/px/m)	P_w	0.96	1.91
	P_h	0.96	1.91

The RGB-standard image appears to be a half-sampled RGB-full image that is cropped at the bottom (Figure 4.18), which is due to the change in aspect ratio from 5 : 4 to 4 : 3. This accounts for the loss in vertical field of view from RGB-full to -standard streams.

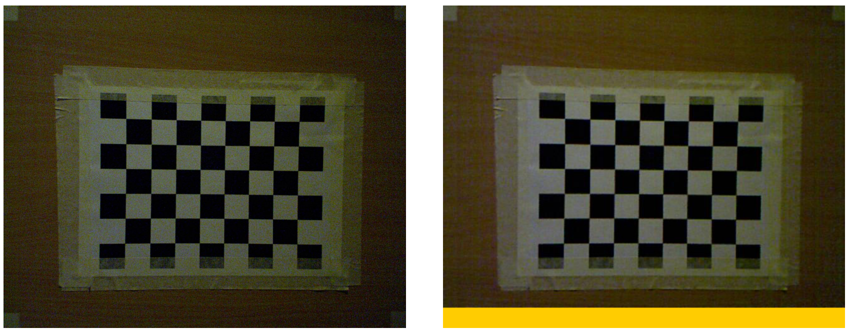

(a) RGB-full. (b) RGB-standard.

Figure 4.18: Comparing RGB-full and -standard colour image sizes. The yellow strip indicates pixels cropped by the Kinect.

The differences between the RGB-full and -standard pixel focal lengths are due to an *apparent* doubling of the RGB-standard pixel pitches. Hence, its spatial resolution is about twice that of RGB-full.

Table 4.12: Kinect post-calibration depth specifications.

SPECIFICATION	SYMBOL	RGB-FULL	RGB-STD
Field of view (deg)	θ_h	57.75 ± 0.02	57.39 ± 0.03
	θ_v	44.42 ± 0.03	44.16 ± 0.04
Spatial resolution (mm/px/m)	P_w	1.72	1.71
	P_h	1.70	1.69

Ultimately, the Kinect's useful field of view is limited by its depth camera's field of view, as disparity images do not cover their entire RGB counterparts. This is emphasised in Figure 4.17.

4.6 Summary

A thorough characterisation of the Kinect has been presented. Calibration results reveal that RGB and depth camera variability is largely due to lens distortions and α and β coefficients, respectively, when compared to the KCT datasets, Herrera et al. [71], and Smisek [64]. While temperature control guidelines were followed, final calibration results do not appear to differ from the results of other authors.

5 VISUAL SERVOING IMPLICATIONS

Tutorials for PBVS and IBVS are covered briefly so that the relationship between visual servoing and the Kinect can be drawn. Conversely, knowledge of the Kinect's characterisation is used to explore its implications to visual servoing.

5.1 Basic Visual Servoing Techniques

Most visual servoing systems assume that the underlying robot is internally stabilised by fine-tuned tight control loops [4]. These robot controllers are driven by velocity commands. Only single camera eye-in-hand systems are discussed.

5.1.1 Task Function

The task function approach to visual servoing is to minimise the task function error [19]

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*, \tag{5.1}$$

where $\mathbf{s}(\mathbf{m}(t), \mathbf{a})$ is a set of extracted visual features, $\mathbf{m}(t)$ is a set of image measurements, $\mathbf{a}$ represents the camera's characteristics, and $\mathbf{s}^*$ is a set of reference visual features.

Consider the relationship between visual feature velocity $\dot{\mathbf{s}}$ and camera spatial velocity $\boldsymbol{\upsilon} = (\boldsymbol{\nu}, \boldsymbol{\omega}) = (\nu_x, \nu_y, \nu_z, \omega_x, \omega_y, \omega_z)$ in the world coordinate frame [19]

$$\dot{\mathbf{s}} = \mathbf{L}\boldsymbol{\upsilon}, \tag{5.2}$$

where $\mathbf{L}$ is the image Jacobian, or interaction matrix, which relates $\dot{\mathbf{s}}$ to $\boldsymbol{\upsilon}$. By solving for $\boldsymbol{\upsilon}$, a velocity controller can be designed [19]

$$\boldsymbol{\upsilon} = \mathbf{L}^+\dot{\mathbf{s}}. \tag{5.3}$$

where $\mathbf{L}^+ = (\mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T$ is the pseudo-inverse of $\mathbf{L}$. If $\mathbf{s}^*$ is constant, then $\dot{\mathbf{s}} = \dot{\mathbf{e}}$. The following condition is imposed [19]

$$\dot{\mathbf{e}} = -\lambda\mathbf{e}, \tag{5.4}$$

which makes the task function error decay exponentially to zero. In other words, $\mathbf{s}$ approaches $\mathbf{s}^*$ exponentially

$$\mathbf{s} = \mathbf{s}^*(1 - \exp(-\lambda t)), \tag{5.5}$$

where $\lambda > 0$ is a chosen rate. It follows that

$$\boldsymbol{\upsilon} = \mathbf{L}^+\dot{\mathbf{e}} = -\lambda\mathbf{L}^+\mathbf{e} = -\lambda\mathbf{L}^+(\mathbf{s} - \mathbf{s}^*). \tag{5.6}$$

Since $\mathbf{L}$ and $\mathbf{L}^+$ are impossible to know perfectly [19], $\mathbf{L}^+$ has to be approximated. This approximation is denoted $\widehat{\mathbf{L}^+}$, which results in the velocity controller, or control law [19]

$$\boldsymbol{\upsilon} = -\lambda\widehat{\mathbf{L}^+}(\mathbf{s} - \mathbf{s}^*). \tag{5.7}$$

The approximation used for $\widehat{\mathbf{L}^+}$ affects the behaviour of the visual feature trajectories.

5.1.2 Image-Based Visual Servoing

An IBVS task function using point features has $\mathbf{s}$ and $\mathbf{s}^*$ as the current and reference image points $\mathbf{p} = (x, y)$ and $\mathbf{p}^*$, respectively. The task function parameters are $\mathbf{m} = (u, v)$, the pixel coordinates of $\mathbf{p}$, and $\mathbf{a} = (f, \rho_w, \rho_h, u_0, v_0)$, the intrinsic camera parameters.

Each image taken by the camera requires each image point feature to be extracted and matched to its corresponding reference image point. Techniques for feature extraction and matching are discussed in [4, 81]. An overview of IBVS is illustrated in Figure 5.1.

To solve for the velocity controller, find the interaction matrix for Equation 5.2 where $\mathbf{s} = \mathbf{p}$

$$\dot{\mathbf{p}} = \mathbf{L}_p\boldsymbol{\upsilon}. \tag{5.8}$$

Recall that the perspective projection equations for normalised image coordinate $\mathbf{p}_n$, where the focal length $f = 1$, are

$$\mathbf{p}_n = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix}. \tag{5.9}$$

Using the quotient rule, the temporal derivative of $\mathbf{p}_n$ is

$$\begin{aligned} \dot{x} &= \frac{\dot{X}Z - X\dot{Z}}{Z^2} \\ \dot{y} &= \frac{\dot{Y}Z - Y\dot{Z}}{Z^2}. \end{aligned} \tag{5.10}$$

Now, a world point $\mathbf{P} = (X, Y, Z)$ that is being observed by a camera in the world frame with instantaneous linear velocity $\boldsymbol{\nu} = (\nu_x, \nu_y, \nu_z)$

and instantaneous angular velocity $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ has a velocity relative to the camera frame that is [19]

$$\dot{\mathbf{P}} = -\boldsymbol{\nu} - \boldsymbol{\omega} \times \mathbf{P}. \tag{5.11}$$

The velocity of this point has the scalar form

$$\begin{aligned} \dot{X} &= -\nu_x - \omega_y Z + \omega_z Y \\ \dot{Y} &= -\nu_y - \omega_z X + \omega_x Z \\ \dot{Z} &= -\nu_z - \omega_x Y + \omega_y X, \end{aligned} \tag{5.12}$$

which is substituted into Equation 5.10 to yield

$$\begin{aligned} \dot{x} &= -\frac{1}{Z}\nu_x + \frac{X}{Z^2}\nu_z + \frac{XY}{Z^2}\omega_x - \frac{(X^2+Z^2)}{Z^2}\omega_y + \frac{Y}{Z}\omega_z, \\ \dot{y} &= -\frac{1}{Z}\nu_y + \frac{Y}{Z^2}\nu_z + \frac{(Y^2+Z^2)}{Z^2}\omega_x - \frac{XY}{Z^2}\omega_y - \frac{X}{Z}\omega_z. \end{aligned} \tag{5.13}$$

Given $X = xZ$ and $Y = yZ$ (Equation 5.9), it follows that

$$\begin{aligned} \dot{x} &= -\frac{1}{Z}\nu_x + \frac{x}{Z}\nu_z + xy\omega_x - (1+x^2)\omega_y + y\omega_z, \\ \dot{y} &= -\frac{1}{Z}\nu_y + \frac{y}{Z}\nu_z + (1+y^2)\omega_x - xy\omega_y - x\omega_z, \end{aligned} \tag{5.14}$$

which has the matrix form

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}}_{\dot{\mathbf{p}}} = \underbrace{\begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & (1+y^2) & -xy & -x \end{bmatrix}}_{\mathbf{L}_p} \boldsymbol{\upsilon}. \tag{5.15}$$

The interaction matrix requires a rank $r \geqslant 6$ for 6-DOF control. Given k points, or $r = 2k$ visual features, the stacked interaction matrix is

$$\mathbf{L}_p = \begin{bmatrix} \mathbf{L}_{p_1} \\ \mathbf{L}_{p_2} \\ \vdots \\ \mathbf{L}_{p_k} \end{bmatrix}, \tag{5.16}$$

which is used for the velocity control law

$$\boldsymbol{\upsilon} = -\lambda \widehat{\mathbf{L}_p^+}(\mathbf{p} - \mathbf{p}^*), \tag{5.17}$$

where $\widehat{\mathbf{L}_p^+}$ is the pseudo-inverse approximation of the interaction matrix for point features. Approximations for $\widehat{\mathbf{L}_p^+}$ are covered in [19].

While IBVS is also called two-dimensional visual servoing, Equation 5.15 indicates that the depth Z of three-dimensional world points are required to solve the interaction matrix.[1] Hence, depth estimation is one of the problem areas of IBVS.

1 Other image features also require visual feature depth [30, 32–34].

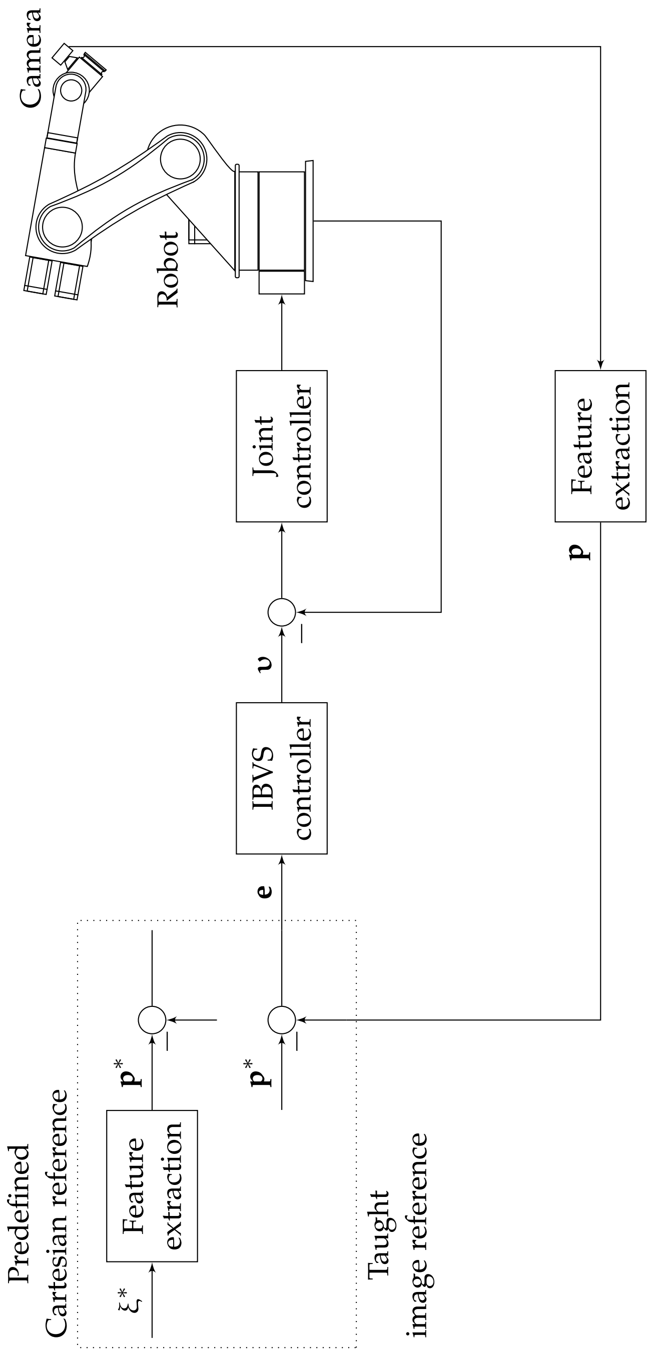


Figure 5.1: Image-based visual servoing

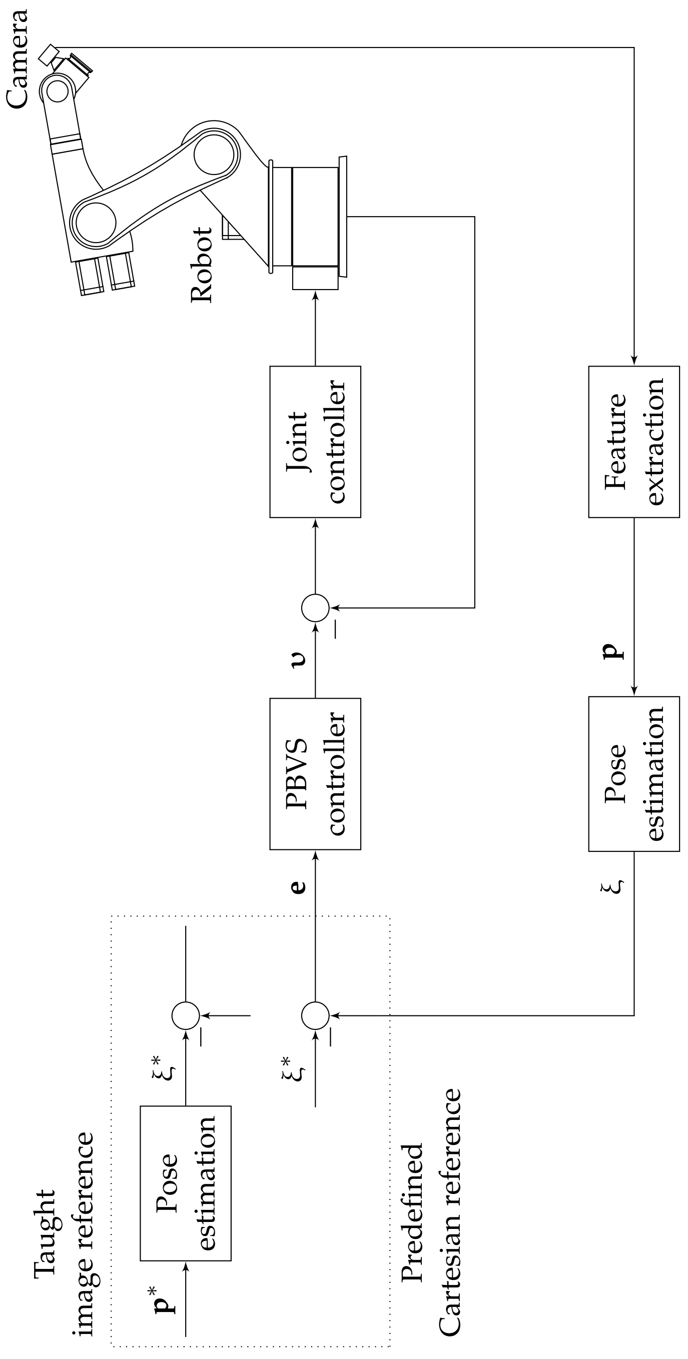


Figure 5.2: Position-based visual servoing

5.1.3 Position-Based Visual Servoing

The goal of an PBVS controller is to drive the camera pose from the current camera frame $\{C\}$ to the desired camera frame $\{C^*\}$. Like IBVS, PBVS requires feature extraction and matching (Figure 5.2). These extracted points are used for the pose estimation algorithm on a three-dimensional object model with coordinate frame $\{O\}$.

Given a three-dimensional object model with frame $\{O\}$ that contains k model points ${}^{o}\mathbf{P}_i = (X_i, Y_i, Z_i)$, $i \in [1, k] \cap \mathbb{Z}$, a known projection matrix $\mathbf{K}_p$, and an arbitrary pose matrix $\mathbf{T}(\mathbf{t}, \theta)$, the image projection of the model $\mathbf{p}_i^* = (x_i, y_i)$ is

$$\begin{aligned}\bar{\mathbf{p}}_i^* &= \mathbf{K}_p \mathbf{T}\ {}^{o}\bar{\mathbf{P}}_i \\ &= \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}.\end{aligned} \tag{5.18}$$

The unknown pose parameters $\mathbf{t}$ and θ can be estimated using an iterative pose estimation method [81] that minimises the squared reprojection error

$$\min_{\mathbf{t},\theta} \sum_{i=1}^{k} \left| \bar{\mathbf{p}}_i - \mathbf{K}\mathbf{T}\ {}^{o}\bar{\mathbf{P}}_i \right|^2, \tag{5.19}$$

where $\mathbf{p}_i$ is a set of extracted image points from a captured image corresponding to model points ${}^{o}\mathbf{P}_i$. The pose estimation will produce ${}^{c}_{o}\mathbf{R}$ and ${}^{c}_{o}\mathbf{t}$, which is required by the PBVS controller.

Following from the task function, $\mathbf{s} = ({}^{c}_{o}\mathbf{t}, \theta\mathbf{u}) \sim \xi$ and $\mathbf{s}^* = ({}^{c^*}_{o}\mathbf{t}, 0) \sim \xi^*$ [19]. Hence, the task function error is

$$\mathbf{e} = \left({}^{c}_{o}\mathbf{t} - {}^{c^*}_{o}\mathbf{t}, \theta\mathbf{u} \right), \tag{5.20}$$

where $\theta\mathbf{u}$ is the Euler axis/angle representation of $\mathbf{R} = {}^{c^*}_{c}\mathbf{R}$, that is, θ is some rotation about an arbitrary vector $\mathbf{u} = (u_1, u_2, u_3)$. The Euler axis/angle parameters are

$$\theta = \arccos\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2} \right), \tag{5.21}$$

and

$$\mathbf{u} = \frac{1}{2\sin\theta} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}, \tag{5.22}$$

where r_{ij} are the elements of the rotation matrix

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

and $\mathrm{tr}(\mathbf{R}) = (r_{11}, r_{22}, r_{33})$ is its trace. The interaction matrix for the PBVS controller is given by [19]

$$\mathbf{L}_{\xi} = \begin{bmatrix} -\mathbf{I}_3 & [{}^{c}_{o}\mathbf{t}]_{\times} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}_{6\times 6}, \tag{5.23}$$

where $\mathbf{L}_{\theta\mathbf{u}}$ is [46]

$$\mathbf{L}_{\theta\mathbf{u}} = \mathbf{I}_3 - \frac{\theta}{2}[\mathbf{u}]_{\times} + \left(1 - \frac{\operatorname{sinc}\theta}{\operatorname{sinc}^2\frac{\theta}{2}}\right)[\mathbf{u}]^2_{\times}, \tag{5.24}$$

and $[{}^{c}_{o}\mathbf{t}]_{\times}$ and $[\mathbf{u}]_{\times}$ are the cross product matrices

$$[{}^{c}_{o}\mathbf{t}]_{\times} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix},$$

and

$$[\mathbf{u}]_{\times} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}.$$

Then, the velocity control law is [19]

$$\boldsymbol{\upsilon} = \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} -\lambda\left(\left({}^{c^*}_{o}\mathbf{t} - {}^{c}_{o}\mathbf{t}\right) + [{}^{c}_{o}\mathbf{t}]_{\times}\,\theta\mathbf{u}\right) \\ -\lambda\theta\mathbf{u} \end{bmatrix}. \tag{5.25}$$

Similar to IBVS, a minimum of three non-collinear feature points are required for the pose estimation [27].

5.2 Kinect Integration

No special considerations are imposed by the Kinect RGB camera, outside of its intrinsic parameters. Additional visual servoing considerations are a result of the depth camera.

5.2.1 Depth Measurements

Regarding PBVS, depth camera measurements can be used directly for pose estimation. For example, instead of minimising the squared reprojection error (Equation 5.19) to find some unknown pose matrix $\mathbf{T}(\mathbf{t}, \boldsymbol{\theta})$, a matching algorithm such as Iterative Closest Point (ICP) can be applied to minimise the distance between a captured object model $\mathbf{P}_i$ and its reference model ${}^{o}\mathbf{P}_i$

$$\min_{\mathbf{t},\boldsymbol{\theta}} \sum_{i=1}^{k} |\mathbf{T}\mathbf{P}_i - {}^{o}\mathbf{P}_i|^2 . \tag{5.26}$$

Depending on the reference model and matching algorithm used, computational overhead can be saved on feature extraction and matching. The required matching algorithm robustness depends on the typical initial camera displacement of a visual servoing application.

If no reference model is readily available, teach-by-showing can be applied to generate a model by segmenting the object from the image and providing a feature descriptor.[2]

Regarding IBVS, point feature depth can be used directly in the point feature interaction matrix (Equation 5.15), by using the undistorted depth from Equation 4.3,

$$\mathbf{L}_p = \begin{bmatrix} -\frac{1}{z_k} & 0 & \frac{x}{z_k} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{z_k} & \frac{y}{z_k} & (1+y^2) & -xy & -x \end{bmatrix} . \tag{5.27}$$

Since feature matching is done before this stage of the image processing pipeline, depth values corresponding to the image point features can be retrieved from registered depth map. The depth estimation that was previously required becomes unnecessary, reducing computational overhead.

5.2.2 Depth Resolution

Within the Kinect's practical depth range of $0.8\,\mathrm{m}$ to $3.5\,\mathrm{m}$ (Table 4.1), the Kinect's depth resolution is $1.8\,\mathrm{mm}$ to $35\,\mathrm{mm}$. Uncertainty propagation from the three point estimate (Figure 4.15) adds negligible uncertainty on top of the quantisation error. This amount of depth uncertainty is insufficient to cause concern over IBVS stability [4].

Small point position errors will significantly impact PBVS task accuracy [19]. With the Kinect, the depth resolution improves as the camera approaches the goal. If the task accuracy is insufficient, teach-by-showing should be used to generate the reference object model and offset any remaining systematic calibration errors.

2 Object models are called point clouds in computer vision literature.

5.2.3 IR Light Limitations

It is well-established that the Kinect depth measurements are easily saturated in outdoor use [77], due to the IR radiation from sunlight. Similarly, IR radiation from heated surfaces also increases depth measurement uncertainty [72]. Since IR radiation is a common phenomenon, visual servo applications require controlled light environments to make Kinect depth measurements reliable.

Objects, such as transparent plastic or mirrors, which refract or absorb IR light will leave depth holes in depth maps, that is, depth map pixels that the Kinect fails to estimate and are reported as zero depth [66]. Furthermore, different viewing angles on objects with specular surfaces, or specular reflection, will also form depth holes. In these cases, much of the object recognition will depend on the RGB image, making the benefits of a having a depth camera insignificant.

5.2.4 Depth Holes

If depth holes form over visual features during a task, the task may fail when there are not enough visual depth features for PBVS pose estimation or the IBVS point feature interaction matrix. The usual visual servoing approaches (without the depth camera) will need to be active to prevent task failure. Alternatively, depth map preprocessing techniques [55] can be applied to fill the depth holes in depth maps before they are passed down the image processing pipeline.

5.2.5 Out-of-Bounds

The near range, far range, and field of view of the depth camera describes its practical range in three-dimensions, which can be imagined as a trapezoidal prism attached to its front. Depth points outside of this prism are considered out-of-bounds. Naturally, this prism represents the visibility constraint of the visual servoing task.

In addition to the field of view visibility constraint, depth values outside of the practical depth range are zero depth, as with depth holes [66]. Similar to one solution for depth holes, methods without the depth camera must be active when visual depth features disappear.

This is also a particular stability issue for methods that use depth map-based visual features. For example, Teulière and Marchand [13] state that their method fails when looking down corridors. Their interaction matrix becomes degenerate from the far range limitation when enough depth map pixels report zero depth. Problems with the near range are not mentioned by them, but it is an issue near task convergence, as depth map pixels will report zero depth.

A solution to the far range limitation is to lock depth values after they exceed a threshold, to prevent zero depth. Pauwels et al. [12] overcome the near range limitation problem by using depth measurements to estimate the depth of rendered models and instead use model depth for the controller. Their approach however, requires a Graphics Processing Unit (GPU) for real-time performance.

5.2.6 Sensor Bandwidth

Suppose the only known bandwidth about the visual servo system is the Kinect. It is reasonable to assume that the Kinect, or camera, is the limiting bandwidth of the visual servo system and the underlying robot and robot compensator of the system is locally stable. Then, the most fundamental conservation law of feedback can be described in terms of Bode's integral [95]

$$\int_0^{\Omega_a} \ln|S(j\omega)|\,d\omega = \begin{cases} \delta_s & \text{stable loops} \\ \pi\sum_{p\in P}\mathrm{Re}(p) + \delta_s & \text{unstable loops} \end{cases} \tag{5.28}$$

where $S(j\omega)$ is the sensitivity function, Ω_a is the available bandwidth of the system, $\mathrm{Re}(p)$ is the real part of any unstable poles, and δ_s is a small error associated with the tail of the complete integral due to finite available bandwidth. The sensitivity function $S(j\omega)$ expresses the system's ability to reject resonant disturbance signals.

If the only constraint on a signal is its bandwidth, the Nyquist-Shannon sampling theorem requires that a perfectly band limited signal with bandwidth f_b can only be reconstructed perfectly with a sampling rate greater than $2f_b$. Given that the sampling rates for RGB-full and RGB-standard are 10 fps and 30 fps (Table 4.1), respectively, their available bandwidths are, at most, 10π rad/s and 30π rad/s.

Stein [95] described a desirable sensitivity function in Figure 5.3. Bode's integral law for this sensitivity function is well approximated by

$$\int_0^{\Omega_c} \ln\left[\frac{\omega S_{\min}}{\Omega_c}\right] d\omega + (\Omega_a - \Omega_c)\ln(S_{\min}) = \pi\sum_{p\in P}\mathrm{Re}(p) \tag{5.29}$$

such that

$$S_{\min} = \exp\left[\frac{\pi\sum_{p\in P}\mathrm{Re}(p) + \Omega_c}{\Omega_a}\right] \tag{5.30}$$

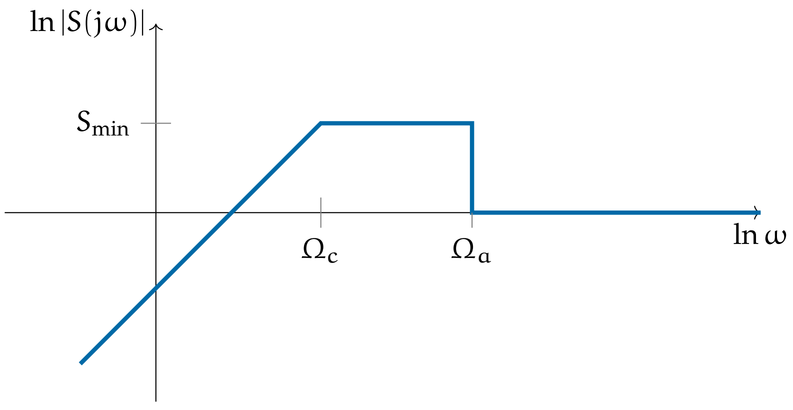


Figure 5.3: A desirable sensitivity function.

where $S_{\min}$ is the smallest sensitivity penalty and Ω_c is the desired corner frequency.

For a given problem domain, Equation 5.30 contains the fundamental limitation on disturbance rejection for any visual servo application using the Kinect. Ω_c approximates the bandwidth of the disturbance rejection capabilities of the system, and $S_{\min}$ is the maximum amplification of a resonant disturbance signal. The unstable poles are a characteristic of the underlying system and Ω_a is a function of the mode of the Kinect.

After the corner frequency Ω_c is chosen and unstable poles P are determined, the only way to reduce $S_{\min}$ is to increase Ω_a, that is, by reducing the RGB image quality. Hence, there is a fundamental trade-off between image quality and closed-loop disturbance rejection.

5.3 Summary

Practical considerations of using the Kinect for visual servoing applications are presented. The Kinect reduces computational overhead on image processing stages, such as pose estimation and depth estimation. Because direct depth measurements are available via the depth camera, two-dimensional image processing methods can be generalised to three-dimensions and depth estimation for the interaction matrices are unnecessary. Its IR light properties suggests that the Kinect should not be used outdoors and objects belonging to classes of non-IR-friendly surfaces should be avoided. Some problems of task stability are due to visual features disappearing into depth holes or when out-of-bounds. Depth holes can be filled by adding a depth map preprocessing stage before the before feature extraction. Alternatively, the usual visual servoing approaches, without using the depth camera, should be activated near task failure, which is also one solution to the practical depth range limitation.

6 CONCLUSION

So much universe, and so little time.

— Sir Terry Pratchett[1]

A through characterisation, modelling, and calibrated parameters of the Kinect cameras has been presented and used to discuss its implications for visual servo control applications. The research questions can now be addressed.

How are the Kinect RGB and depth cameras modelled? The Kinect RGB camera is modelled using the pinhole camera model. The Kinect depth camera is modelled using the usual Kinect disparity to depth model found in literature, which is then augmented with spatial distortion coefficients by Herrera [15] to correct for depth distortion and improve depth accuracy. A joint nonlinear calibration procedure using the KCT is conducted to solve for the intrinsic parameters and relative pose between the Kinect cameras.

What practical considerations does the Kinect impose on visual servo control applications? The Kinect reduces computational overhead on image processing stages, such as pose estimation or depth estimation. Its limitations lie with its IR light, $1.8\,\mathrm{mm}$ to $35\,\mathrm{mm}$ quadratic depth resolution, and $0.8\,\mathrm{m}$ to $3.5\,\mathrm{m}$ practical depth range. IR light properties suggests that the Kinect should not be used outdoors, due to IR saturation, and objects belonging to classes of non-IR-friendly surfaces should be avoided, due to IR refraction, absorption, or specular reflection. Problems of task stability exist, due to depth holes and range limitations. These can be reduced by using depth hole filtering (depth map preprocessing) and activating classical visual servoing techniques when Kinect-based approaches are near task failure.

In conclusion, the Xbox 360 Kinect is a cheap depth sensor that reduces computational overhead on two-dimensional image processing methods which require depth information, but the IR content in its environment requires attention due to the sensitivity of its IR light projector and stability issues of depth holes and practical depth range need to be addressed.

1 *The Last Hero*, Discworld #27, 2001.

7 RECOMMENDATIONS FOR FUTURE RESEARCH

Depth map preprocessing offers improved robustness for Kinect-based visual servoing methods, but their computational costs have not been discussed. It will be useful to know if the computational costs of various preprocessing techniques do not exceed the savings over and above Kinect-based visual servoing approaches.

The same methodology can be applied to other types of RGB-D cameras if they are required for visual servoing, since the KCT can be extended with other camera set-ups, but hardware compatibility with existing software libraries is not guaranteed.

The Kinect 2 offers improved specifications, such as a 1080p RGB camera and wider depth camera field of view [96]. Since the Kinect 2 is not developed by PrimeSense, its depth sensing technology is likely to be different to the Xbox 360 Kinect.[1] Existing literature on the Kinect 2 is still lacking and open source driver support (`libfreenect2`) is still in its very early stages.[2] There is much potential for research on its characterisation for computer vision applications in general.

1 Apple Inc. acquired PrimeSense after the Microsoft Xbox 360 Kinect was launched. Apple Inc. now owns the patents to PrimeSense's depth sensing technology.
2 The first stable release is dated 6 Nov 2015: `https://openkinect.github.io/libfreenect2/`.

REFERENCES

[1] M. Brady, “Artificial Intelligence and Robotics,” Feb. 1984, AIM-756, Massachusetts Institute of Technology. [Online]. Available: http://publications.csail.mit.edu/ai/pubs_browse.shtml

[2] P. I. Corke and S. A. Hutchinson, “Real-Time Vision, Tracking and Control,” in *IEEE International Conference on Robotics and Automation (ICRA)*, no. April, San Francisco, CA, Apr. 2000, pp. 622–629.

[3] G. C. J. Devol, “Programmed Article Transfer,” Jun. 13 1961, uS Patent 2,988,237. [Online]. Available: http://www.google.com/patents/US2988237

[4] P. I. Corke, *Robotics, Vision and Control*, 2nd ed., ser. Springer Tracts in Advanced Robotics, vol. 73, B. Siciliano, O. Khatib, and F. Groen, Eds. Springer-Verlag Berlin Heidelberg, 2011.

[5] M. T. Mason, “Creation Myths: The Beginnings of Robotics Research,” *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 72–77, Jun. 2012.

[6] G. J. Agin, “Real time control of a robot with a mobile camera,” Feb. 1979, Technical Note 179, SRI International. [Online]. Available: http://www.ai.sri.com/pub_list/technotes.php

[7] I. Siradjuddin, L. Behera, T. M. McGinnity, and S. Coleman, “A position based visual tracking system for a 7 DOF robot manipulator using a Kinect camera,” in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, Brisbane, QLD, Jun. 2012, pp. 1–7.

[8] M. Bdiwi and J. Suchý, “Handing-over Model-Free Objects to Human Hand with the Help of Vision / Force Robot Control,” in *10th International Multi-Conference on Systems, Signals & Devices (SSD)*, Hammamet, Mar. 2013, pp. 1–6.

[9] H. Cheng, H. Chen, and Y. Liu, “Object Handling Using Autonomous Industrial Mobile Manipulator,” in *IEEE 3rd Annual International Conference on Cyber Technology in Automation, Control and Intelligent Systems (CYBER)*, Nanjing, May 2013, pp. 36–41.

[10] M. Jokesch, M. Bdiwi, and J. Suchý, “Integration of vision/force robot control for transporting different shaped/colored objects from moving circular conveyor,” in *IEEE International Symposium*

on Robotic and Sensors Environments (ROSE), Timisoara, Oct. 2014, pp. 78–82.

[11] D. Nakhaeinia, P. Payeur, and R. Laganiere, “Adaptive Robotic Contour Following from Low Accuracy RGB-D Surface Profiling and Visual Servoing,” in *Canadian Conference on Computer and Robot Vision (CRV)*, Montreal, QC, May 2014, pp. 48–55.

[12] K. Pauwels, V. Ivan, E. Ros, and S. Vijayakumar, “Real-time Object Pose Recognition and Tracking with an Imprecisely Calibrated Moving RGB-D Camera,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, Sep. 2014, pp. 2733–2740.

[13] C. Teulière and E. Marchand, “A Dense and Direct Approach to Visual Servoing Using Depth Maps,” *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1242–1249, Jun. 2014.

[14] A. Hojaij, J. Zelek, and D. Asmar, “A Two Phase RGB-D Visual Servoing Controller,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, Sep. 2014, pp. 785–790.

[15] D. C. Herrera, “Kinect Calibration Toolbox,” 2011, accessed: 2014-03-31. [Online]. Available: http://www.ee.oulu.fi/~dherrera/kinect/

[16] Y. Shirai and H. Inoue, “Guiding a Robot by Visual Feedback in Assembling Tasks,” *Pattern Recognition*, vol. 5, no. 2, pp. 99–108, Jun. 1973.

[17] G. J. Agin, “Servoing with visual feedback,” Jul. 1977, Technical Note 149, SRI International. [Online]. Available: http://www.ai.sri.com/pub_list/technotes.php

[18] S. A. Hutchinson, G. D. Hager, and P. I. Corke, “A Tutorial on Visual Servo Control,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[19] F. Chaumette and S. A. Hutchinson, “Visual Servo Control Part I: Basic Approaches,” *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, Dec. 2006.

[20] ——, “Visual Servo Control Part II: Advanced Approaches,” *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, Mar. 2007.

[21] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, “Dynamic sensor-based control of robots with visual feedback,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 404–417, Oct. 1987.

[22] E. Malis, "Survey of vision-based robot control," INRIA, Sophia Antipolis, Nice, Tech. Rep., 2002.

[23] F. Janabi-sharifi, L. Deng, and W. J. Wilson, "Comparison of Basic Visual Servoing Methods," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 5, pp. 967–983, 2011.

[24] L. Deng, W. J. Wilson, and F. Janabi-Sharifi, "Characteristics of Robot Visual Servoing Methods and Target Model Estimation," in *IEEE International Symposium on Intelligent Control (ISIC)*, Vancouver, Oct. 2002, pp. 684–689.

[25] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach," in *IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, Mar. 1985, pp. 662–668.

[26] D. B. Westmore and W. J. Wilson, "Direct Dynamic Control of a Robot Using an End-Point Mounted Camera and Kalman Filter Position Estimation," in *IEEE International Conference on Robotics and Automation (ICRA)*, Sacramento, CA, Apr. 1991, pp. 2376–2384.

[27] W. J. Wilson, C. C. W. Hulls, and G. S. Bell, "Relative End-Effector Control Using Cartesian Position Based Visual Servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 684–696, Oct. 1996.

[28] A. C. Sanderson and L. E. Weiss, "Image-based visual servo control of robots," in *Proc. SPIE 0360, Robotics and Industrial Inspection*, 1983, pp. 164–169.

[29] J. Feddema and O. R. Mitchell, "Vision-guided servoing with feature-based trajectory generation," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 5, pp. 691–700, Oct 1989.

[30] P. Rives, F. Chaumette, and B. Espiau, "Visual Servoing Based on a Task Function Approach," in *International Symposium on Experimental Robotics*, Montreal, Jun. 1989, pp. 412–428.

[31] F. Chaumette, P. Rives, and B. Espiau, "Positioning of a Robot with respect to an Object, Tracking it and Estimating its Velocity by Visual Servoing," in *IEEE International Conference on Robotics and Automation (ICRA)*, Sacramento, CA, Apr. 1991, pp. 2248–2253.

[32] B. Espiau, F. Chaumette, and P. Rives, "A New Approach to Visual Servoing in Robotics," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, Jun. 1992.

[33] F. Chaumette, "Image Moments: A General and Useful Set of Features for Visual Servoing," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 713–723, Aug. 2004.

[34] O. Tahri and F. Chaumette, "Point-Based and Region-Based Image Moments for Visual Servoing of Planar Objects," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1116–1127, Dec. 2005.

[35] N. P. Papanikolopoulos and P. K. Khosla, "Adaptive Robotic Visual Tracking: Theory and Experiments," *IEEE Transactions on Automatic Control*, vol. 38, no. 3, pp. 429–445, Mar. 1993.

[36] K. Hosoda and M. Asada, "Versatile Visual Servoing without Knowledge of True Jacobian," in *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, Munich, Sep. 1994, pp. 186–193.

[37] Y. H. Liu, H. Wang, C. Wang, and K. K. Lam, "Uncalibrated Visual Servoing of Robots Using a Depth-Independent Interaction Matrix," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 804–817, Aug. 2006.

[38] K. Hashimoto, T. Kimoto, T. Ebine, and H. Kimura, "Manipulator Control with Image-Based Visual Servo," in *IEEE International Conference on Robotics and Automation (ICRA)*, no. April, Sacramento, CA, Apr. 1991, pp. 2267–2272.

[39] B. Espiau, "Effect of Camera Calibration Errors on Visual Servoing in Robotics," in *3rd International Symposium on Experimental Robotics*, Kyoto, Japan, Oct. 1993, pp. 182–192.

[40] K. Miura, K. Hashimoto, H. Inooka, J. A. Gangloff, and M. F. de Mathelin, "Model-less visual servoing using modified simplex optimization," *Artificial Life and Robotics*, vol. 10, no. 2, pp. 131–135, Nov. 2006.

[41] E. Malis and P. Rives, "Robustness of Image-Based Visual Servoing with Respect to Depth Distribution Errors," in *IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Sep. 2003, pp. 1056–1061.

[42] E. Malis, Y. Mezouar, and P. Rives, "Robustness of Image-Based Visual Servoing With a Calibrated Camera in the Presence of Uncertainties in the Three-Dimensional Structure," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 112–120, Feb. 2010.

[43] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*, ser. Lecture Notes in Control and Information Sciences, D. J. Kriegman, G. D. Hager, and A. S. Morse, Eds. Springer London, 1998, ch. 5, pp. 66–78.

[44] P. I. Corke and S. A. Hutchinson, "A New Partitioned Approach to Image-Based Visual Servo Control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 507–515, Aug. 2001.

[45] K. Deguchi, "Optimal Motion Control for Image-Based Visual Servoing by Decoupling Translation and Rotation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, no. October, Victoria, BC, Oct. 1998, pp. 705–711.

[46] E. Malis, F. Chaumette, and S. Boudet, "2-1/2-D Visual Servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, Apr. 1999.

[47] N. R. Gans, S. A. Hutchinson, and P. I. Corke, "Performance Tests for Visual Servo Control Systems, with Application to Partitioned Approaches to Visual Servo Control," *The International Journal of Robotics Research*, vol. 22, no. 10-11, pp. 955–981, Oct. 2003.

[48] E. Malis, "Hybrid vision-based robot control robust to large calibration errors on both intrinsic and extrinsic camera parameters," in *European Control Conference*, Porto, Sep. 2001, pp. 2898–2903.

[49] N. R. Gans and S. A. Hutchinson, "Stable Visual Servoing Through Hybrid Switched-System Control," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 530–540, Jun. 2007.

[50] G. Silveira, "On Intensity-Based Nonmetric Visual Servoing," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 1019–1026, jul 2014.

[51] C. Collewet and E. Marchand, "Photometric Visual Servoing," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 828–834, Aug. 2011.

[52] A. Dame and E. Marchand, "Mutual Information-Based Visual Servoing," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 958–969, oct 2011.

[53] Q. Bateux and E. Marchand, "Direct visual servoing based on multiple intensity histograms," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, may 2015, pp. 6019–6024.

[54] Z. Zhang, "Microsoft Kinect Sensor and Its Effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, 2012.

[55] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced Computer Vision with Microsoft Kinect Sensor: A Review," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013.

[56] P. Bouffard, J. Gillula, H. Huang, M. Vitus, and C. Tomlin, “Quadrotor Altitude Control and Obstacle Avoidance,” 2011, accessed: 2014-01-23. [Online]. Available: http://wiki.ros.org/openni/Contests/ROS%203D/Quadrotor%20Altitude%20Control%20and%20Obstacle%20Avoidance

[57] Open Source Robotics Foundation, Inc., “TurtleBot,” 2011, accessed: 2015-07-10. [Online]. Available: http://www.turtlebot.com/

[58] Jet Propulsion Laboratory, “Future Robotic Interfaces,” 2014, accessed: 2015-07-10. [Online]. Available: http://www.hi.jpl.nasa.gov/projects/#future-robotic-interfaces

[59] J. St. Jean, *Kinect Hacks*, 1st ed., S. Wallace, Ed. O’ Reilly Media, Inc., 2013.

[60] D. C. Herrera, J. Kannala, and J. Heikkilä, “Joint Depth and Color Camera Calibration with Distortion Correction,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 2058–2064, 2012.

[61] J. Smisek, M. Jancosek, and T. Pajdla, “3D with Kinect,” in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov. 2011, pp. 1154–1160.

[62] K. Khoshelham and S. O. Elberink, “Accuracy and resolution of Kinect depth data for indoor mapping applications.” *Sensors (Basel, Switzerland)*, vol. 12, no. 2, pp. 1437–54, Jan. 2012.

[63] C. Raposo, J. a. P. Barreto, and U. Nunes, “Fast and Accurate Calibration of a Kinect Sensor,” in *International Conference on 3D Vision*, Seattle, WA, Jun. 2013, pp. 342–349.

[64] J. Smisek, “3D Camera Calibration,” Master’s thesis, Czech Technical University, 2011, Center for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering.

[65] R. Macknojia, A. Chavez-Aragon, P. Payeur, and R. Laganiere, “Experimental characterization of two generations of Kinect’s depth sensors,” in *IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, Magdeburg, Nov. 2012, pp. 150–155.

[66] T. Mallick, P. P. Das, and A. K. Majumdar, “Characterizations of Noise in Kinect Depth Images: A Review,” *IEEE Sensors Journal*, vol. 14, no. 6, pp. 1731–1740, Jun. 2014.

[67] J. C. K. Chow and D. D. Lichti, “Photogrammetric Bundle Adjustment With Self-Calibration of the PrimeSense 3D Camera Technology: Microsoft Kinect,” *IEEE Access*, vol. 1, pp. 465–474, 2013.

[68] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns," May 13 2010, uS Patent 2010/0118123 A1. [Online]. Available: https://www.google.com/patents/US20100118123

[69] M. Martinez and R. Stiefelhagen, "Kinect Unleashed : Getting Control over High Resolution Depth Maps," in *13th IAPR International Conference on Machine Vision Applications (MVA)*, Kyoto, Japan, May 2013, pp. 247–250.

[70] K. Khoshelham, "Accuracy Analysis of Kinect Depth Data," in *International Society for Photogrammetry and Remote Sensing (ISPRS) Workshop*, Calgary, Aug. 2011, pp. 29–31.

[71] D. C. Herrera, J. Kannala, and J. Heikkilä, "Accurate and Practical Calibration of a Depth and Color Camera Pair," in *Computer Analysis of Images and Patterns*, ser. Lecture Notes in Computer Science, P. Real, D. Diaz-Pernil, H. Molina-Abril, A. Berciano, and W. Kropatsch, Eds. Springer Berlin Heidelberg, 2011, vol. 6855, pp. 437–445.

[72] H. Haggag, M. Hossny, D. Filippidis, D. Creighton, S. Nahavandi, and V. Puri, "Measuring Depth Accuracy in RGBD Cameras," in *7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Carrara, VIC, Dec. 2013, pp. 1–7.

[73] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[74] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab," 2013, accessed: 2014-10-21. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[75] C. Zhang and Z. Zhang, "Calibration between depth and color sensors for commodity depth cameras," in *IEEE International Conference on Multimedia and Expo (ICME)*, Barcelona, Jul. 2011, pp. 1–6.

[76] H. Yamazoe, H. Habe, I. Mitsugami, and Y. Yagi, "Easy Depth Sensor Calibration," in *21st International Conference on Pattern Recognition*, Tsukuba, Nov. 2012, pp. 465–468.

[77] J. Suarez and R. R. Murphy, "Using the Kinect for Search and Rescue Robotics," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, College Station, TX, Nov. 2012, pp. 1–2.

[78] M. Van den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. Van Gool, and M. Buss,

"Real-time 3D Hand Gesture Interaction with a Robot for Understanding Directions from Humans," in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Atlanta, GA, Jul. 2011, pp. 357–362.

[79] D. Fiedler and M. Heinrich, "Impact of Thermal and Environmental Conditions on the Kinect Sensor," in *International Workshop on Depth Image Analysis*, Tsukuba, Japan, Nov. 2012, pp. 1–8.

[80] K. Berger, "A State of the Art Report on Multiple RGB-D Sensor Research and on Publicly Available RGB-D Datasets," in *Computer Vision and Machine Learning with RGB-D Sensors*, ser. Advances in Computer Vision and Pattern Recognition, L. Shao, J. Han, P. Kohli, and Z. Zhang, Eds. Springer International Publishing, 2014, ch. 2, pp. 27–45.

[81] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

[82] G. Barbastathis, C. Sheppard, and S. Oh, "2.71 Optics, Spring 2009, (Massachusetts Institute of Technology: MIT OpenCourseWare)," accessed: 2015-01-06. [Online]. Available: http://ocw.mit.edu

[83] J. Heikkilä and O. Silvén, "A Four-step Camera Calibration Procedure with Implicit Image Correction," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Juan, Jun. 1997, pp. 1106–1112.

[84] J. Weng, P. Cohen, and M. Herniou, "Camera Calibration with Distortion Models and Accuracy Evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, Aug. 1992.

[85] D. C. Brown, "Decentering Distortion of Lenses," *Photogrammetric Engineering*, vol. 32, no. 3, pp. 444–462, 1966.

[86] J. P. de Villiers, F. W. Leuschner, and R. Geldenhuys, "Centi-pixel accurate real-time inverse distortion correction," in *Proceedings of SPIE Optomechatronic Technologies*, vol. 7266, San Diego, CA, Nov. 2008, pp. 726 611–1—-726 611–8.

[87] J. P. de Villiers, "Correction of radially asymmetric lens distortion with a closed form solution and inverse function," M.Sc. Dissertation, University of Pretoria, 2007, Faculty of Engineering, the Built Environment & Information Technology.

[88] T. York, "Fundamentals of Image Sensor Performance," April 2011, accessed: 2015-02-26. [Online]. Available: http://www.cse.wustl.edu/~jain/cse567-11/ftp/imgsens/index.html

[89] A. R. Smith, "A Pixel is Not a Little Square, a Pixel is Not a Little Square, a Pixel is Not a Little Square! (And a Voxel is Not a Little Cube)," Microsoft, Tech. Rep., Jul. 1995.

[90] Microsoft, "Kinect for Windows Sensor Components and Specifications," accessed: 2013-08-01. [Online]. Available: http://msdn.microsoft.com/en-us/library/jj131033.aspx

[91] OpenKinect, "Protocol Documentation," 2011, accessed: 2015-03-12. [Online]. Available: http://openkinect.org/wiki/Protocol_Documentation

[92] ——, "Imaging Information," 2011, accessed: 2015-03-12. [Online]. Available: http://openkinect.org/wiki/Imaging_Information

[93] M. Viager, "Analysis of Kinect for mobile robots," M.Sc. Course Report, Technical University of Denmark, 2011.

[94] J. Heikkilä, "Geometric Camera Calibration Using Circular Control Points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1066–1077, 2000.

[95] G. Stein, "Respect the Unstable," *IEEE Control Systems*, vol. 23, no. 4, pp. 12–25, Aug 2003.

[96] Microsoft Corporation, "Kinect hardware," 2014, accessed: 2016-02-21. [Online]. Available: https://dev.windows.com/en-us/kinect/hardware