University of the West of England

Autonomous Model Building Using Vision and Manipulation

A thesis submitted in partial fulfilment of the requirements of the University of the West of England, Bristol for the degree of Doctor of Philosophy

Author:
Alan BROUN

Supervisors:
Prof. Tony PIPE
Prof. Majid MIRMEHDI
Prof. Chris MELHUISH

January 3, 2016
Word Count: 37,985

Abstract

Robotic systems often require models in order to successfully control themselves, and to interact with the world. Models take many forms and include kinematic models to plan motions, dynamics models to control forces, and models of 3D geometry to check for collisions, to name but a few. Traditionally, models are provided to the robotic system by the designers that build the system. However for long-term autonomy, it becomes important for the robot to be able to build and maintain models of itself, and of objects it might encounter.

This thesis advances and explores the argument for enabling robotic systems to autonomously build such models, and the main contribution of this research is to show how a layered approach can be taken to building these models. To begin with a system is presented which enables a robot starting with a limited amount of information to first build a Surfel based model of its hand using the Iterative Closest Point (ICP) algorithm. The system is then able to use the hand model and the ICP algorithm to track the movement of the hand in order to autonomously construct a kinematic model which describes the robot's body. This kinematic model can in turn be used to plan and perform future movements.

Key to the incremental, autonomous approach is the use of exploratory actions. These are actions that the robot can perform in order to gain information, either about itself, or about an object with which it is interacting. Another contribution is the presentation of a method whereby a robot, after being powered on, can use exploratory actions to home its joints using just vision, i.e. traditional methods such as absolute encoders, or limit switches are not required.

The next contribution of this research is to look beyond the robot's body and to present methods with which a robot can autonomously build models of objects in the world around it. The first class of objects examined are flat pack cardboard boxes, a class of articulated objects with a number of interesting properties. It is shown how exploratory actions can be used to build an edge based model of a flat pack cardboard box and to locate any hinges the box may have. Specifically, it is shown how when interacting with an object, a robot can combine haptic feedback from force sensors, with visual feedback from an edge based tracking system to get more information from an object than would be possible using just a single sensor modality.

The final contribution of this research is to present a series of exploratory actions for a robotic text reading system that allow text to be found and read from an object. The text reading system highlights how models of objects can take many forms, from a representation of their physical extents, to the text that is written on them.

Acknowledgements

The journey involved in studying for a PhD and in writing up this thesis has been long and arduous, and I could not have made this journey without the help of many different people. Firstly, thank you to my supervisor Tony Pipe for helping me to find my thesis topic and for providing much useful advice along the way. The Leverhulme Trust, provided financial support for this work, and funded my PhD under project number F/00182/CI, and for this I am very grateful. Thank you also to my PhD partner Chris Beck, who has been a true friend over the years, and someone with whom I can commiserate with on the stress of PhD work, and on the woes of robotics research. We did not quite achieve our initial plan of completing a PhD in 3 weeks, but occasions such as trying to wrestle robots into submission at 3am in a German hotel room, have made the whole process very memorable, and great fun.

Thank you to Majid Mirmehdi for the many hours spent reading and suggesting corrections to my papers, they benefited immensely from your input. Thank you to Chris Melhuish, head of the BRL, for inspiration, keeping the lights on and also for the upgrade to the labs midway through my PhD, which made working conditions immeasurably more comfortable. Thank you to Tom Rooney, for diverting me with talk of whiskers and underwater robotics, and for showing me the vast number of uses for Gaffa tape. Thank you to Ian Horsfield and the rest of the technicians in the lab for all your help, and for your patience with me as I advanced from my previous life as a software engineer to becoming a bit more practical. Also, thank you to Matt Studley, who is always a great person to bounce ideas off and a continuing source of advice and inspiration.

I am also extremely grateful to my family, for their love, and for putting up with ridiculous things like a last minute proof reading session on a family holiday. Thank you in particular to my parents, whose constant support gave me the confidence to leave my job, and to embark on this PhD in the first place. Lastly, but in no way least, thank you to my darling wife Kinga, who let me drag her to Bristol so that I could spend my time playing with robots. Thank you for patiently putting up with countless late nights, and frequent bouts of grumpiness. I could not have done this without you.

Contents

List of Figures

List of Symbols

A wave amplitude.

Q a set of quadrilaterals.

Z a set of zoom levels.

$[\hat{\mathbf{s}}]_{\times}$ skew symmetric matrix.

Ψ set of exploratory poses.

α a homing offset angle.

β angle between Kinect camera axis and the instantaneous velocity vector at an observed point.

$\dot{\boldsymbol{\theta}}$ acceleration in joint space.

ψ exploratory pose.

$\boldsymbol{\theta}$ position in joint space, a vector of joint angles.

ε_{hand} threshold used when centring robot's hand.

$\eta(i)$ a surfel indicator function.

γ wrist actuator angle.

$\hat{\phi}$ visual homing measured exploratory angle.

λ_{depth} depth threshold used in Chapter 7.

λ_{hand} threshold used to detect a robot's hand.

λ_{model} model distance threshold used in Chapter 7.

λ_{ray} ray distance threshold used in Chapter 7.

$\mathbf{E}$ Euclidean transformation matrix.

$\mathbf{G}_i$ $SE(3)$ generator matrix.

$\mathbf{I}$ identity matrix.

$\mathbf{J}(\boldsymbol{\theta})$ joint Jacobian matrix.

$\mathbf{K}$ intrinsic camera matrix.

$\mathbf{M}$ interframe motion transformation matrix.

$\mathbf{R}$ rotation matrix.

$\mathbf{T}$ transformation matrix.

$\dot{\mathbf{x}}$ acceleration in Cartesian space.

$\hat{\mathbf{n}}(\mathfrak{e}_i)$ edge point normal.

$\hat{\mathbf{n}}(\mathfrak{p}_i)$ point normal.

$\hat{\mathbf{n}}(\mathfrak{s}_i)$ surfel normal.

$\hat{\mathbf{n}}_{image}$ Kinect image plane normal.

$\hat{\mathbf{s}}$ screw axis normal.

$\mathbf{b}$ the state of a flat pack cardboard box.

$\mathbf{o}$ a homing offset.

$\mathbf{s_o}$ point on a screw axis.

$\mathbf{v}(\mathfrak{e}_i)$ edge point position.

$\mathbf{v}(\mathfrak{p}_i)$ point position.

$\mathbf{v}(\mathfrak{s}_i)$ surfel position.

$\mathbf{x}$ position in Cartesian space.

$\mathfrak{E}$ a set of edge points.

$\mathfrak{F}$ a set of faces.

$\mathfrak{H}$ a set of hinges.

$\mathfrak{P}$ a 3D point cloud.

$\mathfrak{S}$ a surfel model.

$\mathfrak{e}_i$ an edge point.

$\mathfrak{f}_i$ a box face.

$\mathfrak{h}_i$ a box hinge.

$\mathfrak{p}_i$ a point from a point cloud.

$\mathfrak{s}_i$ a surfel.

$m(i)$ a surfel matching function.

$\mathscr{B}$ cardboard box model histogram.

$\mathscr{C}$ calibration index for a kinematic calibration method.

$\mathscr{E}$ cardboard box external colour histogram.

$\mathscr{H}$ a histogram.

$\mathscr{I}$ cardboard box internal colour histogram.

$\mathscr{M}$ mobility index.

$\mathscr{S}$ number of sensed joints.

$\mathtt{E}$ an edge image.

$\mathtt{I}$ image from a camera.

ω angular velocity.

ϕ visual homing exploratory angle.

π pi.

$\rho(X,Y)$ correlation coefficient between two random processes.

θ rotation angle.

φ_{model} surfel model update angle.

$c(\mathfrak{p}_i)$ point colour.

$c(\mathfrak{s}_i)$ surfel colour.

d_s displacement along a screw axis.

d_{max} distance threshold used when building surfel models.

f wave frequency.

k constant gain.

l focal length.

q a quadrilateral.

$r(\mathfrak{s}_i)$ surfel radius.

r distance of point from actuator axis.

t time.

v_{image} linear velocity on image plane.

v linear velocity.

z a zoom level.

Glossary

active computer vision refers to a computer vision system that can change the viewing parameters of the camera (such as its position, or zoom level for example) in order get more information from the observed scene.

Augmented Reality (AR) marker a computer generated marker, designed so that its position and orientation can be easily and reliably estimated using computer vision algorithms.

Bayesian network a probabilistic graphical model that uses a directed acyclic graph to represent random variables and their dependencies.

BERT Bristol and Elumotion Robotic Torso - an upper torso humanoid robot used in this work.

block matching algorithm an algorithm that estimates the movement of small blocks (typically 8x8 or 16x16 pixels) between consecutive video frames.

body schema a sensory and motor systems representation of the body, that is used for performing actions.

CAN Controller Area Network - a communication bus standard commonly used in automation and robotics.

closed world assumption a simplifying assumption made about some robotic systems that the internal models used by the robot are complete, and contain everything about the world that the robot needs to know.

CMM Coordinate Measuring Machine - a device for performing precision 3D measurements.

EPOS motor controller a brand of brushless motor controllers made by Maxon, and used extensively on the BERT robot.

exploratory action an action taken by a robot to try to extract information about itself, its environment, or about an object it is interacting with.

IBVS Image Based Visual Servoing - a technique whereby visual feedback from a camera is used to control the motion of a robot.

ICP Iterative Closest Point - an algorithm for aligning point clouds.

internal model an abstract computer model describing some aspect of the robot, the environment the robot operates in, or an object in the environment. Used to allow the robot to reason about the world and to plan actions.

kinematic calibration the process of refining an existing kinematic model so that it better describes the movement of a robotic system.

kinematic identification the process of building a kinematic model for a robotic system.

kinematic model a model that mathematically describes a robot's motion without considering the forces that affect the motion.

lazy learning methods methods for machine learning that delay the task of generalising from their training data.

LWPR Locally Weighted Projection Regression - a lazy learning method that performs linear regression using training samples close to the input to predict an output.

motor babbling a series of random motor commands often used with machine learning to learn about a robotic system.

MSER Maximally Stable Extremal Regions - a technique for detecting blobs in images.

OCR Optical Character Recognition - a technique which can extract text from images.

optical flow a 2D vector field describing the apparent motion seen between camera image frames as the camera moves relative to the environment or objects in the environment.

PSO Particle Swarm Optimisation - an optimisation method which draws inspiration from the movement of natural flocks and swarms.

ROS Robot Operating System - an open source set of software libraries designed to aid in the writing of distributed robotic software.

SEA Series Elastic Actuator - a compliant actuator formed by inserting a spring element between the gearbox and the load.

SLAM Simultaneous Localization and Mapping - a technique whereby the location of a robot within a map is estimated, at the same time that the map is being constructed.

Smith predictor a predictive controller that makes use of an internal model of a controlled system, and a model of the delays within that system in order to control the system.

surfel short for Surface Element, a surfel is an oriented 3D point that can be used to describe complex geometric objects.

Chapter 1

Introduction

In modern robotic systems, models are often used to help control the robot, and to help the robot to interact with the world. For example, a robot may have kinematic and dynamic models of its body to help it plan motions, and 3D models of its body parts to check motion plans for potential collisions. The robot may also have models of its environment and the objects around it.

In a lot of systems, these models are provided by the system designers. This approach works well for fixed systems working in a known environment, but can face problems if the robot encounters unfamiliar objects, if its body degrades, or if its body is modified by a user.

This thesis looks at using active computer vision, and exploratory actions in order to enable a robot to progressively build a series of internal models. It starts by looking at how a robot can build models of itself, and then extends this work to show how a robot can build models of objects that it encounters in its environment. In particular, emphasis is placed on how interacting with an object can give a robot more information for building a model than it would be able to obtain by just passively observing the object.

1.1 Motivation

To put the problem which this thesis tries to tackle into some kind of context, consider the situation where a robot with one or more arms is switched on, and the goal is to try to get it to pick up and identify an object in front of it.

Looking at existing robotic systems which manipulate objects, the simplest possible system for manipulating objects is one where the object is in a fixed known position and the robot moves through a sequence of pre-recorded joint angles in order to pick the object up, and move it around. This approach is easy to implement, and is the approach used by many industrial robotics applications

where repetitive actions are performed on identical objects. It is also easy to see however that because of its inflexibility, it is not a suitable way to provide a robot with the ability to manipulate unfamiliar objects in arbitrary poses.

1.1.1 Model Based Robotics

One way to introduce more flexibility into the system is to use a model based approach (also called a sense-plan-act approach) to design the robotic system. Here the robot maintains one or more models of itself, its environment, and the object it is trying to manipulate. The robot uses sensors to try to determine as well as possible the current state of the world, and to try to define the world state in terms of the available models it has. With its estimate of the world state, the robot plans its actions and then attempts to execute them by moving its actuators. The models provide an advantage over simply using raw sensor data to plan actions, as they help to provide much more information about the state of the world.

Many robotic systems presented in the literature make use of models in their operation. Two examples which illustrate well the way in which internal models can help a robotic system interact with the world are the ARMAR-III robot (Asfour et al. 2006) and the work of Scholz et al. (2011). The ARMAR-III robot is a recent robotic system that has used pre-built 3D models of objects in order to aid its perception system in the task of identifying objects and estimating their position and orientation in 3D space. Scholz et al. (2011) equip a PR2 robot with a model of a small cart and thus enable the PR2 to navigate around an office environment whilst pushing the cart.

Often, the models that the robotic system uses to plan its actions are created for it by human operators, with models of the robot and the objects it manipulates, being derived from CAD models. A key problem with this approach detailed by Murphy (2000) is the *closed world assumption*. The closed world assumption assumes that the models are complete, and contain everything about the world that the robot needs to know. If the robot encounters a new object that it does not have a model for, or if one of its existing models turns out to be inaccurate, then the plans that the robot creates for manipulating objects may fail and cause damage to both the environment and the robot.

Steps to try to overcome this problem can involve improving the accuracy of the kinematic and possibly dynamic model of the robot, by using kinematic and dynamic identification procedures (Mooring et al. 1991). They can also involve giving the robot new and updated models of objects it is likely to encounter. Manually creating models can be time consuming however, so ideally the robot would be able to autonomously create and maintain models of objects of interest as and when it encounters them.

1.1.2 Behavioural Robotics

Before continuing down the path of building and maintaining models, it is important to be aware of a more radical approach to the problem of having incomplete or inaccurate models. Behavioural robotics as popularised by Brooks (1991), argues that maintaining models of the world and the robot is unnecessary, and that intelligent and useful behaviour can be obtained from a robot by grounding its actions in the real world. The principle is that ideally, actions should be directly linked to sensor inputs. The aim is to allow the robot to use the world itself as a model, and this is certainly appealing as the world is always up to date and correct.

This approach has been extended and applied with some success to humanoid robots built for the Cog (Brooks et al. 1999) and Domo (Edsinger 2007) projects. The work on the Domo robot in particular involved developing a hierarchical system of simple behaviours, that when combined together could be used to develop much more complex behaviours. Figure 1.1 shows the break-down of one of Domo's complex behaviours, the 'help with chores' task.

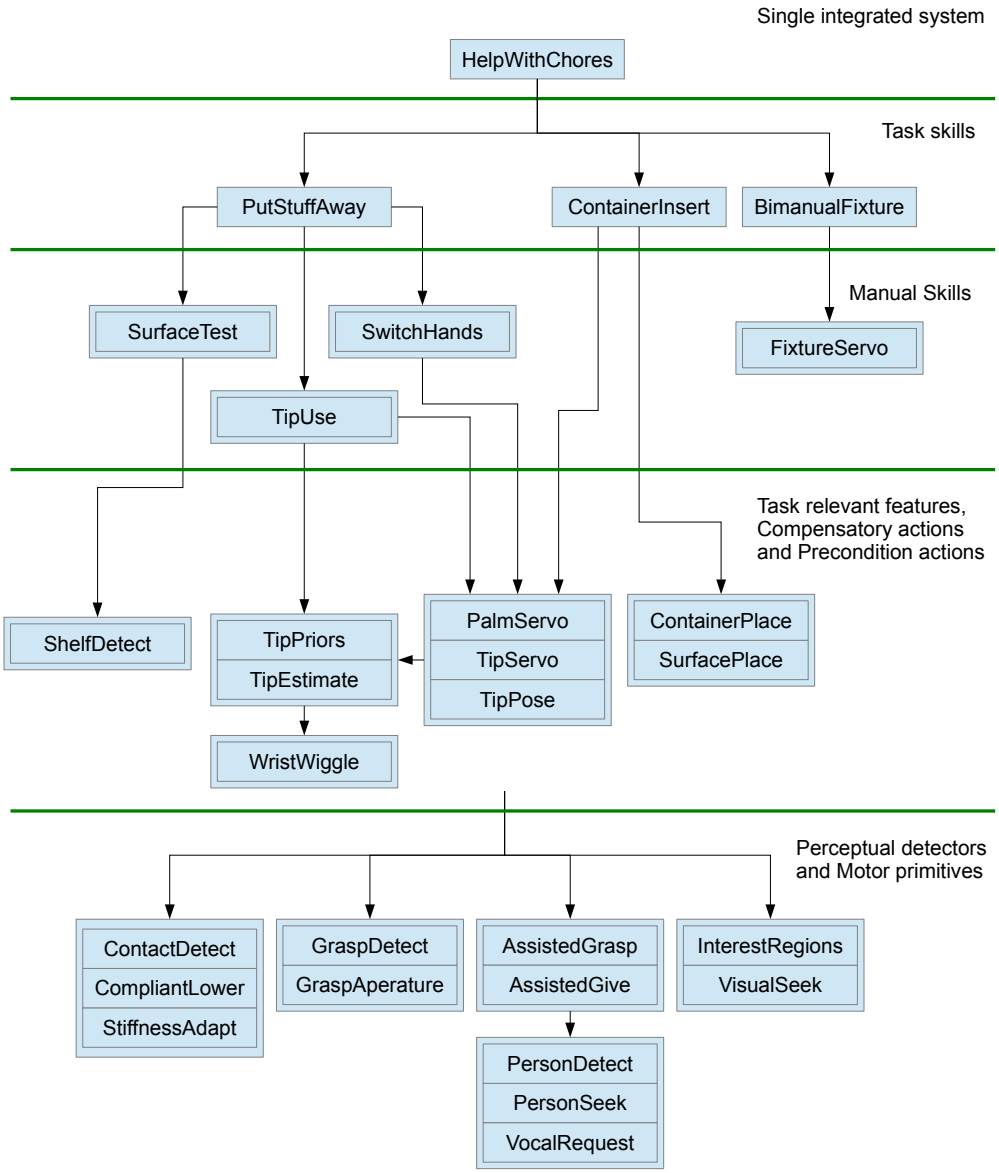


Figure 1.1: An example of how a complex task such as 'help with chores' was programmed for the Domo robot (Edsinger (2007)) by hierarchically composing simpler tasks.

The success of the approach demonstrates the power of decomposition, but looking at the implementation of some of the modules in Figure 1.1, it is hard to deny that models and representations are used in some of the modules, even if they are simple and coupled tightly to real objects in the world. The robot Domo has a representation of the 'tip' of the tool or object it is currently using, it has a representation of the position of a shelf it can put objects on, and a kinematic

model is used to provide predictions for where the robot's hands will appear in camera space. Therefore this thesis argues, that even if they are implicit in the implementation of a system, models are important to allow a robot to perform a variety of high level tasks. Furthermore this thesis also argues, that making models explicit offers certain advantages in that they become easier to reason about, and thus decouple from the rest of the robotic system.

1.1.3 Autonomous Model Building

Accepting that models are important for enabling robots to plan and perform a wide range of actions in unstructured environments, the focus of this research is therefore on getting a robot to automatically build models of both itself and of objects of interest. With these models the robot should be able to plan and execute actions that move its body, and then successfully manipulate the objects. The aim is to enable a robot to cope with the two problems of encountering unfamiliar objects, and of its existing models becoming too inaccurate to be useful.

It is interesting to consider both the order in which models should be constructed, and the order in which they *can* be constructed. Models as well as being seen as a representation of some aspect of the world, can also be seen as enabling artefacts which can improve a robot's performance in some actions, or even allow it to perform new actions that it was unable to perform before. Being able to perform some new action may in turn enable the robot to build a new type of model, allowing knowledge and competence to be built up progressively.

As an example, consider the problem of trying to pick up a cup. If the robot does not know where its hand is, then it will be almost impossible for it to pick the cup up safely. Therefore a kinematic model describing how the robot's hand relates to its vision system, becomes very useful for the robot as it would enable, or at least greatly assist, the robot to position its hand to pick up the cup. Now, in keeping with the theme of autonomous model building, the kinematic model should be constructed by the robot itself. One method for doing this could be to first have the robot track the movement of its hand with its camera. The robot could then use the tracked movements to infer the dimensions of its arm, and the position of the joints that come between the vision system and the hand.

In order to visually track its hand, the robot must have some model of what it looks like. The form the model should take is open for debate. It could be a full 3D model, or it could be much simpler, consisting just of a list of features that can be located in the image that comes from the robot's cameras. The point is, that a visual model of its hand enables the robot to build a kinematic model of its arm, which in turn enables it to plan and execute the actions needed to pick up a cup safely. This chain of models and actions is shown in Figure 1.2.

This chain of actions and models can be built on further by considering that

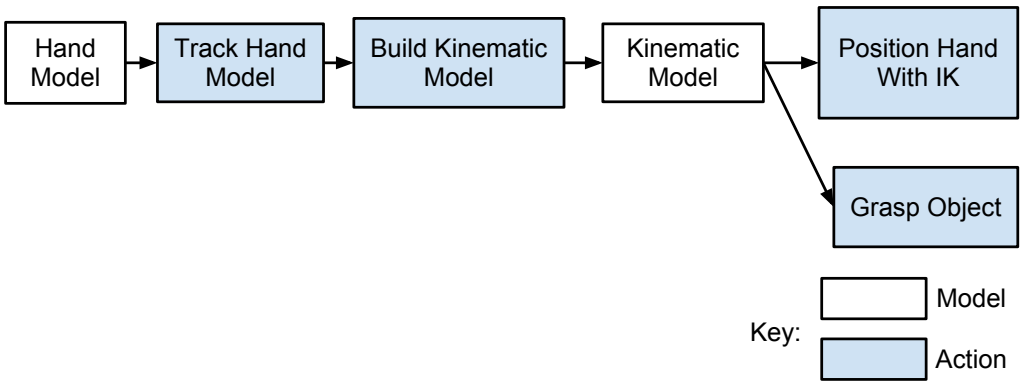


Figure 1.2: Chain of dependencies which shows how a robot with a model of its hand is able to perform actions which allow it to build a kinematic model of its body and then ultimately pick up an object

once the robot can pick up the cup, it can examine it more closely and build a model of it. This is not to say that the robot would not be able to start to build a model describing the cup before it picks it up, but being able to pick up and manipulate the cup, offers the chance to obtain a lot more information about the cup than would be possible from a purely passive observation. For example, whilst manipulating a bottle the robot may discover that the bottle has a cap, which can be twisted off. Perhaps if the robot is equipped with a text reading system it may be able to discover that the bottle contains water. This is all information that could be added to a model of the bottle. Also this extra information may suggest other actions, such as pouring a drink, which could be performed with the bottle.

The chain of actions and models can also be extended backwards in the sense that the model of the hand must come from somewhere, and so it is possible to look for actions that the robot can perform in order to autonomously build the model. These actions may in turn require further models, or it may be possible to implement the actions as purely reactive behaviours.

This is not to say that a robot should necessarily build *all* of the models it uses autonomously. At the low level in particular, it may be much simpler and more convenient to simply hard code knowledge into the robot (joint limits for example) than to have the robot try to discover the information for itself. However the ability to build models, lets the robot cope with unfamiliar objects, and the ability to refine or rebuild existing models with new information, gives the robot the ability to cope with situations where its models become inaccurate. As an example, a robot's kinematic model may become inaccurate due to modifications of the robot, or as a result of wear and tear. Being able to adapt or rebuild its models gives a robot a longer useful life and reduces maintenance costs, and is therefore a valuable tool

in the effort to increase long term autonomy for robots.

1.2 Thesis Overview

This thesis attempts to tackle the problem of enabling a robot to autonomously build models of itself and of objects around it by using vision and manipulation. In particular

- it proposes that a layered approach to model building will allow a robot to proceed progressively from a state of very little knowledge, to having rich representations of itself, and objects in the world.
- it proposes that an active approach to learning, whereby a robot gains information by performing exploratory actions, provides an effective way of obtaining information for building models.

1.2.1 Aims and Objectives

To support the hypotheses set out above, the aim of this work is to build a series of systems that allow a robot to build models of itself, and models of objects in the world around it. Specific objectives of this work are

- to create a system that allows a robot to progressively build models that describe the physical dimensions and kinematics of its hand and arm.
- to extend the system so that the robot can build models of objects in the world around it.
- to explore the use of exploratory actions as a means of enabling a robot to gain information about itself, or about an object it is examining.

To achieve the objectives an 'engineering' approach is taken whereby the proposed systems are constructed explicitly to achieve the aim of being able to autonomously build the desired models. This contrasts with another commonly used approach where systems are constructed in order to mimic (and therefore hopefully validate) potential biological mechanisms identified by developmental psychology research. This was done because the primary objective of this research was to enable robots to autonomously build models, not to determine how biological systems, such as humans and animals achieve this task. This is not to say that biological systems should not be used as a source of inspiration for engineered systems.

1.2.2 Contributions

The main contributions of this thesis, along with the chapters in which they can be found are as follows:

- A method for enabling a robotic system to build a kinematic model of itself using a series of exploratory actions is described and evaluated. It is shown how the robotic system can start with a very limited amount of information, and then use exploratory actions to gather information about its body via its visual (depth camera) and proprioceptive systems. As far as the author is aware this is the first markerless system that allows a robot to build a kinematic model of itself with a depth camera. (Chapter 4)

- A novel method is presented that makes use of vision in order to home the joints of a robot when it is powered on. This method reduces the need for traditional methods of homing a robot, such as absolute encoders, or limit switches, and so offers the possibility of building cheaper and more reliable robotic systems. (Chapter 5)

- In the context of visual homing, the concept of an exploratory joint space that a robot may safely move in whilst attempting to home itself is described. This new concept is important for ensuring that robots can visually home themselves without damaging themselves. (Chapter 5)

- A novel edge based tracker is presented that incorporates colour cues and can be used to track the motion of articulated flat pack cardboard boxes. (Chapter 6)

- A hinge exploration strategy that combines both visual and haptic feedback in order to evaluate the presence or absence of a hinge is described and evaluated. (Chapter 6)

- A novel robotic text reading system is described, which makes use of exploratory actions in order to find and read any text which may be on an object. This system, to the best of the author's knowledge, is the first system that attempts to give to a robot the general purpose skill of reading an object in its hand, in a form that would be recognisable to a human. (Chapter 7)

Much of the work in this thesis has been presented separately as conference or journal papers, with the work for these papers being done by the author of this thesis.

- Alan Broun, Chris Beck, Tony Pipe, Majid Mirmehdi, and Chris Melhuish (2012). Building a kinematic model of a robot's arm with a depth camera. In: *Towards Autonomous Robotic Systems*, pp. 105–116

- Alan Broun, Chris Beck, Tony Pipe, Majid Mirmehdi, and Chris Melhuish (2014a). Bootstrapping a robot's kinematic model. In: *Robotics and Autonomous Systems* 62.3, pp. 330–339
- Alan Broun, Chris Beck, Tony Pipe, Majid Mirmehdi, and Chris Melhuish (2013). Robotic manipulation strategies for detecting and reading text. In: *ICRA Interactive Perception Workshop*
- Alan Broun, Chris Beck, Tony Pipe, Majid Mirmehdi, and Chris Melhuish (2014b). Visual homing of an upper torso humanoid robot using a depth camera. In: *Towards Autonomous Robotic Systems*, pp. 114–126
- Alan Broun, Chris Beck, Tony Pipe, Majid Mirmehdi, and Chris Melhuish (In preparation). Active visual search strategies for robotic text reading.

The author of this thesis also coauthored the following publication, with the work being done by Chris Beck.

- Chris Beck, Alan Broun, Majid Mirmehdi, Tony Pipe, and Chris Melhuish (2014). Text line aggregation. In *International Conference on Pattern Recognition Applications and Methods*, pp. 393–401

1.2.3 Outline

The rest of this thesis proceeds as follows. Chapter 2 describes existing work in the field of autonomous model building, Chapter 3 describes the robotics platform used in this work, Chapter 4 presents a method for autonomously building a kinematic model of a robot's arm, Chapter 5 extends this work with a visual homing system for the robot. In Chapter 6, the focus of the thesis expands to include external objects and shows how exploratory actions can be used to determine the presence and location of hinges when building models of articulated objects. Chapter 7 explores how models can be built of objects within the environment using any text that may be on them. Finally, Chapter 8 presents conclusions and discusses possible directions for future work.

Chapter 2

Background and Existing Work

This chapter reviews existing work that motivates, and provides some background for the task of autonomously building models for robotics. Robotic systems can be applied to a truly vast array of applications, and so the types of models that may be built are similarly diverse. In this work though, the focus is on autonomously building models of the robots themselves, and of small objects in their local vicinity that can be held in, and manipulated by, the robot's end effectors.

The review starts in Section 2.1 by looking at some of the work in the fields of neuroscience, and developmental psychology that suggests that the human brain may make use of internal models. In particular, the concept of the body schema is examined, along with the possibility that internal models are used for motor control.

In robotics, a broad spectrum of approaches can be used to build and maintain models that a robotic system can use to control itself. This ranges from the calibration of existing models, supplied by the system designer, to machine learning systems that attempt to learn how to control a robotic system using methods inspired by nature. Section 2.2 starts at one end of this spectrum by looking at work that has been done on the kinematic identification and kinematic calibration of robotic systems. This work is mature, and has found heavy application in the field of industrial robotics. The emphasis is very much on refining kinematic models that have been built by the designers of a system, but the techniques presented will find use in any system that attempts to build its models autonomously. In Section 2.3 an overview is given of the many different techniques used to build models for robotics systems. Section 2.4 looks at approaches that have been taken to building models of objects for robotics, and in particular looks at the wide range of sensor modalities that have been used to provide data for these models. Finally, Section 2.5 concludes the chapter with a brief discussion of how the work reviewed is relevant to this thesis.

Alongside the review presented in this chapter, additional 'Background and

Existing Work' sections have also been added to a number of other chapters. The reason for doing this is to provide additional background material, and to reference any relevant techniques used at the most appropriate point.

2.1 Models Within Biological Systems

Much of the work that has been done in the field of robotics, has been inspired by findings in biology, neuroscience, and developmental psychology as scientists attempt to understand the workings of natural creatures. Robotic systems are built to act as test-beds in order to examine hypothesised workings for specific biological mechanisms, or else ideas are taken directly from nature and incorporated into more traditionally engineered robotic systems.

Consequently, some of the models which it may be advantageous to learn for controlling a robotic system are analogous to models which are proposed to exist with biological systems.

2.1.1 Body Schema and Body Image

The *body schema*, is a term arising from the study of biological systems. It is defined by Vignemont (2009) as a sensorimotor representation (meaning that it incorporates both sensory and motor systems) of the body that is used for performing actions. Vignemont (2009) distinguishes the body schema from a related concept called the *body image* which provides a way of grouping other representations of the body which are not related to performing actions.

One key feature of the body schema in a lot of animals, but most notably primates and humans, is the plasticity of the body representation, meaning that the representation is not fixed, but can instead adapt over time. This is studied in (Rochat 1998) as young babies learn to use and control their bodies by making exploratory actions, and then over time children are able to cope as their body grows and changes, adapting and incorporating modifications into their body schema. Meltzoff and Moore (1997) describe how young babies use motor babbling (random movements which exercise the body) to learn how to control their bodies and to imitate adult facial expressions.

It has been shown that people's body schemas can adapt and change over quite short time periods. One of the most famous examples of this is the 'rubber hand illusion' investigated by Botvinick and Cohen (1998) in which subjects started to believe that touches they felt on their hand (which was hidden), were actually taking place on a rubber hand which they could see.

The body schema can also expand in order to incorporate external items such as tools, as discussed in (Carlson et al. 2010) and (Maravita and Iriki 2004).

2.1.2 Internal Models for Motor Control

There is strong evidence that internal models are used for motor control in biological systems. Kawato (1999) gives a good review of the evidence for the use of internal models for motor control, along with proposed forms which these models might take.

The view presented by Kawato (1999) is that internal models are needed, because feedback delays in biological systems can be large (visual feedback can involve delays of 150 to 250ms) and even fast spinal feedback loops have delays in the order of 30 to 50ms. Therefore it would be infeasible to execute fast movements accurately using nothing but feedback control. If the biological system is able to learn forward and inverse models of its body, then it can use the forward model to predict the results of its actions, and inverse models can be used to generate control inputs for feed-forward control.

Miall et al. (1993) describe how the behaviour of the cerebellum might be described as a Smith predictor, which is a controller that uses both feedforward and feedback control to control a system which contains delays (see Figure 2.1). Also, Wolpert et al. (1998) present evidence that the cerebellum generates predictions using a forward model and show how multiple paired forward and inverse models could be advantageous for motor learning and control.

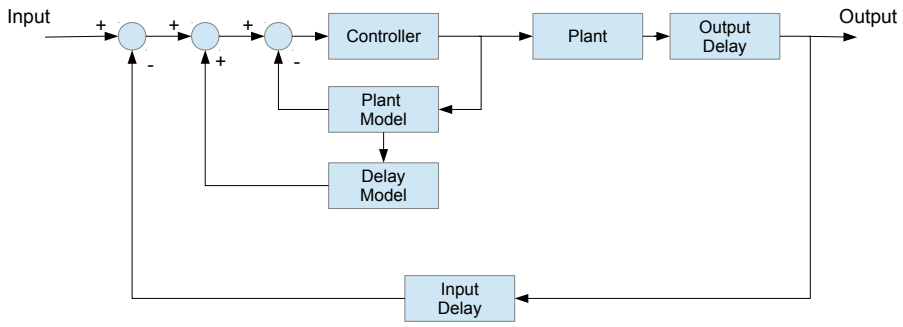


Figure 2.1: A Smith predictor is a controller that makes use of an internal model of a controlled system, and a model of the delays within that system in order to provide good control. Miall et al. (1993) propose that the cerebellum may use something similar to Smith predictors.

2.2 Calibration of Kinematic Models

Within robotic systems, one of the most commonly used model types are kinematic models. Kinematic models of robots describe how the rigid bodies, known as links

which make up a robot, are positioned and orientated by varying the position of the actuators of the robot. A kinematic model assists in determining the forward kinematics of a robot (the mapping from joint angles to an end effector pose), and the inverse kinematics of a robot (the mapping from end effector pose to joint angles).

In the general case, a rigid body in 3 dimensional space requires 6 parameters to fully define its position and orientation. However, by making use of the fact that the position of a link is partially determined by connected links, and carefully placing the coordinate frame for each link, Hartenberg and Denavit (1955) were able to show that the pose of each link can be specified by just 4 parameters, now known as the Denavit-Hartenberg parameters.

Kinematic calibration is the process of refining the kinematic model so that predictions made about the pose of the robot using the model agree as closely as possible with measurements taken of the robot pose. Errors between the predicted and measured poses are divided by Whitney et al. (1986) into *geometric* errors and *non-geometric* errors. Geometric errors arise from differences between the geometry of the robot and the kinematic model. The excessive cost of manufacturing systems to within very tight tolerances means that there are always slight differences in link lengths, and with regards to the orientation of joint axes. Non-geometric errors are all of the other possible errors that can occur and amongst other things includes the effect of backlash in gearboxes and the fact that the robot's links may not be entirely rigid, but may bend.

Mooring et al. (1991) gives a good overview of the general process of kinematic calibration which involves measuring the end effector's pose for a variety of positions in joint space, and then modifying the kinematic model, usually using non-linear optimisation methods to minimise the error between the predicted and measured poses. Variations on this theme include *closed loop* methods such as (Bennett et al. 1992) and (Hollerbach and Lokhorst 1995) where the end effector is attached to the environment, either directly, or through a force sensor, which gives constraints on the geometry of the robot which can be used in the non-linear optimisation step. There are also *screw axis* methods such as (Stone et al. 1986) and (Hollerbach, Giugovaz, et al. 1993) which identifies and refines the position and orientation of joint axes by measuring the position of the end effector as a joint is moved through its range of motion, and then fitting a circle to the measured points. The plane normal at the centre of the circle corresponds to the axis of the joint being measured.

Alongside the many methods for performing kinematic calibration, there are also a variety of forms that the kinematic model can take, as researchers seek representations with desirable characteristics that aid the non-linear optimisation step. The standard Denavit-Hartenberg parameters representation for example has discontinuous regions in its parameter space for the very common occurrence of

nearly parallel axes and so a number of modified Denavit-Hartenberg representations such as that presented by Hayati and Mirmirani (1985) have been proposed to remove this discontinuity.

Hollerbach and Wampler (1996) brought together a lot of work on kinematic calibration by presenting a taxonomy of calibration methods, and also by defining a calibration index $\mathscr{C}$ as

$$\mathscr{C} = \mathscr{S} - \mathscr{M} \tag{2.1}$$

Where $\mathscr{S}$ is the number of *sensed joints* a robotics system has (i.e. where a sensor is available that can be use to measure a joint's angle), and $\mathscr{M}$ is the mobility index of the system which represents the number of degrees of freedom of the system. The calibration index of a kinematic calibration method varies from $\mathscr{C} = 6$ where the full position and orientation of the end effector is measured, down to $\mathscr{C} = 1$ where just 1 dimension of the end effectors 6 dimensional pose is measured.

2.3 Building Models of Robotic Systems

Some researchers have been inspired by evidence for the body schema, and taken techniques from classical kinematic calibration and used this to construct models of humanoid robots. Hersch et al. (2008) described a system able to learn a kinematic model of itself by observing the movement of its end effectors. Sturm, Plagemann, et al. (2009) used Gaussian processes as a method for estimating the most likely transformations to describe the motions of observed robot links.

Work has also been done to look at how robots can use some ideas from developmental psychology to learn how to control themselves. Dearden and Demiris (2005) presented a robotic system that used dynamic Bayesian networks to learn how its actions affected its body, and then used the same model to imitate the actions of a human.

Machine learning provides a promising array of tools for tackling the problem of autonomous model building as it explores the process of learning from data. Nguyen-Tuong and Peters (2011) try to classify the different types of models that can be learnt as different types of graphical models, and discuss attempts that have been made to apply machine learning to building models for robotic systems.

Even if a robotic system is not going to be learning, and building models autonomously for itself, machine learning can still play a large role in the implementation of models provided by system designers, where the process of specifying a model by hand may be infeasible. Angelova et al. (2006) used machine learning to build a set of appearance models for different types of terrain, and then learnt models of wheel slip for each of the different terrain types so that

they could provide predictions of potential wheel slip for a path planner designed for use on a Martian rover.

Models for robotics often need to change and to be adapted over time, as the robotic system and the environment it operates in are unlikely to be constant, and are liable to change. Lazy learning methods are methods for machine learning that delay the task of generalising from their training data, i.e. in general, they keep all of their training samples, and do not transform it into an intermediate form such as the weights of a neural network. Instead, when a query is received, lazy learning methods will make use of a set of relevant training samples in order to provide an answer. Lazy learning methods are flexible in the face of a changing environment, as new training samples can be incorporated with little effort over time and, optionally, old training samples can be discarded.

One class of lazy learning methods are locally weighted learning algorithms, which provide outputs by using a weighted combination of training samples that are physically close to a given input. Locally weighted learning algorithms can take a variety of forms, based on the methods used to combine training samples and are described in detail by Bellamy (2006). In a companion paper Atkeson et al. (1997) also describe experiments showing how locally weighted learning algorithms can be used for learning control policies for a number of robotic systems.

This line of research was used successfully by D'Souza et al. (2001) to learn an inverse kinematics model of a 30 degree of freedom humanoid robot using a method called Locally Weighted Projection Regression (LWPR). Rather than learn a general inverse kinematics function, which for a redundant manipulator is almost never a one-to-one mapping, they instead learnt the mapping

$$f(\dot{\mathbf{x}}, \boldsymbol{\theta}) \mapsto \dot{\boldsymbol{\theta}} \tag{2.2}$$

which given a position in joint space $\boldsymbol{\theta}$ along with a desired instantaneous velocity in Cartesian space $\dot{\mathbf{x}}$ provides the necessary change in joint space $\dot{\boldsymbol{\theta}}$. D'Souza et al. (2001) were able to show that not only could LWPR learn the inverse kinematics needed to draw a figure of eight from a period of motor babbling (semi random exploratory motions), but also it could greatly improve the accuracy of the motion through a process of online learning as it tried to perform the motion. LWPR was also used by Schaal et al. (2002) to learn the inverse dynamics of a robot.

Bongard et al. (2006) enabled a robot to build a model of itself by incrementally generating possible models of its body, and then assessing the validity of those models by comparing sensor readings obtained from performing exploratory actions against predicted sensor readings obtained from running the same actions in a simulator. Bongard et al. (2006) demonstrated how a robot could be made robust to damage by incorporating this model building ability. In

particular their work showed that robot was able to recover from the loss of a leg and to use its model to generate new gaits.

2.4 Object Models

Alongside the work that has been done to build models of a robot's body, work has also been done to build models of objects in the world around the robot. The need for building these models autonomously has perhaps been felt more keenly than the need for autonomously building models of the robot's body, as the vast array of objects that a robot operating in a human environment may encounter makes the notion of providing the robot with models of everything it is likely to encounter obviously unrealistic.

The problem of building 3D rigid models of objects as described by Bernardini and Rushmeier (2002) is well studied and has been the subject of intense research due to its many applications in areas such as industrial design, 3D graphics and archaeology to name but a few. With the advent of lightweight depth cameras it is possible to give robots a 3D view of the world and to enable them to build 3D rigid models of objects held in their hands, as shown by Krainin et al. (2011).

One advantage that robotic systems have when building models of objects is that the model building process does not have to be passive, with the robot just observing the object. This has led to many researchers exploring the idea of *interactive perception*, with the robot performing actions in order to gain more information about the object it is interacting with, and to clarify potentially ambiguous sensor data. Christiansen et al. (1990) built a system that was able to build motion models for the movement of planar objects on a tilting tray. Fitzpatrick and Metta (2003) adapted the Cog robot so that it could use pushing motions in order to help segment an object from the background. More interestingly, they also used the pushing motions to build models of how different objects move in response to pushes from different directions, and so given an object and a desired motion, the robot was able to choose potential push actions that would allow it to move the object in the desired fashion.

Katz and Brock (2008) extended the work of Fitzpatrick and Metta (2003) and used interactive perception in order to build models of articulated objects. Sturm, Pradeep, et al. (2009) used a probabilistic approach to determine the most likely joint type (such as revolute, or prismatic) that described the relative motion of two rigid bodies and was thus able to build kinematic models of articulated objects.

It is interesting to note that the form which an object model may take is not limited to 3D visual models, but is instead much wider, reflecting the vast array of sensor modalities such as hearing, touch etc that may be used to perceive an object. Sinapov et al. (2011) built a robotic object categorisation system that was able to

distinguish between objects based on joint torque readings, and the sounds that were heard whilst performing a variety of interactions such as crushing, pushing and dropping objects. Griffith, Sinapov, et al. (2009) built models capturing the extent to which an object was a container by observing the motion of a cube that was dropped onto the object.

Finally, Griffith, Sukhoy, et al. (2012) showed how a robot could learn to classify objects based on the sounds heard as they were held in various orientations under running water in a sink. This demonstrates that the very tasks that a robot might be called upon to perform, such as washing up dishes, provide a rich environment for learning and enhancing object models in order to potentially improve the reliability of those tasks.

2.5 Relevance to Thesis

This chapter has attempted to lay out the foundational research upon which the rest of the thesis is built. What sets this thesis apart from work that has gone before, is an attempt to take a layered approach to enable a robotic system to progressively build models of itself and of objects in the world around it, coupled with an emphasis on active learning in the form of devising exploratory actions which allow a robotic system to obtain the information needed for those models.

Concretely, the work in this thesis draws inspiration from the work on biological models reviewed in Section 2.1 and presents methods to enable a robot to autonomously identify, and then build models of its hands, before incorporating them into a wider kinematic model of itself, that again it builds autonomously. Within this work though, the aim is to make use of established techniques such as the kinematic calibration methods discussed in Section 2.2 rather than trying to recreate possible biological mechanisms, as was for example done by Brooks et al. (1999). Moving on, the work in Section 2.3 gives context to the search for methods for autonomously building models and provides many interesting ideas. Finally, the work in Section 2.4 has relevance to the work presented in Chapters 6 and 7 where attention expands out from the body of the robot, and attempts are made to autonomously build models of nearby objects.

Chapter 3

Robotic Platform

This chapter describes the robotic platforms used for the experimental work in this thesis and provides details of the supporting software which was written to control the robots.

3.1 Hardware Description

The majority of work presented in this thesis was done with a Bristol and Elumotion Robotic Torso (BERT) robot, shown in Figure 3.1. This is an upper torso humanoid robot, which is composed of harmonic drives and brushless DC motors. The motors are controlled with EPOS motor controllers and these in turn are driven by higher level software, running on a PC with communication over a Controller Area Network (CAN) bus. To give the robot a 3D vision system, a Microsoft Kinect has been used for the robots head. The Kinect is an affordable depth camera developed by the company Primesense, that uses an infra-red projector coupled with an infra-red camera as a stereo pair. The Kinect builds depth maps of the world with a resolution of 640x480 pixels at a frequency of 30Hz. The depth maps are fairly noisy, but a Kinect is very reliable and a lot easier to set-up than an equivalent stereo camera system.

In addition to using the BERT robot, experimental work was also carried out using a Baxter research robot manufactured by Rethink Robotics for the work presented in Chapter 6. The Baxter robot is shown in Figure 3.2, and provides a pair of 7 degree of freedom (DOF) arms powered with Series Elastic Actuators (SEAs). The SEAs are built by inserting a spring element between the gearbox and the load, and this gives the Baxter robot a degree of natural compliance, important for a robot that is designed to work in close proximity to humans. Another benefit of using SEAs which influenced the decision to use Baxter in this work is that by measuring the deflection of the spring element, it is possible to

estimate the forces being experienced by the robot's arms.

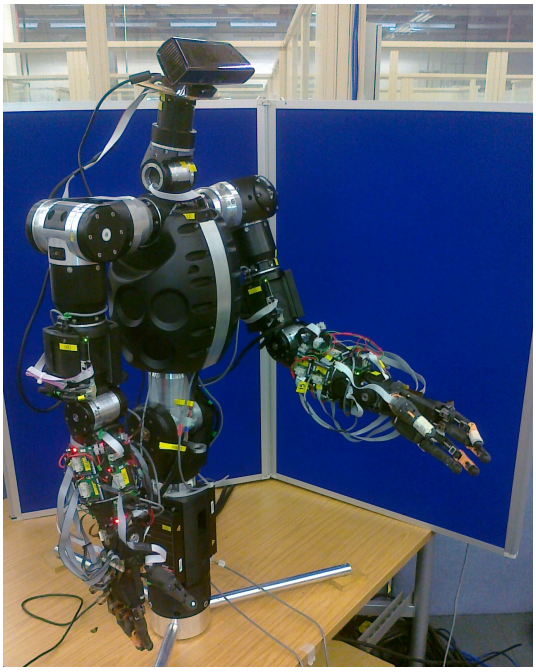


Figure 3.1: The BERT robot examining its hand.

Figure 3.2: The Rethink Robotics Baxter research robot.

3.2 Developed Control Software

A range of software systems were developed to support the experimental work carried out using the BERT and Baxter robots.

3.2.1 Software Developed for the BERT Robot

Figure 3.3 shows how the software systems that drive the BERT hardware are organised.

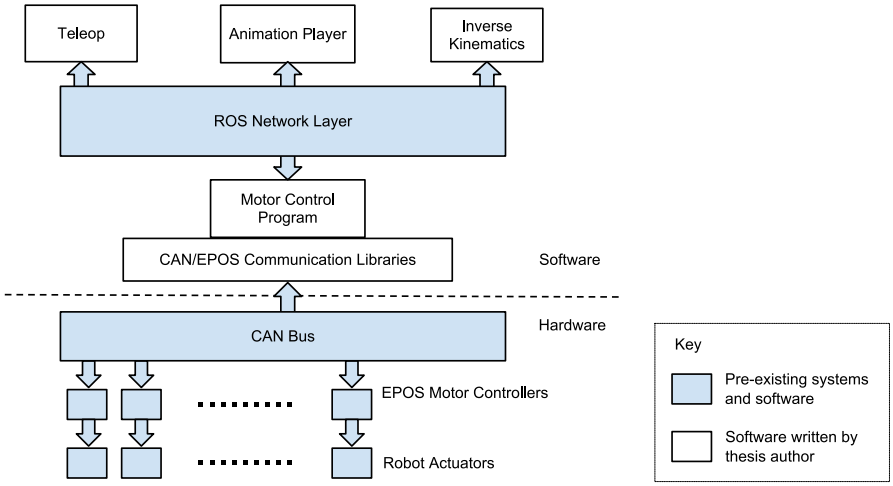


Figure 3.3: The hardware and software systems used to control BERT.

At the start of the project, the hardware shown in Figure 3.3 was already available. The software stack in Figure 3.3 was then built on top of it over the course of the first year. Adopting a bottom up approach, the first thing written was a library that controls the EPOS motor controllers by communicating over the CAN bus. The EPOS motor controllers are sophisticated pieces of hardware, which provide a range of options for controlling the motors attached to them. Options include position control, velocity control and current control. For now, the EPOS communication library exposes just the position control functionality of the motor controllers. The library provides the ability to set the desired position and speed of the actuators of the robot, as well as the ability to read back the current position of the actuators at a rate of 100Hz.

The higher levels of the software stack shown in figure 3.3, have been built using a mixture of C++ and Python. Also, extensive use has been made of the open source Robot Operating System (ROS). This package is not (despite its name) a full operating system, but rather a library for Linux that provides the

means to network together a number of individual software processes, in order to build robotic applications. The flexibility of being able to compose an application from a number of different processes, makes it very easy to distribute the processing over a number of machines. It also allows a number of other open source libraries to be used as services. For example, ROS provides wrapped versions of a number of planning libraries, which makes it very easy to incorporate inverse kinematics into the robotic system.

Using this architecture, it has been easy to build a number of high level control programs, such as a joint teleoperation program and an animation player program. These perform very little work themselves, but instead delegate it to other processes running under ROS. Figure 3.4 shows a graph of the ROS nodes that run for a hand tracking program used for part of the work presented in Chapter 4.

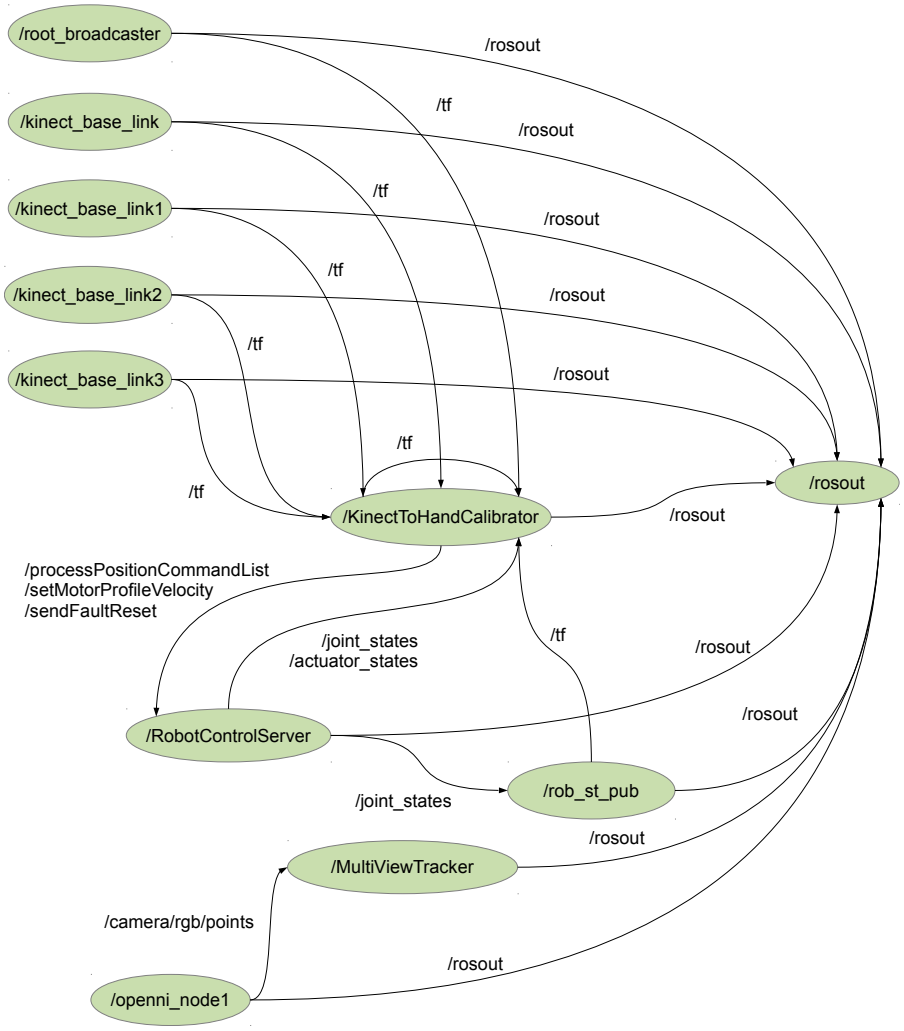


Figure 3.4: This graph showing the nodes (processes) running for a hand tracking program, developed by the author, used for part of the work presented in Chapter 4. The directed edges between nodes represent communication channels, with /tf for example representing 3D transformation data, and /rosout representing status output from each of the nodes.

3.2.2 Software Developed for the Baxter Robot

Rethink Robotics, the company that manufactures the Baxter robot, provide a number of software packages for use with the robot. This means that a lot of the

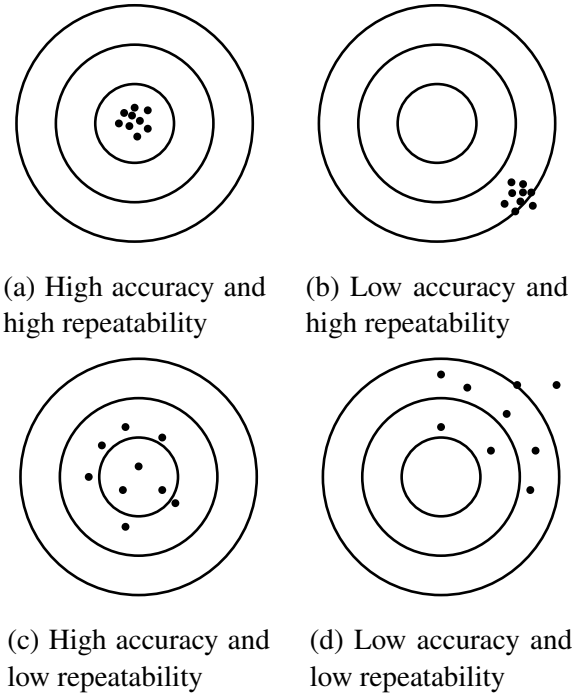
low level work that had to be done to support the BERT robot, and to get it to move, was not necessary for the Baxter robot. In particular, Rethink Robotics provide a ROS interface for the Baxter robot that makes it easy to move the robot in position, velocity, or torque control modes. Therefore, the main work done to support the Baxter robot was to retarget some of the higher level programs to work with both it, and the BERT robot. This was straightforward, as the ROS interfaces the robots use for controlling motion are very similar.

3.2.3 Other Software Developed for the Robots

Alongside the software detailed in the previous sections, other software packages were developed for each of the other chapters in this work. This software made use of a number of open source 3rd party libraries such as Eigen (Guennebaud, Jacob, et al. 2010) for manipulating matrices and SciPy for numerical optimisation (Oliphant 2007). In general though, unless noted in a specific chapter, all software systems presented as part of this work were implemented by the author.

3.3 Evaluating Accuracy and Repeatability

Before embarking on the task of trying to get a robot to autonomously build models of itself, it is useful to get some sense of how precisely a robot can move, in order to gauge the potential resolution of the models that it may be able to build in the future. Standard characteristics measured for a robot include its accuracy and repeatability. Accuracy measures how a robot's actual position compares to its commanded position, and repeatability measures how closely a robot returns to a previously commanded position. Figure 3.5 shows examples of the different combinations of accuracy and repeatability that a robot may exhibit.



(a) High accuracy and high repeatability

(b) Low accuracy and high repeatability

(c) High accuracy and low repeatability

(d) Low accuracy and low repeatability

Figure 3.5: The possible combinations of accuracy and repeatability.

For industrial robots, the methods and procedures for measuring accuracy and repeatability are laid out in great detail in (ISO 9283 1998). To be done properly, these tests require a temperature controlled environment, and expensive equipment such as a 3D Coordinate Measuring Machine (CMM), and specially manufactured test load. For the work done in this thesis, it was only necessary to obtain a general indication of the accuracy and repeatability of the robots, and so it was decided that it was not worth the cost and effort required to perform the ISO 9283 (1998) tests in full. Instead some less complicated, easier to perform tests were devised and are described below.

3.3.1 Accuracy and Repeatability of the BERT Robot

The accuracy of the BERT robot is fairly low as no accurate kinematic model is available, and the initial pose of the robot is always fairly uncertain as the robot's joints have no absolute encoders, and few physical end stops that can be used to home the joints. To measure the accuracy of the robot, the joints were first zeroed as carefully as possible by eye using home positions marked with pieces of tape. Then the left arm of the robot was commanded to move to a number of different positions. The arm was moved slowly (less than 5cm/s for the end effector) and allowed to settle fully after each movement to minimise any dynamic effects on the

position measurements. After each move the actual (x,y,z) position of the robot's wrist joint relative to its base was measured with a tape measure, and compared to the commanded position. This showed that the BERT robot was accurate to within 3cm, in that all measured positions were within 3cm of the corresponding commanded positions.

The accuracy of a robot can be improved a lot with an accurate kinematic model, and so the relative inaccuracy of the BERT robot is not really something to be concerned about, especially considering that it was originally commissioned for recreating gestures for human robot interaction which do not generally require a high degree of accuracy. More important is the repeatability. Given a set of N position measurements, with coordinates (x,y,z), the repeatability r of a robot is defined by ISO 9283 (1998) as follows.

$$r = \bar{l} + 3\sigma, \tag{3.1}$$

where

$$\bar{l} = \frac{1}{N}\sum_{j=1}^{N} l_j, \tag{3.2}$$

$$l_j = \sqrt{(x_j - \bar{x})^2 + (y_j - \bar{y})^2 + (z_j - \bar{z})^2}, \tag{3.3}$$

$$\bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j, \bar{y} = \frac{1}{N}\sum_{j=1}^{N} y_j, \bar{z} = \frac{1}{N}\sum_{j=1}^{N} z_j, \tag{3.4}$$

and

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N}(l_j - \bar{l})^2}{N-1}}. \tag{3.5}$$

The repeatability of the BERT robot was measured by attaching a pen to the BERT robot's left hand, and commanding it to a pose that made a mark on some graph paper, secured with tape in front of the robot. Making marks on paper meant that the repeatability was only measured in 2 dimensions (i.e. $z = 0$ for all measurements), but this made measuring the repeatability easier and it was felt that repeatability in the z-axis was likely to be very similar to repeatability in the xy-plane. After the first mark had been made, the arm was then moved to a new pose, away from the paper, before being commanded back to the mark making pose, again all moves were performed slowly, and the robot was given time to settle. This was completed a total of $N = 10$ times, and the repeatability of the arm is calculated using Equation 3.1. The marks made are shown in Figure 3.6 and the calculated repeatability is 2.91mm.

3.3.2 Accuracy and Repeatability of the Baxter Robot

The tests carried out on the BERT robot were repeated for the Baxter robot, and the results obtained gave an accuracy of 1cm along with a repeatability of 3.02mm. The marks recorded for the repeatability test are shown in Figure 3.7

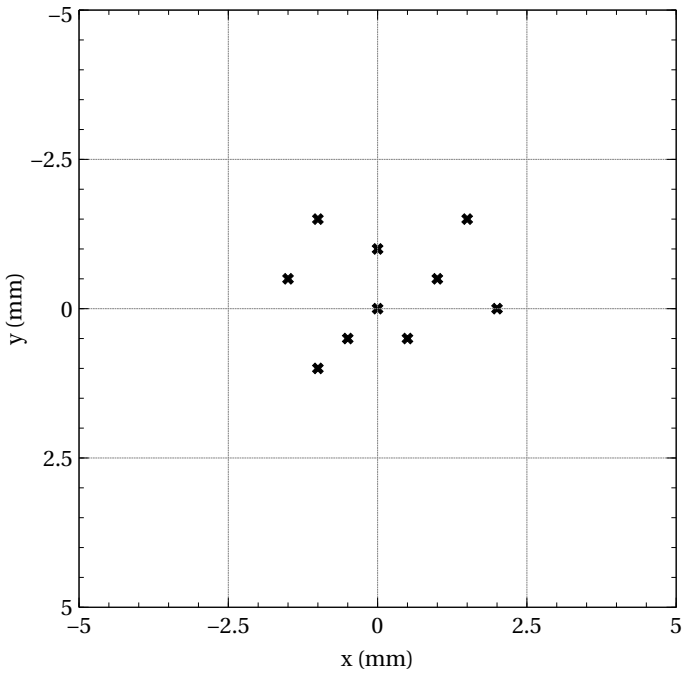


Figure 3.6: Marks made when measuring the repeatability of the BERT robot.

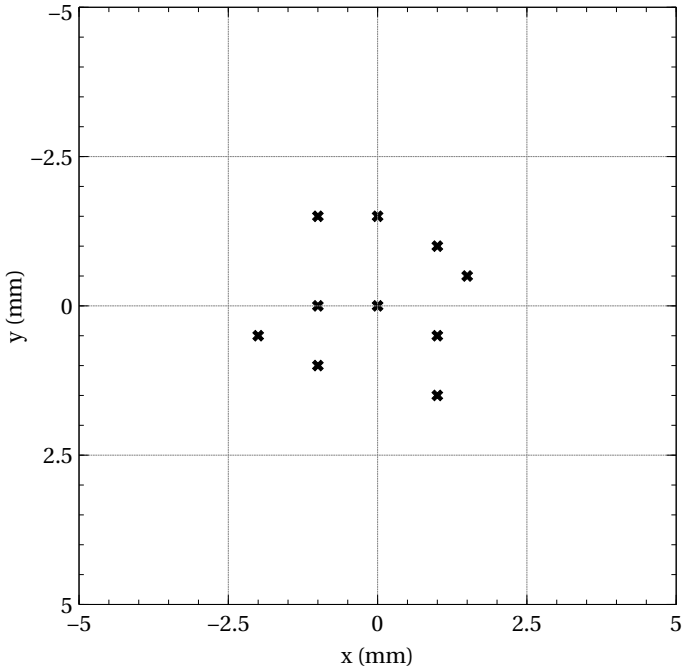


Figure 3.7: Marks made when measuring the repeatability of the Baxter robot.

3.3.3 Discussion

The repeatability values measured for the BERT and Baxter robots are not good compared with the repeatability quoted on datasheets for modern industrial robots. For example, the repeatability given for the ABB IRB 120 robot (a small industrial robot arm) is 0.01mm (ABB (2012)) which is 2 orders of magnitude better than the repeatability values measured for the BERT and Baxter robot.

Part of this discrepancy undoubtedly comes from deficiencies in the testing process that was used to measure the repeatability, which is much less rigorous than the ISO 928 method used to characterise the IRB 120 robot. However the main reason that the BERT and Baxter robots are less repeatable than the IRB 120 is that they have not been designed for high repeatability and are much less rigid than the IRB 120. The BERT robot has been constructed using 3D printed parts, and after a number of years there is fair amount of play in some of its joints. The Baxter robot is specifically designed to be compliant so that it is safe to work alongside, and in close proximity to humans (unlike the IRB 120 which must be installed in a safety cage). Indeed, the data sheet for the Baxter robot (RethinkRobotics (2015)) makes no mention of repeatability instead emphasising the safety of the compliant arms.

Having said that, the repeatability measured for the BERT and Baxter robot is adequate for the experimental work presented in this thesis which involves getting

the robot to learn about itself, and manipulating hand held objects in ways which do not require very precise and repeatable motion.

3.4 Controlling the Movement of the Robots

The focus of this work is on building systems that allow robots to autonomously build models of themselves, and objects in the world around them using active computer vision and exploratory actions. Overall, the models built focus on characteristics such as appearance, or physical dimensions. The dynamic characteristics of the robot and examined objects, i.e. how they respond to forces and torques are largely ignored and considered beyond the scope of this thesis. This is because it was felt that incorporating dynamic effects into the model building process would over-complicate things.

With that in mind, in all of the experiments described in this thesis, the robot is commanded to move in a slow, controlled manner so as to minimise dynamic effects. When taking sensor readings, these are either taken when the robot is moving slowly (as is the case for a lot of camera readings) or alternatively the robot is stopped, and a suitable pause is inserted to give the robot time to settle before a sensor reading is made.

Chapter 4

Autonomously Building Models of the Robot's Arm

This work focuses on building robots and robotic systems which can autonomously construct models of themselves and the external objects with which they interact. In particular, active 3D vision is used as a tool that a robot can use to explore itself, and its surroundings, in order to autonomously construct models.

The increasing availability of reasonably priced depth cameras such as the Mesa Imaging SwissRanger, or the Microsoft Kinect has made it easier for robotic systems to perceive the world in 3D. These cameras provide depth values for pixels in the image, and so produce a 3D point cloud in camera space. The quality of the point clouds produced by these cameras reduces the need for researchers to setup technically challenging stereo camera systems, which often rely on the presence of highly textured areas in order to achieve reasonably similar results.

A practical problem for a robot with 3D vision, is the task of relating the movements of its body to the Cartesian space of its camera system, so that it can interact with objects it sees. More fundamental than that, it may also be a problem for the robot to work out which parts of a 3D image belong to its body and to its hand.

This chapter presents a solution to both of these problems in the form of a system that allows a robot to reliably identify its hand in its field of view, and then to build a kinematic model of its arm in camera space. Building the kinematic model in camera space implicitly determines the transformation between camera space and the model. Once the kinematic model is built, inverse kinematics can be used to accurately move the manipulator of the robot in camera space. In effect, the robot is therefore able to 'bootstrap' itself, from a state of fairly limited knowledge, to having a kinematic model of its arm, which it can then use to further interact with, and to explore, the world.

The system works by first using a series of exploratory motions to roughly

identify the location, and extents of the robot's hand in its field of view. It then uses a simple form of visual servoing to move the hand to the centre of the field of view, in order to maximise the quality of the subsequent processes. Once centred, the system builds a model of the robot's hand by turning the hand in front of a Kinect depth camera, whilst aligning and merging the point clouds obtained from the Kinect into a common reference frame. The model is then used to track future movements of the hand, by aligning the model against incoming point clouds. This system has the useful property of not just providing an estimate for the transformation of the hand in camera space, but also providing a 3D model of the hand which can be useful for other tasks such as planning grasps, or checking for collisions between the robot and the environment.

Once a model of the robot's hand has been built and is being tracked, it is then used to automatically build a kinematic model of the robot's arm by tracking the movement of the hand as each revolute joint in the arm is rotated in turn. This allows the robotic system to build an accurate model of its arm, starting with very little information. This is an advantage, as a robot that can deduce information for itself, is potentially more robust, and requires less work to commission.

The rest of the chapter proceeds as follows. Section 4.1 describes related work and reviews the techniques which were used to build the system. Section 4.2 provides a description of how the robotic hand is modelled and tracked along with details of automatically building a kinematic skeleton for the robotic system. Section 4.3 evaluates the accuracy of the system, and Section 4.4 presents conclusions along with ideas for future work.

4.1 Background and Related Work

4.1.1 Exploratory Motion and Active Vision

The work of Ballard (1991) was amongst the first to look in depth at camera systems which were not simply passive. Ballard observed that more information may be obtained from a visual scene, or obtained at a lower computational cost, through the process of moving the camera system and observing the scene from a number of different viewpoints. Such systems are often termed *active* vision systems to distinguish them from passive vision systems.

An alternative to moving the camera in a system, is to move the object or scene being observed. The idea of using exploratory motions to both identify a robot's end effector, and also to segment objects of interest from the background was explored in work by Marjanovic et al. (1996) and later Fitzpatrick and Metta (2003) at MIT as part of the work on the Cog robot. The technique was explored in detail by Broun and Studley (2011), who showed that a waving exploratory

motion could be reliably detected, even in the presence of a large amount of distracting motion. Work has also been done by Katz and Brock (2008) on using exploratory motions to autonomously identify the structure of articulated objects.

4.1.2 Object Modelling and Tracking

When building models from range data, such as that obtained from a laser scanner or depth camera, the Iterative Closest Point (ICP) algorithm presented by Y. Chen and Medioni (1991) and Besl and McKay (1992) is a widely used algorithm for aligning one depth camera frame with either another frame, or with a reference model.

The ICP algorithm has been the subject of much research since its initial presentation. Rusinkiewicz and Levoy (2001) identified the key stages that make up the ICP algorithm, outlining a number of techniques for making the algorithm more efficient and speeding up convergence. The ICP algorithm was used as a key part of an object modelling and tracking system built by Weise et al. (2009), and a very similar system was used to build models of objects held in a robot's hand by Krainin et al. (2011). In both of these systems, models were constructed by first aligning point clouds from a depth camera into a common coordinate frame, using the ICP algorithm. Subsequently, corresponding points from the point clouds were averaged together to form surfel (surface element) models. Surfels as described in (Pfister et al. 2000) are orientated 3D points, which can be used to describe complex geometric objects without explicit connectivity information. The advantage of averaging point clouds together to form a surfel model is that it smooths out a lot of noise that would otherwise accumulate as a result of estimating lots of small transformations (Henry et al. 2010).

Tracking a robot's hand in camera space is a special case of tracking an arbitrary 3D object in camera space, and this is an active area of research with Lepetit and Fua (2005) providing a comprehensive survey of the main techniques. Fiducial tracking (Sturm, Plagemann, et al. 2009) involves tracking markers attached to the object of interest. Model based tracking involves posing a 3D model of the object of interest to best match the information coming from the camera. This method has been used extensively in human hand tracking applications, such as (Sudderth et al. 2004).

4.1.3 Kinematic Identification

There are a number of methods available for identifying a robot's kinematic model. Early work from the 1980s, includes Circle Point Analysis (CPA) used by Stone et al. (1986), and described in detail by Mooring et al. (1991). CPA involves fitting circles to observed endpoint locations in order to identify the axis of revolution

for a revolute joint. Another method for identifying the joint axes of a robot is the Jacobian Matrix Method of Bennett et al. (1992), this method requires either joint torque sensors or a method of estimating the linear and angular velocity of the robot's end link.

More recently, the field of developmental robotics has taken an interest in kinematic identification. Here, it has been explored as part of more general efforts to enable robots to build and maintain a *body schema* for themselves. In the context of robotics, Hoffmann et al. (2010) describe a body schema as a group of body representations, which allow an embodied agent to control its actions, and to integrate sensory information such as vision or touch into common frames of reference. These representations may include kinematic and dynamic models, and the emphasis is on building the models autonomously. The aim is to give a robot the ability to adapt to changes in its body due to damage, or to dynamically extend its body schema to allow the use of tools. Hersch et al. (2008) built a kinematic model of a robot by observing end effectors using an iterative gradient descent approach. Sturm, Plagemann, et al. (2009) presented a system that uses a Bayesian network to learn arbitrary kinematic chains, which can also cope with changes in the kinematic chains as the system runs. This system however, requires observations of all joint positions to build the kinematic chain, whereas the system presented here only needs to observe the movement of the end of the chain.

Finally, recent work by Hart and Scassellati (2011) takes a similar approach to the one presented here. The difference lies in the fact that the method of Hart and Scassellati requires an Augmented Reality (AR) marker to track the hand, whereas the method presented here builds and tracks a complete model of the robot's hand, and so can operate without AR markers.

4.2 Method

As this work is largely concerned with the development of self-learning robots, an attempt is made to reduce the amount of *a priori* information available, while making certain assumptions to ensure that the task remains tractable. It is assumed that an attainable starting pose for the robot is given, such that the robot's hand is visible to the robot's vision system. It is also assumed that the range of motion, and gearing ratio (used to convert from encoder ticks, to angles), of all of the robot's actuators are known. Finally, it is assumed that the relative order of the robot's actuators is available, i.e., that it is known that the elbow actuators come after the shoulder actuators etc. Everything else the robot needs to know, it deduces in a series of steps.

The assumptions made here were made in part to accommodate limitations of the robotic platform. In particular, actuator range of motion has to be known as the

BERT robot can damage itself if it attempts to move beyond its range of motion. It would be interesting however to consider how this initial set of assumptions could be reduced still further, but discussion of how this may be done is postponed until Section 4.4.

The complete system for building a kinematic model of the robot's arm involves a number of stages. The first thing the robot must do is to work out where in its field of view its hand is located, as described in Section 4.2.1. Once the robot has located and identified its hand, it needs to centre the hand in its field of view to obtain a good clear view of it, and to avoid any inconsistencies that could arise if the later hand modelling step, was to be performed from a variety of different viewpoints. The problem is that at this stage the robot has no kinematic model, and so therefore, no way of working out how moving its actuators will change the position of its hand, in its field of view. Section 4.2.2 describes how this problem can be overcome, and shows how the robot centres its hand in its field of view. Once the robot's hand is centred, the Kinect is used to construct and track a 3D model of the hand. This work is described in Section 4.2.3. Finally, Section 4.2.4 describes how the ability to track the movement of the robot's hand is then used to deduce the position and orientation of the actuator joint axes, and therefore, to finally build the kinematic model of the robot's arm.

4.2.1 Finding the Hand

Building on the work in (Fitzpatrick and Metta 2003) and previous work in (Broun and Studley 2011), a waving motion is used as the exploratory clue to locate the hand. To generate this motion, a sine wave is passed through the wrist actuator, which means that the angle of the wrist actuator γ as a function of time t is

$$\gamma(t) = A\sin(2\pi ft), \tag{4.1}$$

where A is the amplitude of the wave, and f is the frequency of the wave. The instantaneous linear velocity v of a point on the hand at distance r from the actuator axis is

$$v(t) = r\omega(t), \tag{4.2}$$

where ω, the angular velocity of the actuator, i.e., the derivative of the wrist actuator angle, is

$$\omega(t) = \frac{d\gamma}{dt} = 2\pi Af\sin(\frac{\pi}{2} - 2\pi ft). \tag{4.3}$$

The proportion of the linear velocity that will be seen on the image plane v_{image}, when the moving hand is viewed by the Kinect is

$$v_{image}(t) = \sin\beta v(t), \tag{4.4}$$

where β is the angle between the Kinect camera axis and the instantaneous velocity vector at the observed point. Combining (4.2), (4.3), and (4.4), it can be seen that the sine wave put into the actuator will be phase shifted, and have a very different amplitude by the time it is observed on the image plane. However, crucially, it will still be recognisable as a sine wave, and it will still have the same frequency.

In order to determine the velocity of points in the Kinect image, use is made of a block matching algorithm (J. R. Jain and A. K. Jain 1981). This algorithm works by first dividing up the image with a grid of blocks, and then for each block the algorithm estimates the optical flow to the next frame, by searching in a small window about the centre of the block for the best match in the next frame in a SAD (Sum of Absolute Differences) sense. An example of the optical flow obtained from this algorithm is shown in Figure 4.1.

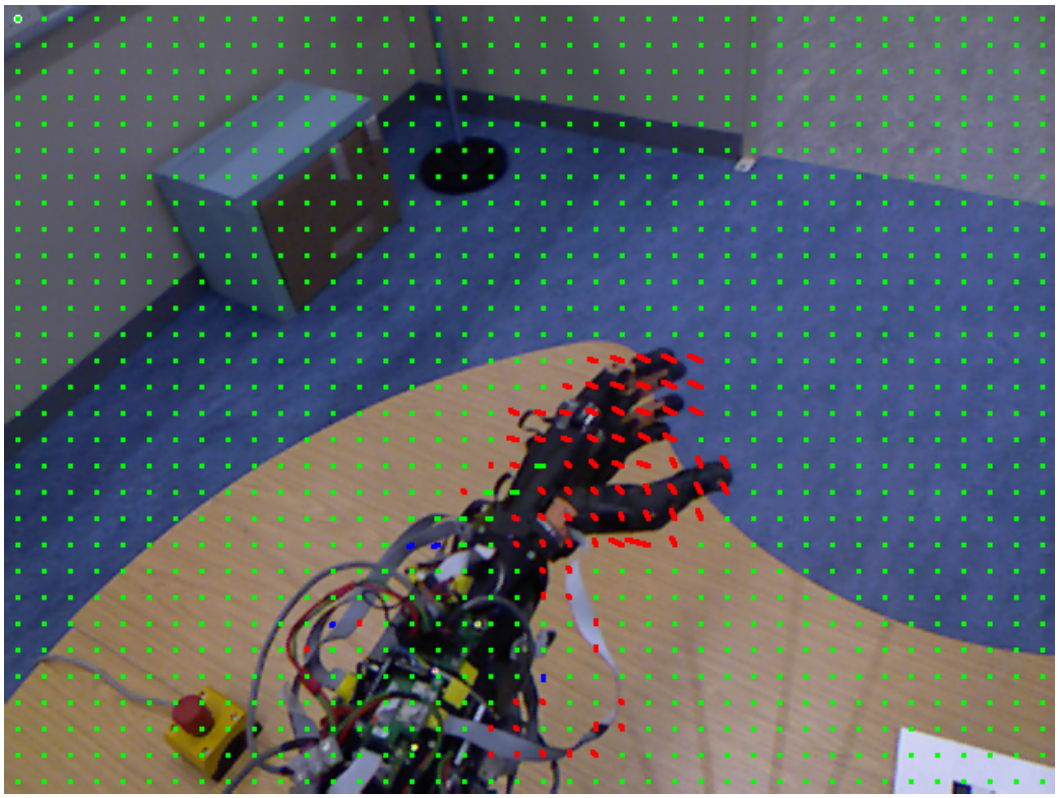


Figure 4.1: An example of the optical flow obtained from the block matching algorithm, in response to an exploratory waving motion.

To detect its hand, the robot starts a *detection episode* by briefly holding its hand still, and then passing a repeated sine wave through its wrist actuator (the exploratory motion). Whilst the sine wave plays out of the wrist actuator, the Kinect records images of the hand, and optical flow is calculated for each recorded image.

A small delay follows the end of the sine wave, to allow all of the motion to be recorded by the camera, and then the detection episode is concluded. The optical flow from all of the recorded images is then concatenated, to give a time dependent optical flow series for each block.

In order to identify the input sine wave in the output optical flow signal, Normalised Cross Correlation (NCC) is used. This involves treating the input sine wave and the output optical flow as two random processes, X and Y, and then calculating the correlation coefficient $\rho(X,Y)$, defined as

$$\rho(X,Y) = \frac{cov(X,Y)}{\sqrt{var(X)var(Y)}}, \tag{4.5}$$

where $var(.)$ and $cov(.,.)$ are variance and covariance, respectively. The correlation coefficient will be a value in the range [-1,1], with 1 signifying a perfectly matching pair of signals, -1 signifying that X is an inverted version of Y, and 0 signifying that the two signals are completely uncorrelated.

In order to detect the hand at a particular point in the Kinect image, the value of (4.5) is calculated multiple times between the optical flow, and repeatedly delayed versions of the input signal. If the correlation coefficient goes above a given threshold λ_{hand}, for both the x and y components of the optical flow, then the block is identified as having motion that matches the exploratory wave. Figure 4.2 illustrates a typical application of NCC to the optical flow of a block containing the hand.

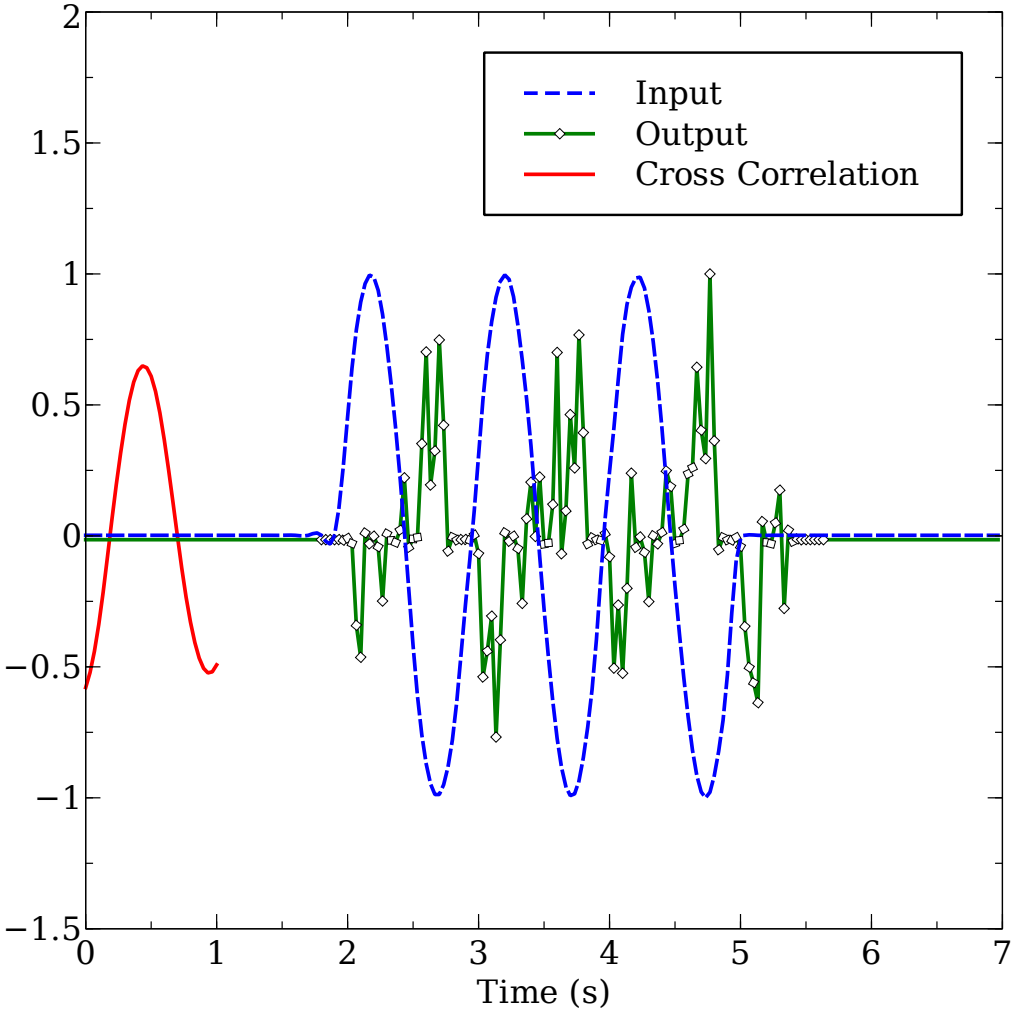


Figure 4.2: An example of performing NCC on the input signal and on optical flow from an 8x8 block of the camera image that contains the hand. NCC is repeatedly applied to compare the output signal to delayed versions of the input signal, with each application giving a point on the Cross Correlation line.

Once NCC has been run on all blocks, the hand is declared to be the largest connected set of positively identified blocks, and a bounding box gives the expected extents of the hand. An example of this is shown in Figure 4.3. For this work, the optical flow block size was set to be 8x8, and λ_{hand} was set equal to 0.52. The block size gave a good trade-off in terms of being able to calculate the optical flow reasonably quickly, and also view optical flow on the fingers of the hand. The value for λ_{hand} was chosen as a trade-off between being sensitive to the motion of the hand, and providing robustness in the presence of background distractor motion.

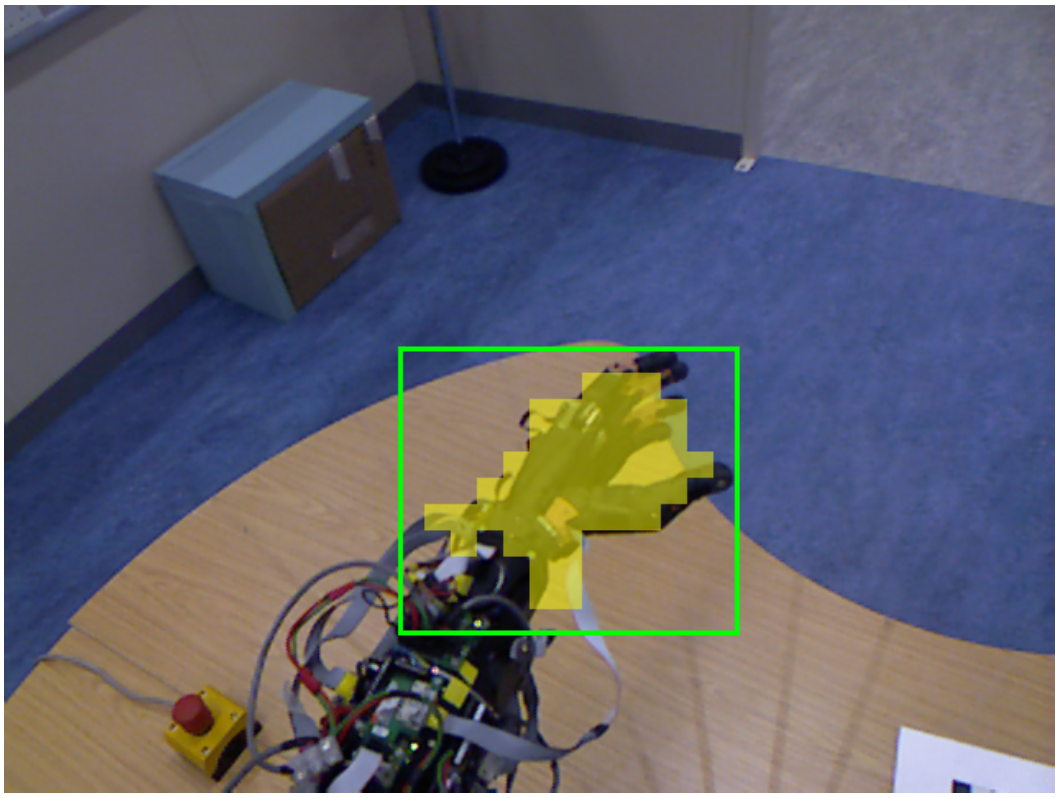


Figure 4.3: Typical output from the wave detection process. The regions in yellow represent blocks where the NCC response was above a preset threshold λ_{hand}.

It is important to realise that with this technique, points on the robot's hand are not being tracked as it waves. Instead, the instantaneous linear velocity is being observed at a grid of points on the image plane, and then compared to the signal put into the wrist actuator. The analysis in this section shows that a sine wave will be preserved during the transform from joint space to image space. However, it will only be preserved perfectly if the object being observed rotating back and forth about the wrist actuator is a solid disc. This is because, if there are gaps in the disc, then the observed velocity will drop to 0 as the gaps pass in front of the observation point on the image plane.

The robot's hand is obviously not a disc. However, it was still found to be possible to get good results with this technique by keeping the amplitude of the waves fairly small, so that during the waving motion, the long thin fingers of the hand, spent most of their time close to a small group of observation points.

4.2.2 Centring the Hand

Image Based Visual Servoing (IBVS) (Hutchinson et al. 1996) is used to centre the robot's hand in its field of view. IBVS makes use of the image Jacobian $\mathbf{J}(\boldsymbol{\theta})$, which gives a first order approximation of how the hand moves, given a change in the robot's joint angles. This means that

$$\mathbf{x}' \approx \mathbf{x} + \mathbf{J}(\boldsymbol{\theta})\Delta\boldsymbol{\theta}, \tag{4.6}$$

where $\boldsymbol{\theta}$ is a vector of the robot's joint angles, $\mathbf{x}$ is the position of the robot's hand in Kinect image space at joint angles $\boldsymbol{\theta}$, and $\mathbf{x}'$ is the position of the hand in image space that comes from applying the angle change $\Delta\boldsymbol{\theta}$.

Given the task of moving the hand to the centre of the robot's field of view $\mathbf{x}_c$, the joint angle change required can be calculated by using the Moore-Penrose pseudoinverse $\mathbf{J}(\boldsymbol{\theta})^+$ such that

$$\Delta\boldsymbol{\theta} = k\mathbf{J}(\boldsymbol{\theta})^+(\mathbf{x}_c - \mathbf{x}). \tag{4.7}$$

where k is a constant gain. Unfortunately, as a kinematic model of the robot is unavailable at this stage, it is not possible to calculate the image Jacobian $\mathbf{J}(\boldsymbol{\theta})$. Instead, it is estimated by making a series of exploratory motions with the robot, moving each joint in turn, one after another. After each move, the measured change in the hand position is used as the corresponding column in the image Jacobian matrix.

In order to track the hand whilst these exploratory motions are made, the assumption is made that for a small move, the appearance of the hand is unlikely to change significantly. Therefore, it is possible to construct a template of the hand using the bounding box obtained in Section 4.2.1, and then use template matching to track the hand. For this work the highly efficient LINE-MOD template matcher of (Hinterstoisser, Holzer, et al. 2011) is used.

Algorithm 1 provides a pseudo code overview of the approach used to centring the robot's hand.

Algorithm 1 Locating and centring the robot's hand

```
x ← LocateHandWithWave
while |x_c − x| > ε_hand do
    BuildJacobian
    Δθ ← kJ(θ)^+(x_c − x)
    x ← LocateHandWithWave
end while
```

4.2.3 Building a Model of the Hand

The method proposed for building a model of the robot's hand draws inspiration from the work presented in (Weise et al. 2009) and (Krainin et al. 2011). Essentially, a surfel model $\mathfrak{S} = \{\mathfrak{s}_1, \mathfrak{s}_2, ..., \mathfrak{s}_M\}$ is built and tracked, where each surfel $\mathfrak{s}_i \in \mathfrak{S}$ has position $\mathbf{v}(\mathfrak{s}_i)$, normal $\hat{\mathbf{n}}(\mathfrak{s}_i)$, radius $r(\mathfrak{s}_i)$, and colour $c(\mathfrak{s}_i)$ attributes.

The concept of a surfel is illustrated in Figure 4.4. Surfels allow the properties of an object's surface to be captured, without having to maintain connectivity information between parts of the surface. Approximating a surfel as a hexagon, allows it to be drawn quickly, and efficiently, with two quads.

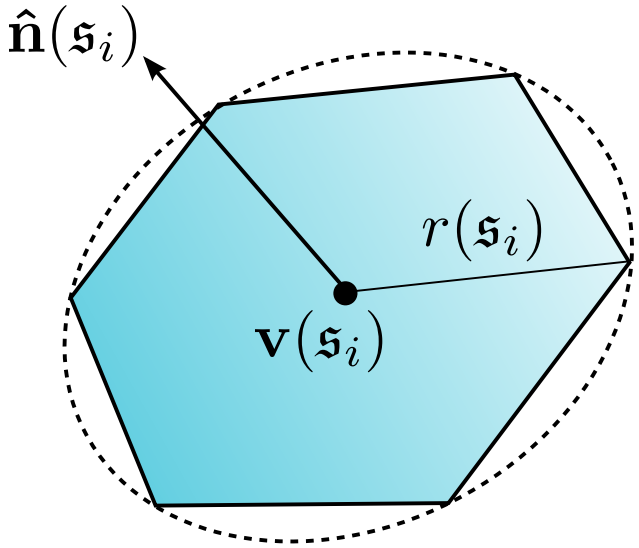


Figure 4.4: A 3D model is formed from a collection of surfels.

Following on from the previous two sections, the robot's hand is now roughly centred in its field of view. At this point a rough 2D bounding box is obtained, that describes the image extents of the hand, by using an exploratory waving motion.

Prior to building a model of the hand, the pose of the robot, and the corresponding joint angles at this stage are defined as the *initial position* for the hand model, and also for the subsequent kinematic model. To build the model of the hand, the 2D bounding box in image space is converted to a 3D bounding box in Kinect camera space. The x and y dimensions of the 3D bounding box are easily obtained from the 2D bounding box. The z dimension of the 3D bounding box is set by assuming that the depth of the hand will not be greater than the diagonal of the 2D bounding box. Once built, the 3D bounding box is used to filter out any 3D points which lie outside the box. Figure 4.5 shows a box

positioned around the points that represent the robot's hand. All points outside the box have been filtered out. The box is moved along with the hand, as it is tracked in camera space, to filter each frame as it comes from the Kinect.

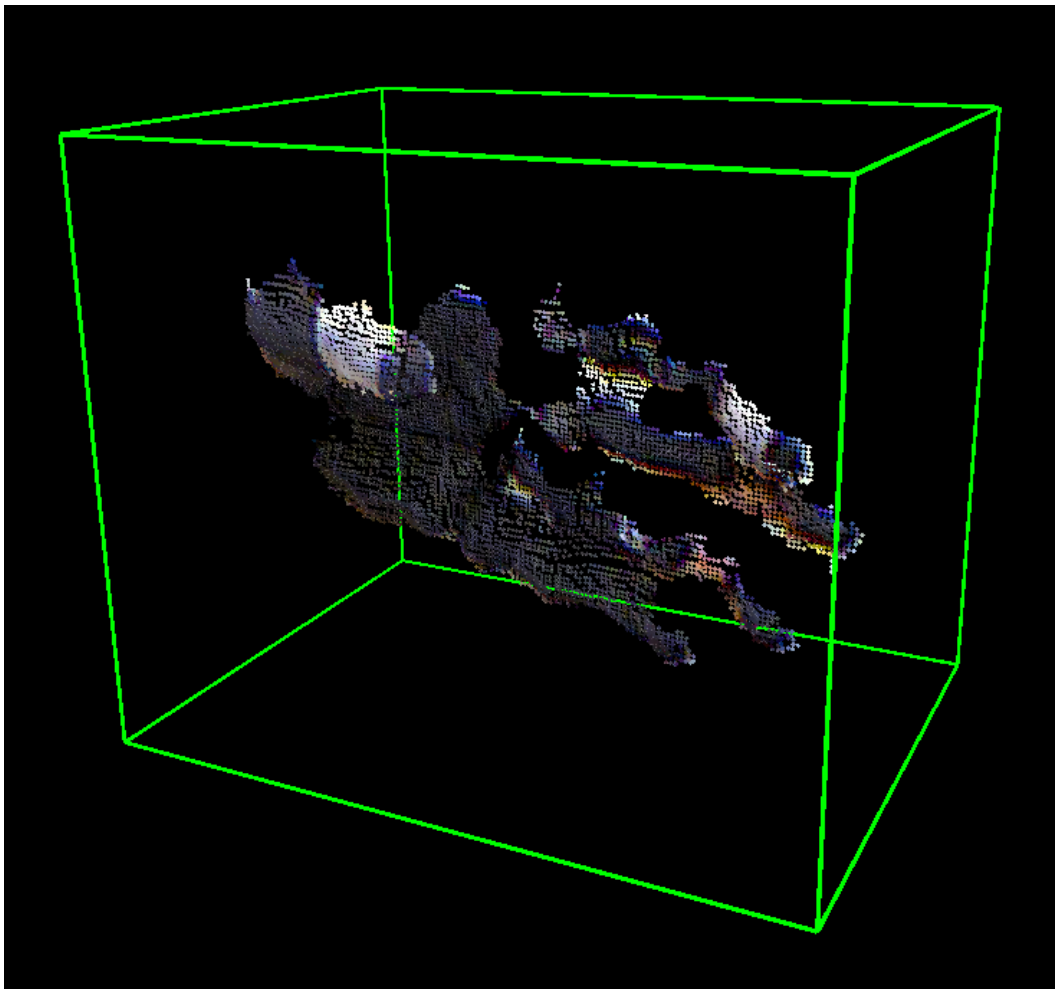


Figure 4.5: Points not part of the hand are filtered out with a box that moves with the tracked model.

Once a point cloud has been filtered, normals are estimated for the remaining points by treating the cloud as a polygonal height map and setting the normal of each point to be the weighted average of the normals of the neighbouring 4 polygons. This is a simplistic method, which gives unreliable normals close to depth discontinuities. To address this, points close to depth discontinuities are removed by treating the Kinect depth-map as a grayscale image, and applying morphological erosion. The filtering and normal estimation step gives a point cloud $\mathfrak{P} = \{\mathfrak{p}_1, \mathfrak{p}_2, ..., \mathfrak{p}_N\}$ where each point $\mathfrak{p}_i \in \mathfrak{P}$ has a position $\mathbf{v}(\mathfrak{p}_i)$, normal

$\hat{\mathbf{n}}(\mathfrak{p}_i)$, and a colour $c(\mathfrak{p}_i)$ attribute.

To initialise the model of the hand, the first point cloud is converted directly into a surfel model. So, for each point $\mathfrak{p}_i$ a surfel $\mathfrak{s}_i$ is created with the same position, normal, and colour as the point. The radius $r(\mathfrak{s}_i)$ is set as

$$r(\mathfrak{s}_i) = \frac{\mathbf{v}_z(\mathfrak{p}_i)}{\sqrt{2}l\hat{\mathbf{n}}(\mathfrak{p}_i)^T \cdot \hat{\mathbf{n}}_{image}} \tag{4.8}$$

where $\mathbf{v}_z(\mathfrak{p}_i)$ is the depth of $\mathfrak{p}_i$, l is the focal length of the Kinect, and $\hat{\mathbf{n}}_{image}$ is the Kinect image plane normal. This sets the radius for a surfel so that it roughly covers one pixel when it is observed.

After that, the robot rotates its wrist joint through 180°, in order to turn the hand in front of the Kinect. The Kinect produces new point clouds roughly every 1°, and as each point cloud comes in, the existing model is aligned with the observed point cloud using the ICP algorithm.

4.2.3.1 Point Cloud Alignment with ICP

The process of finding a transformation which aligns the existing model with the observed point cloud, involves repeatedly finding the closest point in the point cloud for each surfel in the model. A naive algorithm for finding the closest point would have a runtime complexity of $O(NM)$, which would slow the alignment process down to the point of making a real time implementation infeasible. Therefore, use is made of projective matching as detailed in (Rusinkiewicz and Levoy 2001), in order to get a quick approximation of the closest point for each surfel.

Projective matching works by first positioning the surfel model in the camera space of the observed point cloud, using the current estimate for the transformation between the model and point cloud. Then, the normal of each surfel is tested in turn, against the camera axis, to see if the surfel is visible. If it is, then the surfel is projected onto the image plane. If a point from the point cloud occupies the same pixel on the image plane, then the point is declared to be a match for the surfel. The case of multiple surfels matching the same point is handled by using a z-buffer, so that only the surfel closest to the image plane gets a match. This allows the matching to be done with a runtime complexity of $O(M)$.

Once the matches have been found, a search is made for the transformation that minimises the point-to-plane metric (Rusinkiewicz and Levoy 2001). This involves looking for the transformation that minimises the sum of squared distances between each surfel, and the plane on which its corresponding point lies. The orientation of a point's plane is obtained from its normal, which was estimated above.

Formally, this means that, on each iteration the optimal transform $\mathbf{T}_{opt}$ between the surfel model and the point cloud is

$$\mathbf{T}_{opt} = \arg\min_{\mathbf{T}} \sum_{i=1}^{M} \eta(i) \left((\mathbf{v}(\mathfrak{s}_i) - \mathbf{v}(\mathfrak{p}_{m(i)})) \cdot \hat{\mathbf{n}}(\mathfrak{p}_{m(i)}) \right)^2 \tag{4.9}$$

where $\eta(i)$ is an indicator function that returns 1 if the surfel $\mathfrak{s}_i$ has a matching point and 0 otherwise. Also, the function $m(i)$ gives the index of the point that matches the surfel $\mathfrak{s}_i$. To minimise equation 4.9 the approach taken in (Low 2004) is followed and the rotation part of $\mathbf{T}_{opt}$ is approximated as a linear transformation. This approximation is reasonable, as there will only be a small difference in the pose of the hand from one frame to the next. It also enables the efficient minimisation of equation 4.9, by allowing the use of linear optimisation, rather than non-linear optimisation.

4.2.3.2 Updating the Surfel Model

Once the surfel model has been aligned with the observed point cloud, a decision is made as to whether to merge the point cloud into the surfel model. As the hand is being rotated around the wrist axis, a point cloud is merged with the surfel model every $\varphi_{model} = 5°$. Lower values of φ_{model} give a higher fidelity model at the expense of slower tracking speed as there are more surfels to align with the incoming point clouds. The point cloud is merged into the surfel model, using a simplified form of the process presented in (Weise et al. 2009). The process used, is to first find matches between surfels and points using the projective matching procedure given above. Then either new surfels are added to the model, or else existing surfels are updated.

New surfels are created from points which do not have a matching surfel in the model. When a point is matched to an existing surfel, the depth of the surfel $\mathbf{v}_z(\mathfrak{s}_i)$ is compared to the depth of the point $\mathbf{v}_z(\mathfrak{p}_{m(i)})$.

If $\mathbf{v}_z(\mathfrak{s}_i) - \mathbf{v}_z(\mathfrak{p}_{m(i)}) < -d_{max}$ where d_{max} is a distance threshold (set to 10mm in this work), then the observed point is far behind where the surfel predicts the surface should be. The surfel is assumed to be incorrect, deleted, and replaced with a new surfel created from the point.

If $\mathbf{v}_z(\mathfrak{s}_i) - \mathbf{v}_z(\mathfrak{p}_{m(i)}) > d_{max}$ then the observed point is far in front of the surfel. This may not be a problem, as the camera may be viewing an occluding surface, so for now this case is ignored as a surfel may be added later on, in response to a view of the surface from a different angle.

Finally, if $\left|\mathbf{v}_z(\mathfrak{s}_i) - \mathbf{v}_z(\mathfrak{p}_{m(i)})\right| < d_{max}$ then it is assumed that the point is a new observation of an existing surfel. The position and normal of the surfel is updated from the position and normal of the point using a running average. If the radius

suggested by the point is smaller than the radius of the surfel, (due to an observation being made closer to the camera) then the radius of the surfel is replaced. Also, if the normal of the point is closer to the camera axis than previous observations, then it is assumed that the camera is getting a more direct view of the surface, and thus the colour of the surfel is updated as well. The process of averaging the point clouds into a surfel model has the effect of filtering out a lot of the noise, which would be present if the point clouds were used directly as the model.

Figure 4.7 shows typical changes in the built hand model that result from varying d_{max} and φ_{model}.

4.2.3.3 Tracking the Hand

After the model of the hand has been built, the system continues to track the hand by continually aligning the hand model against incoming point clouds. Figure 4.6 shows a typical surfel model of the robot's hand; built and tracked using this technique. A GPU implementation has been produced for key parts of this model building and tracking system. As such, once the hand has been detected and centred, it takes just over a minute for the system to autonomously build a 3D model of the robot's hand. It is then able to reliably give a reading for the position, and orientation of the hand, in camera space, at a rate of 15Hz. Further evaluation of the accuracy and precision of the tracking system is presented in Section 4.3.

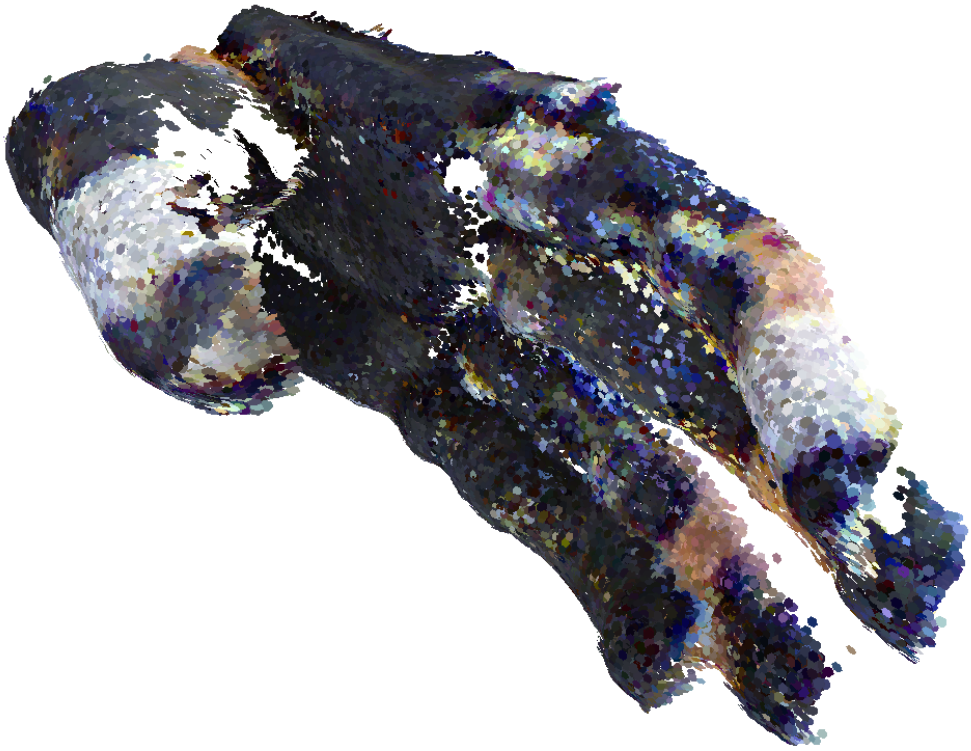
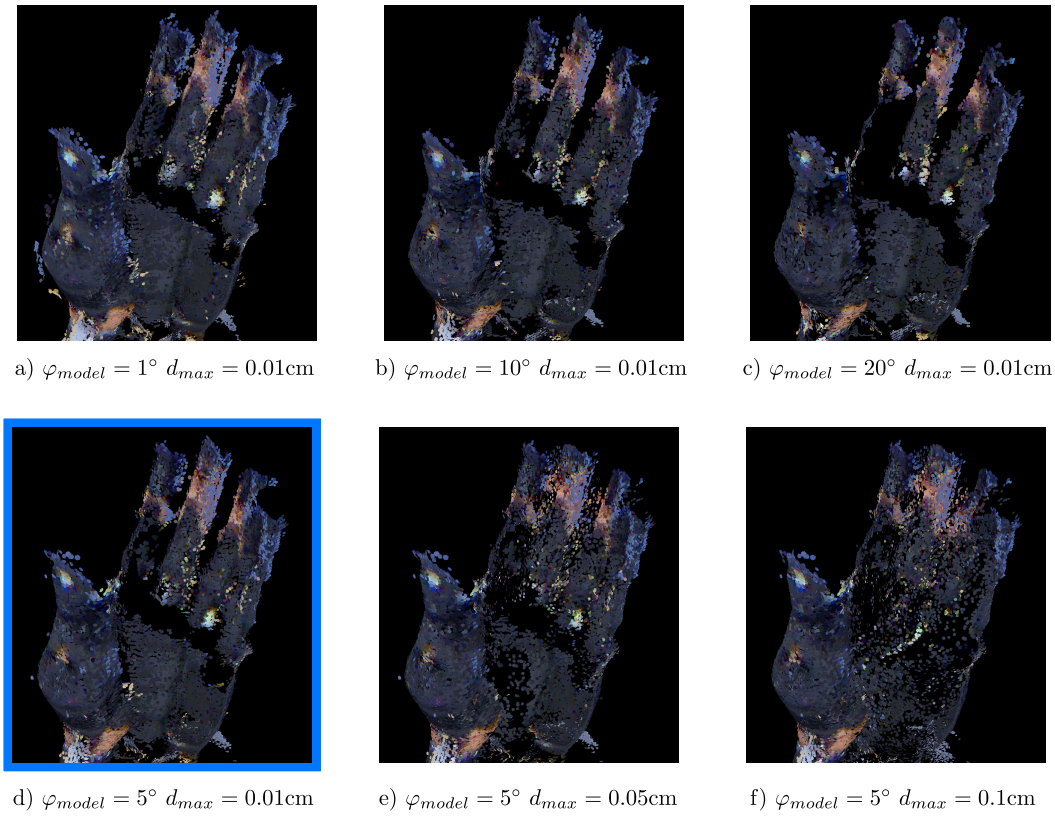


Figure 4.6: A surfel model obtained by aligning, and merging, multiple point clouds.



a) $\varphi_{model} = 1^\circ$ $d_{max} = 0.01$cm b) $\varphi_{model} = 10^\circ$ $d_{max} = 0.01$cm c) $\varphi_{model} = 20^\circ$ $d_{max} = 0.01$cm

d) $\varphi_{model} = 5^\circ$ $d_{max} = 0.01$cm e) $\varphi_{model} = 5^\circ$ $d_{max} = 0.05$cm f) $\varphi_{model} = 5^\circ$ $d_{max} = 0.1$cm

Figure 4.7: The effect of varying d_{max} and φ_{model}. Subfigures a) to c) show that as φ_{model} the angle between model updates is increased, the number of surfels used in the model decreases. Eventually if φ_{model} gets too high, then interframe tracking of the hand will fail. Subfigures d) to f) show how varying d_{max} can affect the quality of the produced model. Subfigure d) (highlighted in blue) show the parameters that subjectively gave the best models.

4.2.4 Building a Kinematic Model

Having built a 3D model of the robot's hand, the tracking system is now used to build a kinematic model of the connecting links between the Kinect and the robot's hand.

To build the kinematic model, the robot moves each of its joints back and forth through their range of motion in order to deduce their axis of rotation. The joints in the BERT robot are all revolute joints, but this system could also be used to identify prismatic joints. The order of the joints is given, but it could equally well be deduced by rotating sequences of joints in different permutations, and observing which joints have their axis moved by other joints. The position of a joint's axis will not be affected by the position of a child joint's axis, but it will be affected by

the position of a parent joint's axis. For the range of motion that the hand is visible to the camera, the motion of the hand can be tracked. Tracking the hand gives a rigid transformation, which describes how the hand model moves from the point where it is first visible, to the point where tracking is lost. To get useful information from this transformation, *Chasles' theorem* (Kajita and Espiau 2008) is used which states

> "Any rigid transformation can be represented by a rotation about a fixed axis, followed or preceded by a translation along that axis."

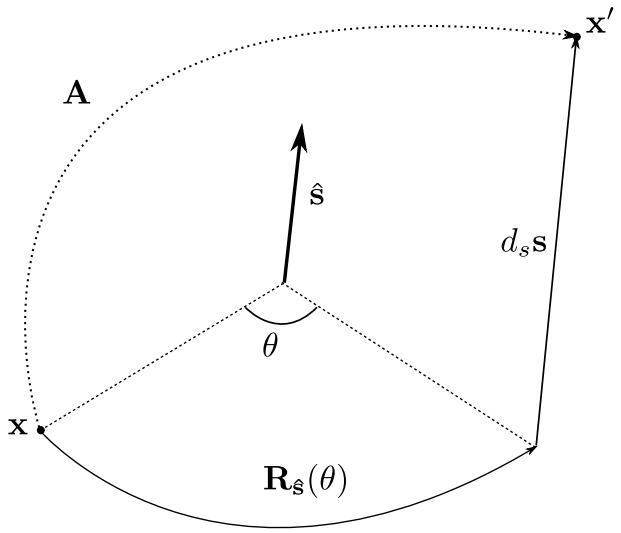


Figure 4.8: Screw decomposition of a Euclidean transformation.

To take advantage of this fact the *screw decomposition* of the tracked transformation is deduced. This gives the direction of the joint axis, and a point on the joint axis. It also gives the rotation around the axis, and the displacement along the axis. For a perfect revolute joint the screw decomposition should have only produce rotation around the axis; and likewise, a perfect prismatic joint should exhibit only displacement.

4.2.4.1 Constructing a Screw Transformation

Before the problem of finding the screw decomposition is tackled, a transformation matrix $\mathbf{T}$ must be constructed from $(\hat{\mathbf{s}}, \mathbf{s_o})$, θ, and d_s, where $\hat{\mathbf{s}}$ is the axis normal, $\mathbf{s_o}$ is a point on the axis, θ is the angle rotated around the axis, and d_s is the distance translated along the axis.

Figure 4.8 shows the component parts of a screw transformation. A screw transformation can be constructed from simpler transformations, in the sense that it can be viewed as the composition of

1. a translation -$\mathbf{s_o}$ to move the screw axis so that it passes through the origin,
2. a rotation θ around the screw axis at the origin,
3. a translation $\mathbf{s_o}$ to restore the position of the screw axis,
4. a displacement d_s along the screw axis $\hat{\mathbf{s}}$.

This gives

$$\begin{aligned}\mathbf{T} &= \begin{pmatrix}\mathbf{I} & d_s\hat{\mathbf{s}}\\ \mathbf{0} & 1\end{pmatrix}\begin{pmatrix}\mathbf{I} & \mathbf{s_o}\\ \mathbf{0} & 1\end{pmatrix}\begin{pmatrix}\mathbf{R}_{\hat{\mathbf{s}}}(\theta) & \mathbf{0}\\ \mathbf{0} & 1\end{pmatrix}\begin{pmatrix}\mathbf{I} & -\mathbf{s_o}\\ \mathbf{0} & 1\end{pmatrix}\\ &= \begin{pmatrix}\mathbf{R}_{\hat{\mathbf{s}}}(\theta) & \mathbf{s_o} - \mathbf{R}_{\hat{\mathbf{s}}}(\theta)\mathbf{s_o} + d_s\hat{\mathbf{s}}\\ \mathbf{0} & 1\end{pmatrix} \end{aligned} \tag{4.10}$$

Now $\mathbf{R}_{\hat{\mathbf{s}}}(\theta)$ can be built using *Rodrigues' rotation formula* (R. M. Murray et al. 1994)

$$\mathbf{R}_{\hat{\mathbf{s}}}(\theta) = \mathbf{I} + [\hat{\mathbf{s}}]_\times \sin\theta + [\hat{\mathbf{s}}]_\times^2(1-\cos\theta) \tag{4.11}$$

where $[\hat{\mathbf{s}}]_\times$ is the skew symmetric matrix

$$[\hat{\mathbf{s}}]_\times = \begin{pmatrix}0 & -s_z & s_y\\ s_z & 0 & -s_x\\ -s_y & s_x & 0\end{pmatrix} \tag{4.12}$$

4.2.4.2 Obtaining the Screw Decomposition

Now, given a Euclidean transform

$$\mathbf{T} = \begin{pmatrix}\mathbf{R} & \mathbf{t}\\ \mathbf{0} & 1\end{pmatrix} \tag{4.13}$$

it is possible to produce the screw decomposition. Matching the elements of (4.13) to those of (4.10) gives

$$\mathbf{R} = \mathbf{R}_{\hat{\mathbf{s}}}(\theta) \tag{4.14}$$

$$\mathbf{t} = \mathbf{s_o} - \mathbf{R}_{\hat{\mathbf{s}}}(\theta)\mathbf{s_o} + d_s\hat{\mathbf{s}} \tag{4.15}$$

As $\mathbf{R}$ is a rotation matrix, the axis of rotation $\hat{\mathbf{s}}$ is the eigenvector corresponding to the unit eigenvalue. Also, if (4.11) is expanded, it can be shown that θ is easy to find as

$$\text{trace}(\mathbf{T}) = 1 + 2\cos\theta \tag{4.16}$$

Using the dot product gives d_s, the distance travelled in the direction of $\hat{\mathbf{s}}$

$$d_s = \hat{\mathbf{s}}.\mathbf{t} \tag{4.17}$$

Now, if $\theta = 0$ it is not possible to determine a value for $\mathbf{s_o}$. Although if a prismatic joint was being tracked, it would be possible to use any of the tracked positions for $\mathbf{s_o}$. When $\theta \neq 0$, then $\mathbf{s_o}$ can be found by creating the points $\mathbf{x}'$ and $\mathbf{x}''$ from an arbitrary point $\mathbf{x}$ which is not on the axis of rotation.

$$\mathbf{x}' = \mathbf{T}\mathbf{x} - d_s\hat{\mathbf{s}} \tag{4.18}$$

$$\mathbf{x}'' = \mathbf{T}\mathbf{x}' - d_s\hat{\mathbf{s}} \tag{4.19}$$

This applies $\mathbf{T}$ to $\mathbf{x}$ and removes any translation in the direction of $\hat{\mathbf{s}}$ leaving just the rotation around $\hat{\mathbf{s}}$. The three points are used to construct intersecting planes as shown in Figure 4.9, and using these planes it is possible to calculate $\mathbf{s_o}$ as a point where the planes intersect. For the special case of $\theta = \pi$, $\mathbf{s_o}$ is set as the midpoint between $\mathbf{x}$ and $\mathbf{x}'$.

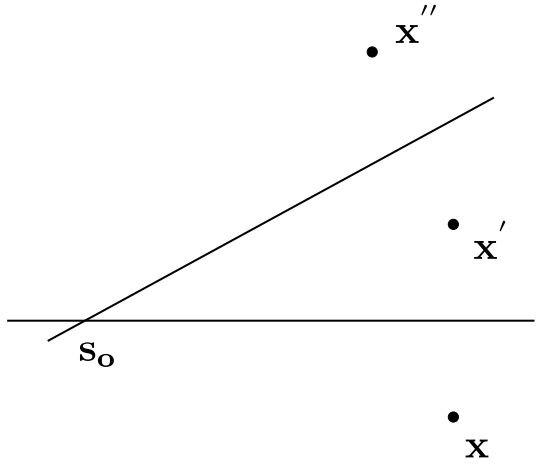


Figure 4.9: A point on the intersection of the bisecting planes is used for $\mathbf{s_o}$.

When implementing the screw decomposition, it was ensured that $\mathbf{x}$ did not lie on the axis of rotation, by testing the distances between $\mathbf{x}$ and $\mathbf{x}'$ to ensure that they were greater than 0. A maximum of 3 non co-linear points need to be tested to guarantee that a suitable $\mathbf{x}$ is found.

4.2.4.3 Constructing the Kinematic Model

The system moves and tracks each of the robot's arm and neck joints in turn in order to identify the position and orientation of the joint axes. Each joint is moved fairly slowly to ensure that the tracking system is able to reliably track the motion of the hand, and in total it takes about 20 minutes to determine the joint axes of all 9 joints (2 for the neck and 7 for the arm) between the camera and the hand.

Once all of the axes have been identified, the system has a kinematic model consisting of the position and orientation of the joint axes in camera space. To complete the model, the transform between the wrist frame, and a frame placed at the centroid of the hand model is estimated by measuring the hand pose at the initial position. The kinematic model that has been constructed is a *Zero-Reference* model as defined by (Mooring et al. 1991). It can be converted to the more common Denavit-Hartenberg parameters by using the algorithm given in (Mooring et al. 1991). Figure 4.10 gives an example of a kinematic model constructed using this method. The skeleton is built in camera space, so the robot is now able to plan movements to pick up objects in camera space.

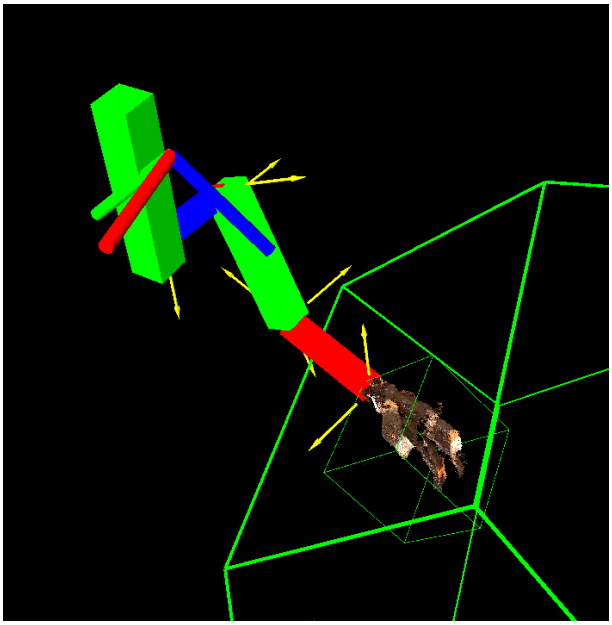


Figure 4.10: The constructed kinematic model and hand model in camera space. The yellow arrows represent the estimated joint axes used to construct the model.

4.3 Validating the System

The robotic system discussed so far is able to locate its hand in its field of view using exploratory waves. It can then centre its hand in its field of view using a simplified form of visual servoing, and once centred, it can build and track a model of its own hand. Finally, it can also make use of the tracking ability, to construct a kinematic model of its arm. Experiments are now presented which were carried out in order to assess the system performance.

4.3.1 Hand Location and Centring

The reliability of the proposed method for locating and centring the hand is explored, by starting the robot in 6 different poses, which give an initial position for the hand at various points in the robot's field of view. The hand location and centring process was run for 3 iterations, and at each step, the position of the hand was measured roughly by manually picking out a fixed point on the hand.

Figure 4.11 shows the result of this experiment. It can be seen that the process works well for a number of different starting points, with the hand brought within 20 pixels for all start points in 3 steps, and in some cases just 2 steps.

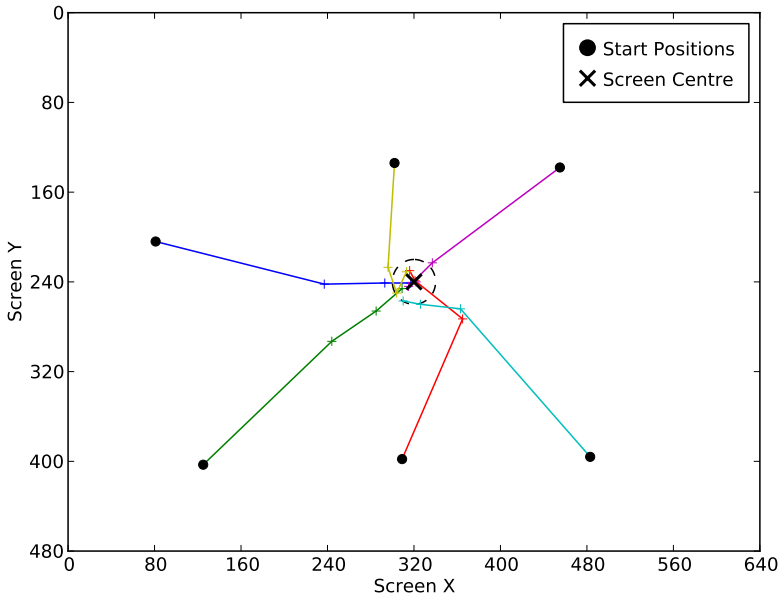


Figure 4.11: The result of running the visual servoing procedure for 3 steps from 6 different starting points.

4.3.2 Hand Tracking Accuracy and Precision

The accuracy and precision of the tracking system is assessed using the standard definitions given in (Kijewski 1999). Accuracy is defined as the average absolute error when measuring known values. Precision is defined as the standard deviation of repeated measurements of a fixed value.

Precision was the easiest quantity to assess. It was first confirmed that the measurements of the hand's pose were normally distributed, using Pearson's chi-squared test. Then, the robot's hand was moved around a 3D grid of points in camera space (see Figure 4.12), with the necessary joint angles being calculated using the constructed kinematic skeleton, and inverse kinematics. At each point, 10 measurements of the position and orientation of the hand were taken, and the precision at each point was taken as the standard deviation of the measurements.

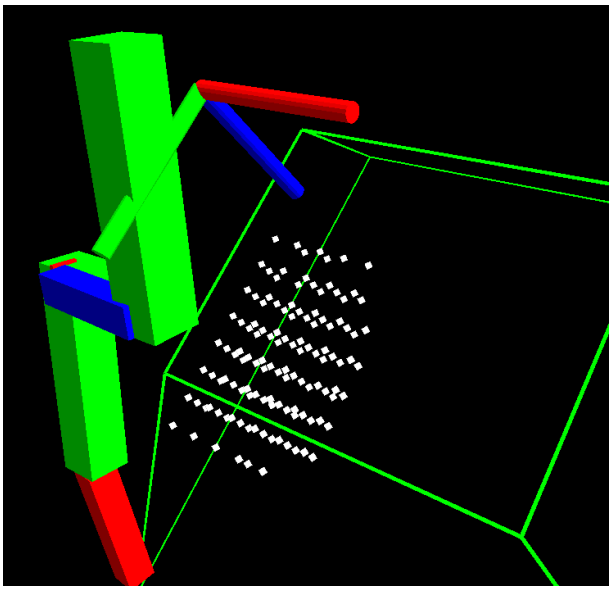


Figure 4.12: The grid of test points used for assessing precision.

The precision of the hand tracking was found to be good for a large number of the test positions, with the average translational precision, over all the test points, being in the order of 1mm, and the average rotational precision being in the order of 0.1° (see Table 4.1).

The accuracy of the hand tracking was harder to assess because it requires knowledge of the hand's true location, obtained from some more accurate measurement system. There was no easy way of measuring the position of the robot's hand when it was being moved by the robot, so instead a separate, spare robot hand was used in order to assess the accuracy of hand tracking.

Dim.	Average Precision	Best Precision	Worst Precision
x	1.14	0.27	8.94
y	1.68	0.40	9.22
z	0.31	0.06	4.23
θ_x	0.13	0.03	0.80
θ_y	0.10	0.03	0.78
θ_z	0.18	0.07	0.56

Table 4.1: The precision of the repeated pose measurements taken at each test point. Values are given to 2 d.p. with distances given in mm and angles given in degrees.

The spare robot hand was mounted on a separate base, and moved between 8 fixed test points on a table in front of the robot. Then, a coordinate system was defined on the table-top, and the positions of the test points were measured to the closest half millimetre with a ruler. The table-top was confirmed to be flat with a spirit level. The hand was moved between the test points, taking care to keep the orientation of the hand constant, and at each test point the position and orientation of the hand was measured by the hand tracking system.

It was not possible to compare the hand tracking measurements directly with the ruler measurements, because the coordinate systems were related by an unknown rigid transformation. However, as the orientation of the hand was held constant, it was possible to compare the distances between the test points. The distances between all pairs of test points measured by the hand tracking system were compared with the corresponding distances from the ruler measurements, and were found to agree with an average absolute error of just 2mm, and standard deviation of 1.8mm.

4.3.3 Assessing the Hand Model

In an attempt to quantitatively assess the quality of the hand model built by the modelling system, the hand was detached from the robot, and a high resolution model of the hand was built using a 3dMD scanner (See Figure 4.13). Using the MeshLab program (Cignoni et al. 2008), the high resolution model was then aligned using ICP with a model of the hand produced by the robotic modelling system. The Hausdorff distance from the high resolution model to the system model was calculated; again using the implementation by MeshLab. Essentially, this samples a large number of points on the high resolution model, and measures the distance from each of those points, to the closest corresponding point on the hand model produced by the robotic system.

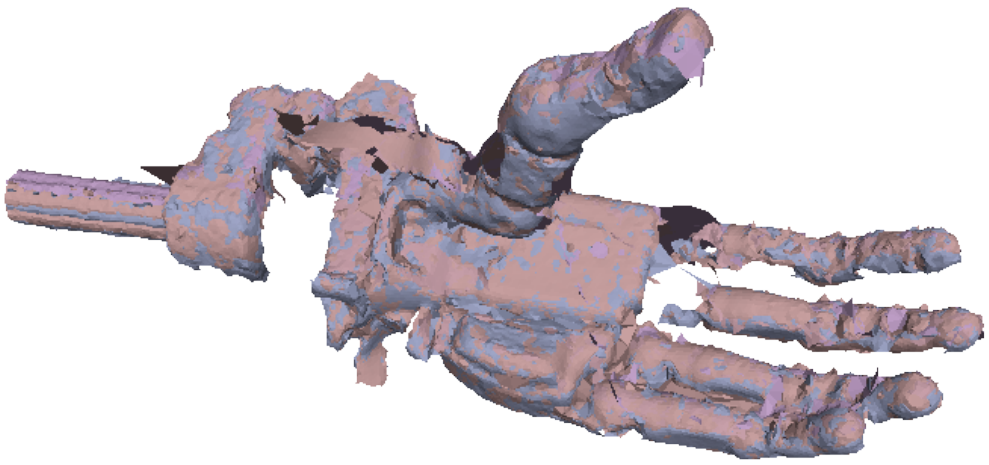


Figure 4.13: A higher resolution model of the hand captured with a 3dMD scanner.

The Root Mean Square (RMS) of these distances was found to be 4mm. Obviously, this only evaluates the accuracy of the generated hand model in terms of how well it agrees with the high resolution model. However, 3dMD quote their scanner as having a geometry accuracy of 0.2mm RMS or better. Therefore, it seems reasonable to claim that the robotic modelling system has a geometry accuracy of 5mm RMS or better.

The accuracy of the robotic model building system is good considering that it uses a low resolution depth camera in the form of the Kinect, and has the added benefit that the hand does not have to be detached from the robot, in order to build the model. Also, the results from Section 4.3.2 suggest that any inaccuracies in the model do not appear to degrade the tracking performance significantly.

4.3.4 Assessing the Kinematic Model

The accuracy of the estimated kinematic model is evaluated by using it to predict the position of the hand in camera space for a number of joint configurations. These can then be tested experimentally, and the better the agreement, the greater the accuracy of the kinematic model. To carry out this test, the hand was again moved around the grid of test points shown in Figure 4.12, and the predicted hand positions were compared against the measured hand positions.

Figure 4.14 shows histograms of the absolute errors between the predicted and measured positions and orientations. The positioning accuracy is good enough to

enable the robot to move its arm with confidence around camera space, and in large areas of camera space it seems as if the accuracy is good enough ($<$10mm) for the robot to be able to pick objects up. However, the accuracy of the kinematic model could also be improved, as a number of joint configurations have quite large errors, especially in the Cartesian z axis. Also, there appears to be a small systematic rotational error. It seems likely that these inaccuracies are due to a combination of errors in the measured positions of the joint axes, and unmodeled effects such as joint compliance.

4.4 Summary

With the advent of affordable depth cameras, it becomes far easier to give robots true 3D vision, which enables them to build detailed models of themselves, their environment and the objects that they interact with. In this chapter, a system has been presented in which a robot turns a depth camera on itself, and is therefore able to progressively build a model first of its hand, and then of the kinematic structure that links its camera to its hand.

The system is able to identify and locate the robot's hand using exploratory motions, and therefore, it is able to build the 3D and kinematic models with very little prior information.

There is nothing new about kinematic identification *per se* Bennett et al. (1992), Stone et al. (1986), and other researchers have already programmed robots augmented with fiducial markers to build models of themselves (Hart and Scassellati 2011), (Hersch et al. 2008). However, the contribution of this work is to show how a depth camera can remove the need for fiducial markers, by building a model of the robot's end effector, and tracking that directly. The use of exploratory motions in this context, to detect and centre the robot's hand prior to model building, is also novel, and it reduces further the *a priori* knowledge which must be supplied to the robot by a system designer.

A system such as the one presented in this chapter, would be of potential use for a robot designed for service in a domestic environment, or for an assisted living application. This is because such a robot may have to work for extended periods without being serviced, and so, the ability to rebuild its kinematic model in response to a change, or a degradation, in its hardware, could greatly improve the reliability of the robot.

The system would also be very useful in situations where a new kinematic model had to be built for a robot, but where it would be difficult, or impractical for a human to gain access to take the required measurements. For example, this may be because the robot is in a remote or hazardous location.

The next chapter considers the case where a robot has a kinematic model and

is being powered on. In that situation the robot must be *homed* so that the pose of the kinematic model corresponds to the actual joint angles of the robot. There are a number of established ways of achieving this, but the next chapter presents a novel method of homing a robot using vision. The presented method makes use of exploratory actions to allow the robot to find out about its current pose in a similar way to which exploratory actions were used in this chapter.

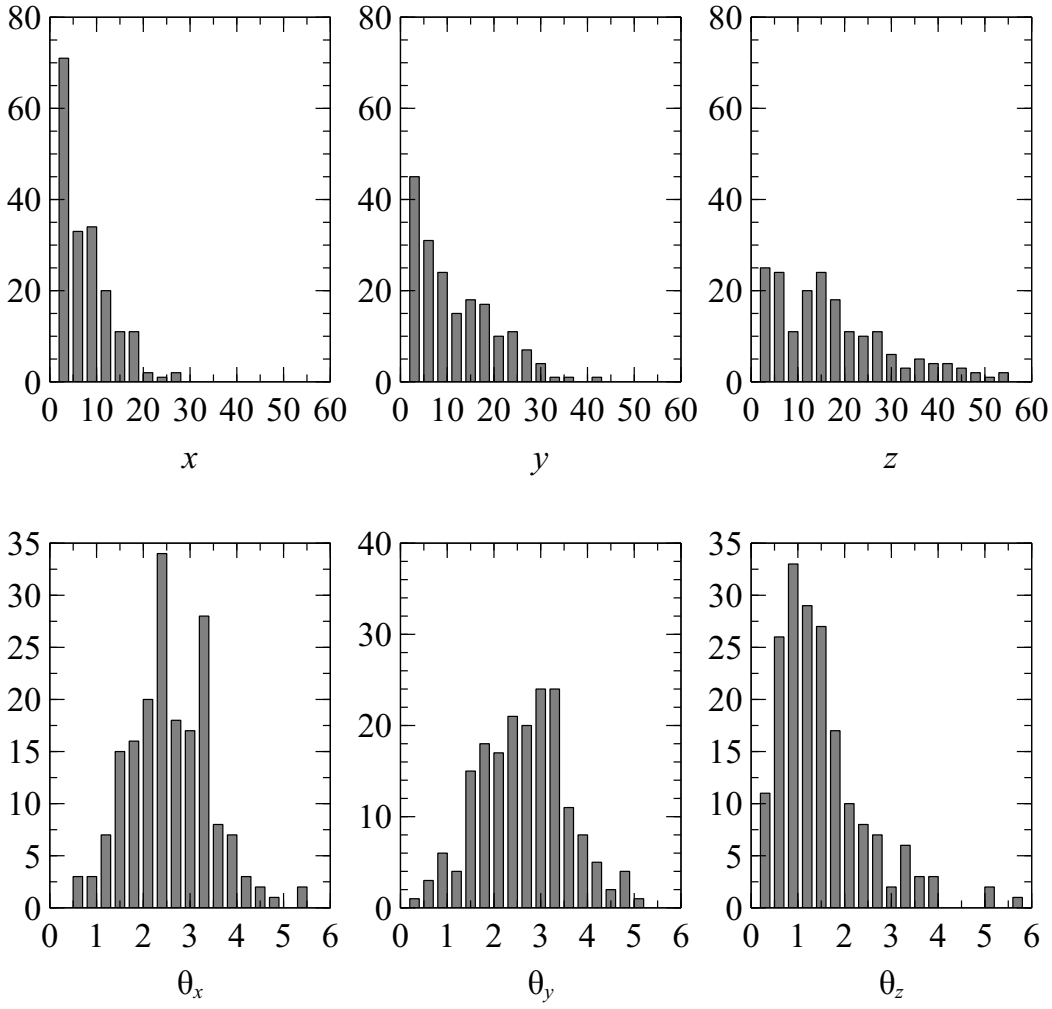


Figure 4.14: Error histograms showing the distribution of absolute errors in translation (x,y,z) and rotation ($\theta_x, \theta_y, \theta_z$) between the predicted and measured positions of the hand in camera space when testing the constructed kinematic model. Translation errors are given in mm, and rotation errors are given in degrees.

Chapter 5

Visual Homing

In the last chapter it was shown how a robot could use exploratory actions in order to generate a kinematic model of its body with only a small amount of initial information.

In order to make use of a kinematic model however, a robot needs to know the current angle of its joints relative to a starting or *home* position. During operation, the joint angles relative to the home position can be tracked with incremental joint encoders. However if power is lost, or if the system shuts down, the robot may lose track of its joint angles, because the joints may move before power is restored. For example, incremental encoders give an angular reading relative to the position of the joint at power on, and so movements using these angles will be inconsistent, unless the robot is powered on at exactly the same starting position each time it is used.

Modern robotic systems employ a variety of methods for coping with a lack of knowledge about the home position. For robots having only incremental encoders, the simplest solution is to always start the robot from a known home position. However, it can be tedious to manually position the robot and it may be hard to do this with a high level of accuracy. Another method is to install homing switches on the robot, or to have hard mechanical end-stops that robot joints can be driven against, so that a control program can automatically seek and find the home position. Mechanical end-stops may not always be available however, i.e. on joints designed for continuous rotation, and homing switches increase system complexity and may fail.

Absolute encoders or resolvers solve the problem of losing the position relative to a home position, but absolute encoders can often be both more expensive and have lower resolution than incremental encoders. This is because incremental encoders can be implemented cheaply with hall sensors, and get a resolution increase from any reduction gearing which may be present in the joint.

In this chapter, an alternative approach to homing is presented. Instead of

using limit switches, or absolute encoders to home a robot's joints, a visual homing system is presented instead. The visual homing system presented in this chapter enables a robot to autonomously find its home position using visual information from a depth camera, such as a Microsoft Kinect.

By using visual information to home a robot, it is possible to forgo the cost of alternative systems, such as limit switches, and also to eliminate the possibility that the robot would be unable to home itself if the limit switches were to fail. It is envisaged that this system would find use on service robots in the home where a robot may only have incremental encoders in order to keep costs low, but at the same time is likely to have a depth camera available as a cheap and flexible sensor modality.

The visual homing system presented in this chapter is also useful in the sense that it could also act as a backup system on a robot, where a more traditional homing system, such as limit switches was installed, and thus increase the reliability of the robot.

The system seeks to find the home position for a known kinematic model. The pan and tilt joints in the neck of the robot are first homed by locating an Augmented Reality (AR) marker board, fixed to the base of the robot. After this, the hand of the robot is located in the robot's vision system by using exploratory motions. By tracking the movement of the hand, using depth data from the Kinect, it is possible to measure the position and orientation of the robot's wrist frame in Kinect camera space. At this point the system then uses inverse kinematics to identify possible joint configurations which would match the observation.

As the inverse kinematics of the robot may give rise to multiple possible joint configurations for a given pose of the wrist frame in camera space, a way is needed of choosing between possible joint configurations to identify the correct one. This is done by having the robot move its hand to different points in its field of view in order to take more measurements of its wrist frame. These extra measurements are combined with the initial wrist pose in a kinematic calibration step, which seeks to find the set of joint angles which best explains the observed wrist frame measurements. Finally, the estimated joint angles can be compared to joint angles measured by the robot's incremental encoders, in order to find the robot's offset from its true position. This effectively homes the robot.

Visual homing of a humanoid robot's joints does not seem to have been the subject of much research so far. Jin and Xie (2000) describe a system that performs visual homing by detecting and tracking a number of feature points on a robot's arm. They do not give a quantitative evaluation of the accuracy of their system however, and so it is not clear how robust their system would be when viewing the feature points from oblique angles or in the presence of varying illumination. Santos et al. (2010) give an elegant method that uses homography estimation to home the panning joints in the pan-tilt joints of an iCub's head. However, as this

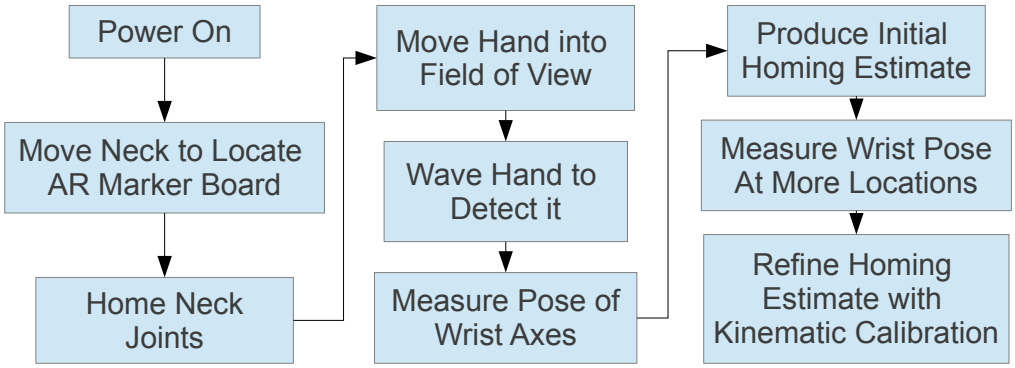


Figure 5.1: An overview of the visual homing process.

method seems to be restricted to pan-tilt joints that move a camera, it would not be suitable for homing joints in a robot's arms.

The main contribution of this chapter is to present a novel visual homing system that uses motion, and the data from a depth camera, in order to achieve reliable homing of an upper torso humanoid robot. Another contribution made is to explicitly consider how the joint limits of the robot, and uncertainty about the starting position of the robot, restrict the areas of joint space in which the robot can move whilst homing itself.

An important point to make is that, whilst the use of the system is described for an upper torso humanoid robot, there is no reason why the method could not be applied to a variety of other robotic systems.

The rest of the chapter proceeds as follows. Section 5.1 presents the method for performing visual homing, and Section 5.2 describes the experiments that were carried out to validate the system and to determine its accuracy. Finally, Section 5.3 discusses the system performance and ideas for future work.

5.1 Visual Homing

The method for Visual Homing is outlined in Figure 5.1. The robot when turned on, first homes its neck motors by locating an AR marker board fixed to its base. It then proceeds to move its arm, to bring its hand into the field of view of its camera. The hand is identified through the use of exploratory waving motions, which allows the robot to distinguish its hand from the background. Once the hand has been identified, it is possible to use the Iterative Closest Point (ICP) algorithm (Besl and McKay 1992) to track small movements of the hand in 3D camera space. This ability means that the current position and orientation of the robot's wrist axes can be measured in camera space. Being able to measure the position and orientation

of the wrist axes exposes the current pose of the wrist frame. At this point, inverse kinematics is used to find a feasible pose for the robot which could give rise to the observed wrist frame.

An alternative to measuring the pose of the wrist frame would be to measure the pose of the robot's hand, as the centre of the robot's hand is related to the wrist frame by a fixed 3D transformation. The measurement could be performed using 3D object pose estimation techniques, such as those presented by Hinterstoisser, Cagniart, et al. (2012). However, it was found that using motion allowed the pose of the wrist frame to be measured more reliably.

If there was just one possible set of joint angles, which could give rise to an observed pose for the wrist frame in camera space, then at this point the robot would be homed. The joint angles read from the incremental encoders could simply be compared with the joint angles from the inverse kinematics, and this would give the necessary angle offsets needed to home the robot. Unfortunately, with a redundant kinematic chain such as the arm of the robot considered here, there may be infinitely many poses which could give rise to a particular wrist frame measurement in camera space. Therefore, more measurements of the pose of the wrist frame are taken, and the estimate of the joint angles is refined using kinematic calibration.

5.1.1 The Exploratory Joint Space

Making use of visual homing carries with it some level of risk, in that if a joint is powered on too close to its joint limit, then the exploratory motions needed to perform visual homing may take the joint beyond its range of motion. Depending upon the design of the robot, this may cause physical damage. For example, on the BERT robot there are some joints where movement beyond safe limits will cause cables to break. This risk is mitigated by assuming that each joint starts within a certain distance α of its true zero position. This is reasonable because it is not an onerous task to make sure that the robot is roughly in a pose close to the true zero position before start up. The greater challenge is ensuring that the robot is finely homed, so that it can do useful work, and that problem is addressed by this chapter.

Assuming that a joint is subject to limits that confine it to the angular range $[\theta_{low}, \theta_{high}]$ where, without loss of generality, $\theta_{low} \leq 0$ and $\theta_{high} \geq 0$, knowing that the initial zero angle of the encoder is within α degrees of the real zero of the encoder means that the joint limits are reduced to $[\min(\theta_{low} + \alpha, 0), \max(\theta_{high} - \alpha, 0)]$.

The initial homing error to which the robot is subject to, is defined as the homing *offset*. This emphasises the fact that in attempting to home the robot, the quantity that is being sought is the distance that the robot's joints were from their home positions when the robot was powered on. An offset $\mathbf{o}$ is a vector in $\mathbb{R}^N$

Figure 5.2: The AR marker board used to home the neck joints.

where $\mathbf{o} = [\alpha_1, \alpha_2, ..., \alpha_N]^\intercal$ and N is the number of joints to be homed.

The initial distances α_i, may be different for each joint. The only requirement is that the resulting limits should provide enough movement to locate the AR marker board used for homing the neck, and enough movement so that the hand can be brought within the field of view of the robot's camera. Once the robot has been homed, the joint limits can be restored to their unrestricted form, and the possible range of motion of the robot is extended.

5.1.2 Homing the Neck Joints

The pan and tilt joints in the neck of the robot are homed by locating an AR marker board in the robot's field of view. An AR marker board is a collection of AR markers arranged in a grid. The ArUco library (Munoz-Salinas and Garrido-Jurado 2013) is used to locate and track AR markers. By looking for a collection of AR markers, the detection and localisation process is made more robust. Figure 5.2 shows the AR marker board used in this work. It is assumed that for a pair of pan and tilt angles, θ_{pan} and θ_{tilt}, it is known what the image coordinates of the centre of the AR marker board (x_b, y_b) should be.

Upon powering on, the robot performs a search from left to right and top to bottom in the exploratory joint space of its two neck joints, in order to locate the AR marker board. When the AR marker board is found, the robot adjusts its joint angles in order to bring the marker board to the recorded position (x_b, y_b) in the image space of the Kinect. When the marker board has reached this position, the

angles of the neck joints are known, and therefore, the neck joints of the robot are successfully homed.

5.1.3 Making the Hand Visible

In order to be able to see its hand, the robot must position its joints so that its hand lies within the field of view of its camera. If the robot was homed, then preprogrammed positions could be used to get the hand into the field of view. However, homing errors may mean that these preprogrammed positions will fail to bring the hand into the robot's field of view. One option for finding a position where the hand can be seen, is to randomly pick positions in the exploratory joint space. The question to answer then is, how long on average would the system have to wait until a visible position was found?

In order to answer questions about the workspace of the BERT robot, sampling methods similar to those described by Zacharias et al. (2007) were used. Zacharias et al. (2007) built a 3D representation (which they called a *capability map*) of the space close to a dual arm humanoid robot which described how easy it was for the robot to reach items in that space. This was done by uniformly drawing thousands of samples from the joint space of the robot and building up a reachability score in the robot's Cartesian workspace.

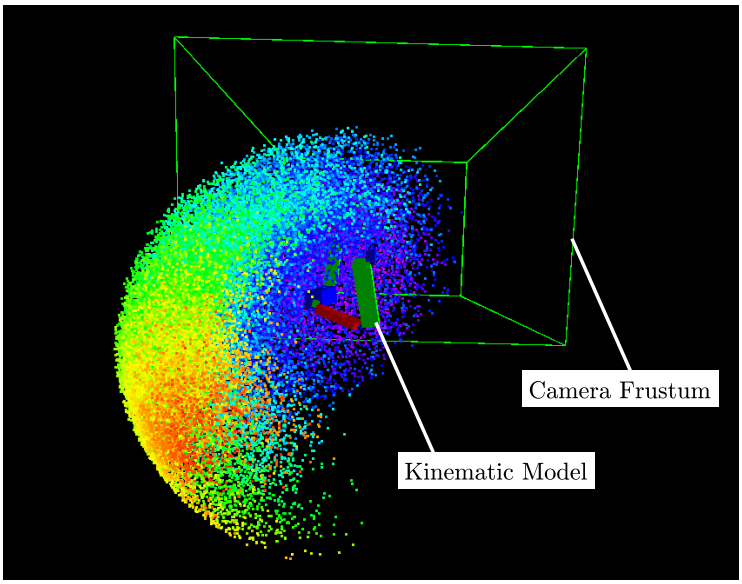


Figure 5.3: The coloured points in this image show the position of the wrist in camera space for 50,000 random positions in joint space. The green camera frustum represents the space viewable by the camera, and it can be seen that the majority of possible wrist poses fall outside this frustum (and will therefore not be visible).

Following this idea, Figure 5.3 shows the result of sampling $50,000$ positions in the joint space of the BERT robot, and displaying the resulting position of the centre of the robot's hand in camera space. This clearly shows that for the majority of joint configurations, the hand will fall outside the frustum of the camera in camera space, and will therefore not be visible to the robot. In fact, the proportion of points that fall within the camera frustum is approximately 10%, and so, on average 10 random positions would need to be chosen before seeing the hand.

Having to try 10 positions in order to see the hand, would clearly greatly extend the time needed for homing. Therefore, a strategy is needed that can increase the chance of seeing the hand. The problem can be considered as one of trying to pick the smallest set of poses Ψ, which give the best chance of seeing the hand.

This problem is formalised by defining an exploratory pose ψ as an element of $\mathbb{R}^N$ where $n \in N$ gives the actuator index, and each dimension is constrained to the appropriate exploratory range. To evaluate a pose, M homing offsets are picked $\mathbf{o}_i$ at random, from the space of possible homing errors. The cost of a set of poses $poseSetCost(\Psi)$ is defined as

$$poseSetCost(\Psi) = 1.0 - \frac{1}{M}\sum_{i=1}^{M} isVisible(\Psi, \mathbf{o}_i) \tag{5.1}$$

where $isVisible(\Psi, \mathbf{o}_i)$ is an indicator function that returns 1 if the robot's hand is visible to the robot for any of the poses $\psi \in \Psi$ when added to the offset $\mathbf{o}_i$, and 0 otherwise. To be robust, the hand is only declared to be visible if its predicted position is at least m pixels away from the edge of the screen. The value of $m = 30$ is used for the Kinect camera mounted on the BERT robot, which has a resolution of 640x480.

Equation 5.1 gives an estimate for the proportion of offsets that would not be visible at any of the poses in a pose set Ψ. The accuracy of this estimate depends on the size of the value M. This method is performing Monte Carlo integration, and so the error in the estimate decreases with rate $\frac{1}{\sqrt{M}}$ (Caflisch 1998). The estimates for this work were obtained using $M = 10,000$.

To find the set of poses which minimise Equation 5.1, use is made of Particle Swarm Optimisation (PSO) (Kennedy and Eberhart 1995), simply because it gave good results quickly. The process used was to start with a single pose, and then to add an extra pose each time PSO converged.

Table 5.1 shows the best pose set that was found, whilst varying the size of the pose set. It was found that just 3 poses were enough for the robot to be able see its hand for the vast majority ($\sim 99.47\%$) of all possible offsets. Things could be improved further with more poses, but returns quickly diminish.

Therefore once the neck joints are homed, the robot moves between the 3 poses in its exploratory pose set, ordered so that the most likely pose for seeing the hand is

Number of Exploratory Poses	Visible Offsets (%)
1	89.52
2	95.34
3	99.47
4	99.98
5	99.98

Table 5.1: The best observed visibility of joint offsets for exploratory pose sets of different size.

visited first, and attempts to find the hand at each pose. Assuming that all possible homing offsets are equally likely, then the robot is highly likely to find its hand at one of the poses. If the hand is not found at any of the poses in the pose set, then the system falls back on the method of picking random points in the exploratory joint space, and so the hand will fall within the robot's field of view eventually.

5.1.4 Finding the Hand

When the robot reaches an exploratory pose, the method presented in the last chapter in Section 4.2.1 is used in an attempt to find the hand. Briefly this involves passing a sine wave through the wrist joint of the robot and then using Normalised Cross Correlation (NCC) to find areas of the image from the robot's camera where the optical flow looks a lot like the sine wave.

5.1.5 Locating Joint Axes

If the hand is found then this gives the position of the hand in camera space and gives a rough bounding box which can be applied as a filter to the point cloud obtained from the depth camera, leaving only points from the hand. Using this knowledge of which points make up the hand, it is possible to track the movement of the hand in camera space by using the ICP algorithm. The ICP algorithm, introduced in works by Y. Chen and Medioni (1991) and Besl and McKay (1992) is a well known algorithm for aligning 3D point clouds, which gives good results providing that a reasonably accurate, initial estimate of the 3D transformation is available. In this case, the fact that the robot's actuators can move in small steps means that the identity transform can be used as the initial transformation when estimating the 3D transformation of the hand between frames.

Briefly, given two point clouds $\mathfrak{P} = \{\mathfrak{p}_1, \mathfrak{p}_2, ..., \mathfrak{p}_O\}$ and $\mathfrak{Q} = \{\mathfrak{q}_1, \mathfrak{q}_2, ..., \mathfrak{q}_O\}$, a single step of the ICP algorithm involves finding the optimal transformation $\mathbf{T}_{opt}$

such that

$$\mathbf{T}_{opt} = \arg\min_{\mathbf{T}} \sum_{i=1}^{O} \left\| \mathbf{T}\mathbf{v}(\mathfrak{p}_i) - \mathbf{v}(\mathfrak{q}_{match(i)}) \right\|^2 \tag{5.2}$$

where $\mathbf{v}(\mathfrak{p}_i)$ gives the 3D position of a point, and where $match(i)$ gives the index of a point in $\mathfrak{Q}$ which is closest to a given point $\mathfrak{p}_i$. This step is iterated to convergence, thus giving the ICP algorithm its name. This gives the ability to track the movement of the hand, and in turn, to estimate the position and orientation of the wrist joint axes. This is done by moving an axis actuator by a small angle ϕ and measuring the transformation $\mathbf{T}_\phi$ that is induced by the movement.

At this point it is possible to once more use *Chasles' theorem* (Kajita and Espiau 2008) and apply the Screw Decomposition (see Section 4.2.4.2) to obtain $\mathbf{s_o}$, $\hat{\mathbf{s}}$ and $\hat{\phi}$ from $\mathbf{T}_\phi$; where $\mathbf{s_o}$ is a point on the axis of rotation, $\hat{\mathbf{s}}$ is the direction of the axis of rotation, and $\hat{\phi}$ is the measured angle of rotation.

Ideally, $\hat{\phi}$ should match ϕ. It may be that $\hat{\phi} = -\phi$, as a rotation around an axis is equivalent to the negative rotation around an axis pointing in the opposite direction. In this case, the direction of $\hat{\mathbf{s}}$ is reversed. If however, the absolute value of $\hat{\phi}$ differs too much from that of ϕ, then it is assumed that the measurement of the axis has failed and the measurement is redone.

This technique is used to measure two of the robot's wrist axes. These axes are sequential in the kinematic model, they intersect, and they are practically orthogonal. It is therefore possible to use the cross product to find a mutually orthogonal axis and so measure the position and orientation of the wrist frame controlled by the last wrist axis.

5.1.6 Kinematic Calibration

The final step in the visual homing process is a kinematic calibration step. This uses a set of measurements for the pose of the wrist frame to estimate the true joint angles of the robot.

It is possible to estimate the joint angles of the robot using just one measurement of the wrist pose. Measurement noise can cause this estimate to be inaccurate however, so nearby poses are chosen at random, and further measurements are taken of the wrist pose are taken to give a total of K measurements.

Each measurement $k \in K$ provides a vector of joint angles $\boldsymbol{\theta}_k$ in exploratory joint space, along with the measured pose $\mathbf{x}_k$ of the robot's wrist in camera space. A search is then conducted to find the optimal offset $\mathbf{o}_{opt}$ such that

$$\mathbf{o}_{opt} = \arg\min_{\mathbf{o}} \sum_{k=1}^{K} error(\boldsymbol{\theta}_k + \mathbf{o}, \mathbf{x}_k) \tag{5.3}$$

where $error(\boldsymbol{\theta}_k + \mathbf{o}, \mathbf{x}_k)$ gives the error between the wrist pose predicted by a given offset and the actual measured wrist pose. A variety of functions could be used as the error function. For this work the chosen function was the square of the Euclidean distance between the predicted wrist frame position and orientation, and the measured position and orientation. Equation 5.3 is non-linear as it makes use of the forward kinematics of the robot, which are non-linear. This equation is minimised using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method (Nocedal and Wright 2006), coupled with random restarts to avoid local minima.

5.2 Experiments

A number of experiments were conducted to explore the performance of the visual homing system, both in simulation and on the BERT robot. In simulation, it was possible to test how well the system performed for a large number of randomly generated joint offsets. This gave some idea of the convergence properties of the algorithm throughout the joint space. Two aspects of the algorithm which were explored in simulation were (a) the effect of varying the number of extra wrist poses that were gathered when trying to refine the initial guess, and (b) the effect of noise when measuring the position and orientation of the wrist frame in camera space. When exploring a particular set of parameters for the system, 100 random offsets were generated for the robot and then the proportion of the offsets which were correctly identified to within 1° for all joints was recorded. Figures 5.4 and 5.5 show the results that were obtained from the simulation. The graphs show both the result of varying levels of noise when measuring the pose of the wrist frame, and varying the number of wrist measurements which were taken for use in the kinematic calibration step. Figure 5.4 also provides justification for using a separate homing step for the neck joints. Initially, an attempt was made to home the entire kinematic model by just observing the wrist frame in camera space. However, it was found that as more noise was introduced to the wrist frame measurements, the number of different wrist poses that needed to be measured increased to an impractical level. Figure 5.5 shows that if the neck joints have already been homed, then the kinematic calibration step requires much fewer wrist pose measurements in order to converge to the correct joint angles.

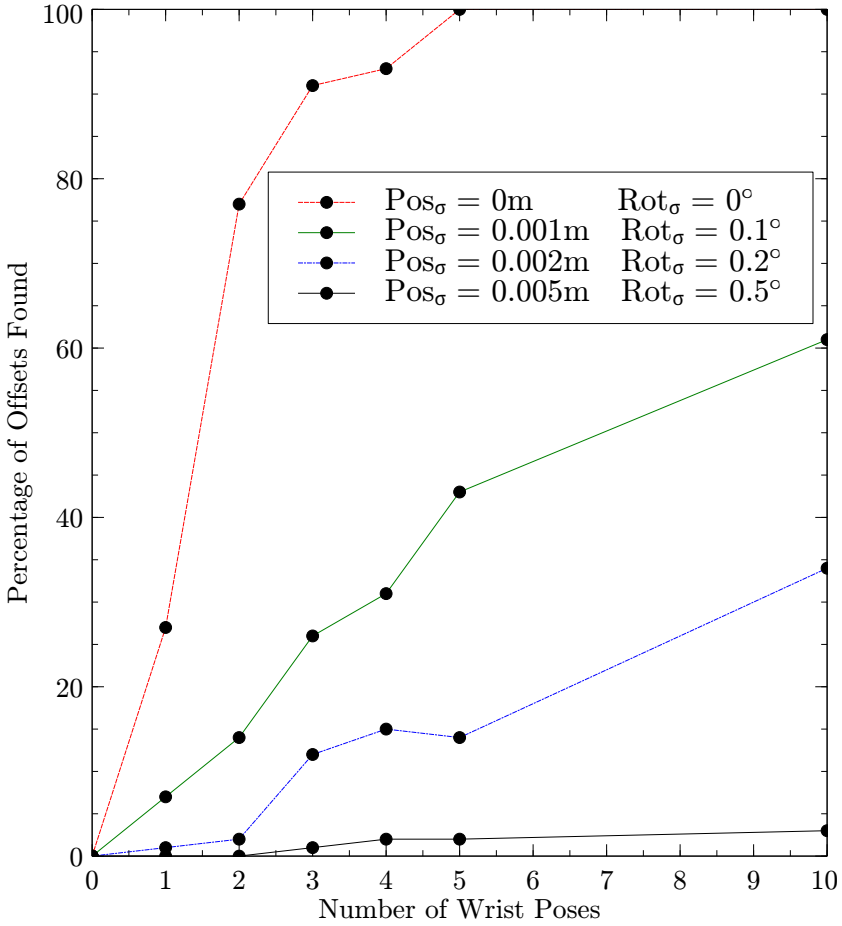


Figure 5.4: The effect of varying the number of measured wrist poses in simulation. Different lines reflect the results that were obtained for different levels of zero mean Gaussian noise, added to the simulated measurements of the wrist frames position and orientation. These are the results for homing the entire skeleton (including neck joints) by locating the hand.

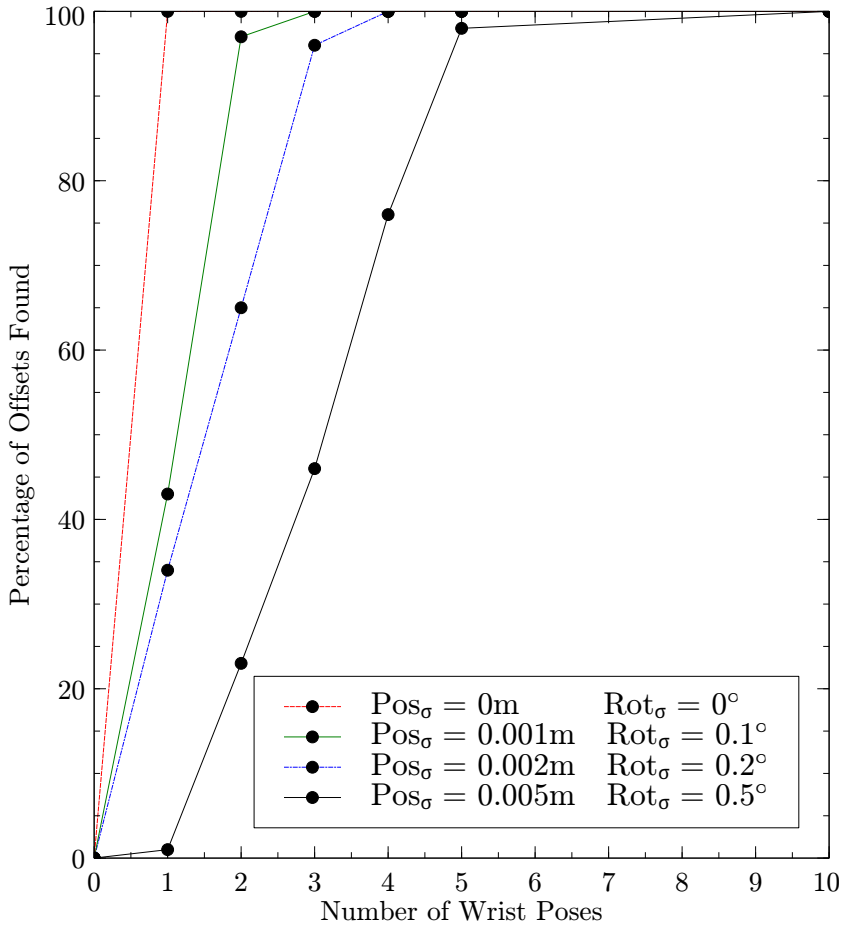


Figure 5.5: Also showing results from simulation - much better results were obtained when the neck joints were homed separately before attempting to home the rest of the skeleton.

The system was also implemented and evaluated on the BERT robot. From turning the robot on, it is possible to home the robot using the system in approximately 10 minutes. This is not especially quick, but assuming a robot is only powered on or off a few times a day it is not very significant. Also there is scope to optimise the system to reduce the homing time further.

To evaluate system performance on the BERT robot, the robot was manually homed by carefully positioning its joints prior to start up. This meant that the true joint angles were available, as they were tracked by BERT's incremental encoders. A number of homing errors were then simulated by manually generating joint offsets, allowing both the observation of how well the system was able to identify these offsets and the evaluation of the performance of the system. In a trade off between the accuracy of the result and the time required for the homing method,

Maximum Joint Error (°)	Number of Offsets
0.5	23
1.0	74
2.5	92
5.0	100

Table 5.2: Grouping of 100 random offsets in terms of the accuracy to which they were recovered by the visual homing system for the physical BERT robot.

the system was set to gather $K = 5$ measurements of the wrist pose for use in the kinematic calibration step.

For the experiments on the BERT robot, a kinematic model generated using the techniques in Chapter 4 was used, and the neck joints were homed separately as described in Section 5.1.2.

Table 5.2 shows the results of testing the system with 100 randomly generated offsets and classifies each in terms of the accuracy that was achieved. This shows that for 92% of the tested offsets, the visual homing system was able to recover the offset with a maximum absolute error of not more than 2.5°, and all of the test cases were homed successfully to within 5°.

5.3 Summary

A system has been presented which allows a robot to home itself using visual information from a depth camera coupled with exploratory motions that allow it to measure the pose of its wrist frame in camera space. The homing process can be carried out autonomously in a relatively short period of time provided that the robot has been roughly homed before being powered on. In the implementation on the BERT robot, joints can start up to 20° to 30° away from the home position, depending upon the joint.

It is not possible to home the robot from a completely arbitrary position, as with no knowledge about a joint's position, the robot may accidentally move a joint beyond its range of motion and thus damage itself. However by explicitly defining and considering an exploratory joint space for the robot, it is possible to ensure that the robot does not damage itself whilst performing the exploratory moves needed to home itself.

It would be preferable if the neck joints of the robot did not have to be homed in a separate step, prior to homing the rest of the robot. However tests in simulation

showed that without this step, it was not possible to determine the robot's pose unambiguously in the presence of noise.

Tests on a real robot show that the method is reliable and gives reasonably accurate results.

In the next chapter the focus moves out from considering the body of the robot, and expands to look at objects in the world around the robot. The next chapter looks at how a robot can build a model of a previously unseen object. It also shows how interacting with an object enables a robotic system to obtain more information about the object than would otherwise be available by just passively observing the object.

Chapter 6

Hinge Exploration for Building Articulated Object Models

So far, this thesis has looked at constructing an incremental, layered approach, that a robot can take in order to build a kinematic model of itself. The robot can perform actions to first locate and identify its hand, or end effector. It can then build a 3D model of its hand which it can track, and this in turn allows the robot to perform exploratory motions with its arm in order to build a kinematic model of its arm.

Having a kinematic model of its arm opens up a range of possibilities for the robot as it enables it to plan movements in order to accurately and deliberately interact with the world and objects within the world. The layered approach of performing exploratory actions in order to build more, and better models can be continued, and in this chapter an effort is made to move beyond the robot, in order to look at building models of objects in the world around the robot.

When learning about itself, the robot took an active role. It had a large amount of control over its own body, and could take actions designed to give it more information about the kinematic structure of its arm. When learning about objects in its environment, the robot can take both a passive and an active role. It may be that the robot learns about an object by observing a human interact with it, or it may be that the robot takes actions itself in order to learn about the object.

The main idea explored in this chapter is that a robot can either verify, or improve a model that it has of an object, by taking specific actions in order to obtain further information for the model. The robot can conduct a period of experimental work in order to methodically test aspects of the model it is constructing.

This approach has a number of useful applications. A robot can predict the behaviour of an object, and then perform an experiment to confirm or invalidate this prediction. A robot could also perform experiments by manipulating an object in order to try to determine its material properties, i.e. heavy/light, stiff/flexible

etc. This could be useful in a disaster zone, or during search and rescue operations where a robot may need to find suitable objects to perform a certain task, or to confirm that an object is safe and strong enough to use.

6.1 Building Articulated Models of Flat Pack Cardboard Boxes

A motivating problem is needed to provide a physical test bed for investigating ways in which a robot can use exploratory actions in order to build models of objects. To meet this demand, a decision was made to limit the class of objects examined by the robot to flat pack cardboard boxes. Where, for the purposes of this research, flat pack cardboard boxes are those that can be folded from a single sheet of pre-cut cardboard.

This restriction allows the complexity of having to deal with all possible object types, which could obscure aspects of the investigation, to be avoided. Instead, focusing in on flat pack boxes provides a class of objects which are simple to model, and reason about, whilst at the same time retaining a number of interesting properties. Flat pack boxes are straightforward to model as they can be represented as a set of planar facets connected by hinges. They are also interesting as they can be folded into a wide variety of shapes, and may look very different from their initial and final states as they are being assembled and folded. This work primarily looks at the task of manipulating the boxes in order to locate and learn about the hinges of the box, but going further, the problem of programming a robotic system that can learn to manipulate and to assemble flat pack boxes poses many challenges. Also a robot that has the ability to learn about and manipulate a range of flat pack cardboard boxes potentially has a large number of commercial applications.

This chapter presents a system which allows a robot to autonomously build a model of a flat pack cardboard box. This involves identifying the outline of the box, and then determining the position and orientation of any hinges that may be present in the box. The task is challenging as the robot must not only track the pose of the box, but also keep track of its articulation. In doing so, this work shows how a robot can extract information about the structure of a box by interacting with it.

The reason for building a robotic system that can fold cardboard boxes is that it shows how the idea of exploratory actions that were applied to the robot learning about itself in Chapters 4 and 5 can be applied equally well to allow the robot to learn about items in the world around it. By attempting to fold a cardboard box, and evaluating whether the attempt was successful or not, the robotic system gains information about the box that it can use to augment information obtained from

other sensing modalities, such as its vision system. Exploratory actions therefore allow the robot to build more accurate models of objects in the world around them than would be possible by just passively using one sensing modality.

The presented system combines vision and manipulation in order to improve the quality of the generated models. A robot may be able to identify potential hinge locations by observing visual cues, or by examining the outline of the box in order to find likely hinges, but a system relying on purely visual data is likely to return a number of false positives. The system presented here overcomes this problem by attempting to manipulate the box in order to exercise potential hinges, and to therefore either prove or disprove the presence of the hinge. If a hinge is there, then manipulating the hinge can also help to refine its location.

Feedback is obtained regarding the absence or presence of a hinge by combining information from 2 different sensor modalities. The first sensor modality is vision, with a visual tracker used to track the movement of the box as hinges bend. The second sensor modality is obtained by making use of force sensing capabilities in the robot to determine the effort applied by the robot. If the visual tracking quality is low, or if it takes a lot of effort to try to bend a hinge then it is assumed that the hinge is not present. Alternatively, a hinge may be present, but not positioned as expected, in this case the visual tracker can be run multiple times, with different hinge parameters, in order to determine the maximum likelihood estimate of the location of the hinge joint.

The contributions made by the research in this chapter are

- a novel edge based tracker is presented that incorporates colour cues and can be used to track the motion of articulated flat pack cardboard boxes.
- a strategy that a robot can use to autonomously build models of flat pack cardboard boxes is described.
- a hinge exploration strategy that combines both visual and haptic feedback in order to evaluate the the presence or absence of a hinge is described and evaluated.

The rest of this chapter proceeds as follows. Section 6.2 describes related work. Section 6.3 describes the robotic platform that was used to carry out the work. Section 6.4 describes the working of the model building system in detail with subsections on representing flat pack cardboard boxes (6.4.1), tracking the boxes (6.4.2) and the process of investigating and evaluating potential hinges (6.4.4.2). Section 6.5 describes the work done to evaluate the presented system, and Section 6.6 summarises the work of the chapter.

6.2 Background and Related Work

The models built of flat pack cardboard boxes take the form of edge based models, and, as will be described later, edge based tracking is used to follow the motion of the cardboard box as the robot attempts to manipulate potential hinges on the box.

Much work has been done on edge based tracking, with Lowe (1991) giving an early technique for determining the pose of parametrised 3D models by matching model edges projected into image space with edges detected using the Canny edge detector. Drummond and Cipolla (2002) presented an efficient method for edge based tracking that considered the 3D motion of objects using Lie groups, and was therefore able to cast the process of tracking objects from frame to frame as an iterative re-weighted least squares optimisation problem.

Edge based tracking can sometimes fail to be robust, as there are often a large number of distractor edges in images which may cause tracking to fail (Vacchetti et al. 2004). Some researchers have attempted to incorporate Particle Filters into edge based tracking systems in order to improve robustness; the work of Klein and D. Murray (2006) and Choi and Christensen (2012) (which builds on the work of Drummond and Cipolla (2002)) are notable examples of such systems.

Colour can provide a useful cue to assist in tracking objects and to improve the robustness of tracking when combined with other methods. Birchfield (1998) presented a system which made use of both colour histograms and intensity gradients in order to track the position of a person's head. Taiana and Nascimento (2008) and Panin et al. (2008) both showed how sampling the colours alongside detected edges could be used to enhance an edge based tracker.

Articulated object tracking has been tackled in a number of the works already mentioned in this section (Lowe 1991) and (Drummond and Cipolla 2002). Schulman et al. (2013) however, took a particularly interesting approach and presented a physics based tracker where the tracked object was simulated in a physics engine. This had the advantage that it was easily able to incorporate a wide variety of physical constraints such as the non intersection of solid links within a kinematic chain.

Previous researchers such as Fitzpatrick and Metta (2003) and Katz and Brock (2008) have used robotic systems and active vision to interact with objects in order to build models of them. This system is different however in that haptic feedback is combined with visual feedback when interacting with objects.

6.3 Robotic Platform

The work in this chapter was performed using a Baxter robot from Rethink Robotics. The Baxter robot was chosen because unlike the BERT robot its arms

are able to sense forces, which therefore allows it to detect when a box is unable to bend due to a hinge not being present. The Baxter robot is able to sense forces as its arms are built using Series Elastic Actuators (SEAs) which are formed by inserting a spring element between the gearbox of a joint and the load. Measuring the extension of the spring element gives an indication of the force being experienced by the arm.

The Baxter robot can be fitted with a variety of different end effectors, and for this work, a suction gripper was used on the left arm, and a parallel jaw gripper was used on the right arm. These simple grippers were easier to use to manipulate the cardboard boxes than the more complicated anthropomorphic hands that the BERT robot is equipped with.

Figure 6.1 shows the configuration of the Baxter robot that was used for this work. In addition to the 2 grippers, an extra camera was mounted onto the front of the robot in order to observe the bending of the hinges. This is because the cameras that the Baxter robot is equipped with as standard did not provide a good view of the target area.

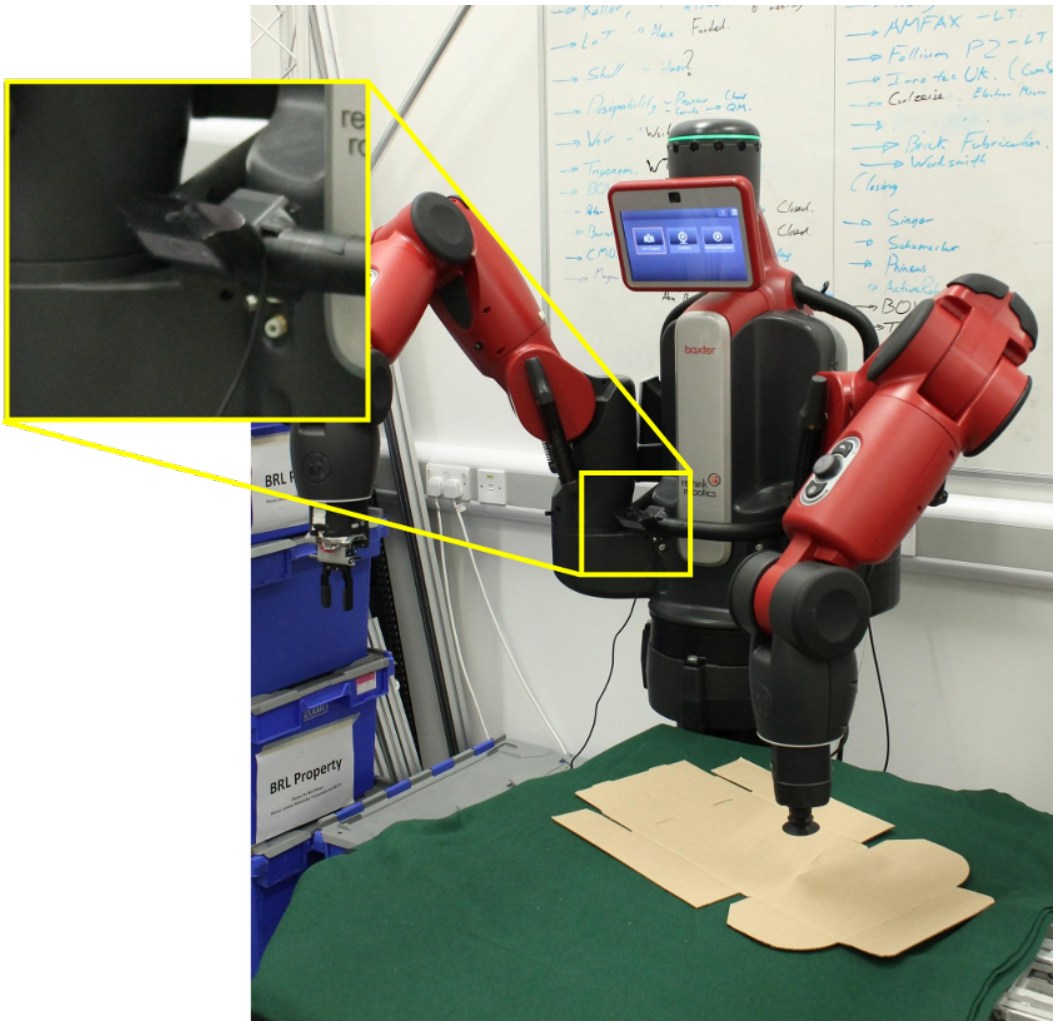


Figure 6.1: The Baxter robot used for this work. A vacuum gripper on the left arm is used to move the box, the right arm is used to try to bend hinges, and the camera mounted to the front of the robot observes the hinge bend motion.

6.4 Method

The robotic system builds a model of a flat pack cardboard box using the method shown in Figure 6.2.

The process starts by building an initial model of the box using just the outline of the box. At this point, no hinges are included in the model. To add hinges, the robotic system goes through a process of identifying potential hinges. There are a number of ways in which this could be done, most methods will involve vision. For example, the robotic system could use edge detection filters, coupled

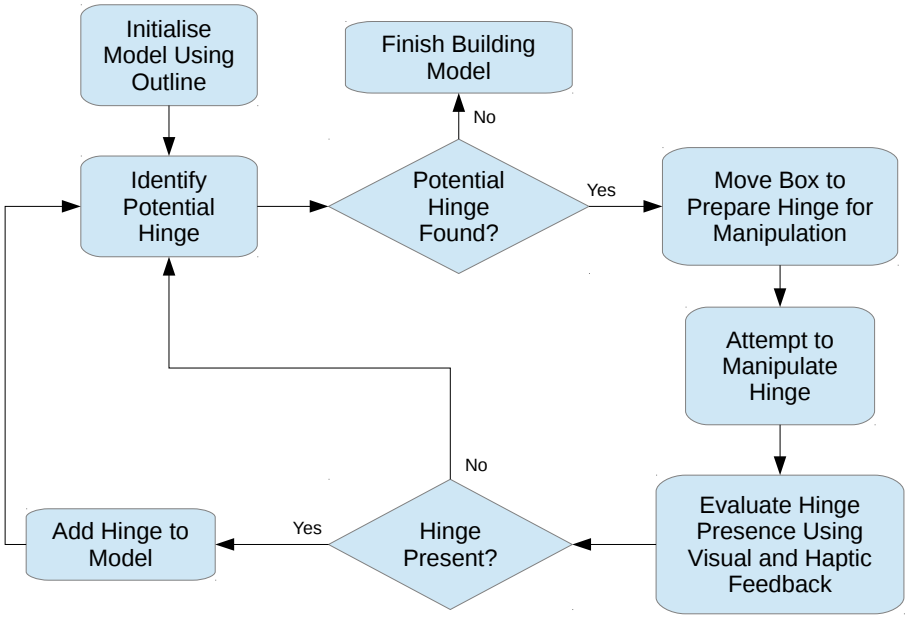


Figure 6.2: The process used to autonomously build a box model.

with a line detection algorithm such as the Hough transform to identify creases that correspond to hinges. Alternatively the system could assume that hinges start and end at concave points on the outline of the box. At the extreme end of the scale, simply guessing possible start and end points of a hinge, is also a method that could be used to identify potential hinges.

In this work, the autonomous validation of potential hinges was the main topic of interest, *not* the identification of potential hinges. Therefore for this work, potential hinges are specified manually.

Assuming that a potential hinge can be identified, the robot manoeuvres the box so that the potential hinge can be manipulated. This is done by holding the box in place using the left arm of the robot, and then moving the right arm of the robot though a pre-recorded motion that attempts to move the hinge up and down.

As the potential hinge is being manipulated, the robot evaluates the likelihood that it actually exists using a combination of visual and haptic feedback. That is, the robot expects to see the hinge move as expected, and it expects to not experience too much resistance to its exploratory motion. Based on the result of the exploratory manipulation, the robot is able to decide if the potential hinge exists or not. If it is decided that the hinge does exist then it is added to the model of the flat pack cardboard box. In either case, at this point the robot returns to looking for another potential hinge and the exploration process repeats until all potential hinges have been explored, and no more potential hinges can be identified.

To build the model of the flat pack cardboard box, the Baxter robot starts with information about itself in the form of a kinematic model, but it does not know anything about the box it will be investigating aside from the fact that it is a flattened cardboard box. It would be possible to build a kinematic model for the Baxter robot using the techniques presented in Chapter 4 but for the experiments presented here the decision was made to use the one supplied by the manufacturer to reduce the work that needed to be done.

6.4.1 Representing the Cardboard Box Model

Figure 6.3 shows a model for a simple cardboard box with 4 hinges. A cardboard box is modelled as a planar contour which is represented as an ordered set of edge points $\mathfrak{E} = \{\mathfrak{e}_1, \mathfrak{e}_2, ..., \mathfrak{e}_N\}$. The contour encloses a simply connected region, i.e. for now cardboard boxes which have holes in them are ignored. Each edge point $\mathfrak{e}_i \in \mathfrak{E}$ has a position $\mathbf{v}(\mathfrak{e}_i)$ attribute and a normal $\hat{\mathbf{n}}(\mathfrak{e}_i)$ attribute pointing out from the interior of the cardboard box. The edge points are positioned in a 3D space $\mathbf{v}(\mathfrak{e}_i) \in \mathbb{R}^3$ but for the canonical pose of the box, all points are positioned on the $z = 0$ plane, the mean position $\bar{\mathbf{v}}(\mathfrak{e}_i)$ is coincident with the origin, and points are ordered so that the contour is traversed in a clockwise fashion about the positive z-axis when viewed from above.

The hinges of the box are represented as the set $\mathfrak{H} = \{\mathfrak{h}_1, \mathfrak{h}_2, ..., \mathfrak{h}_L\}$ where each $\mathfrak{h}_i \in \mathfrak{H}$ is an ordered pair of edge points $\mathfrak{h}_i = (\mathfrak{e}_j, \mathfrak{e}_k)$ where $j < k$. Hinges are not allowed to cross, and a hinge is only valid if it does not pass outside the contour of the box.

The act of adding hinges to the box has the effect of dividing it into a number of faces $\mathfrak{F} = \{\mathfrak{f}_0, \mathfrak{f}_1, ..., \mathfrak{f}_L\}$ where L is equal to the number of hinges the box has. The first face $\mathfrak{f}_0$ is termed the *root* face and this has a Cartesian coordinate frame located at the origin. The remaining faces each correspond to one of the hinges, and have a Cartesian coordinate frame which is positioned on the edge point at the start of the hinge, with the z-axis parallel to the z-axis of the root face's coordinate frame, and the x-axis pointing along the hinge to the other edge point. Given a vector of hinge angles $\boldsymbol{\theta} \in \mathbb{R}^L$ it is possible to calculate a transform $\mathbf{T}(\mathfrak{f}_i)$ for each face which gives the pose of the coordinate frame for face $\mathfrak{f}_i$ in the coordinate frame of the root face. A function $f(\mathfrak{e}_i) : \mathfrak{E} \rightarrow \mathfrak{F}$ is defined which gives the particular face to which an edge point belongs. Edge points which lie on hinges (an edge point may be used in more than one hinge) are assigned arbitrarily to one of the faces that contains one of the hinges.

6.4.2 Visually Tracking the Box Model

For the robot to *perceive* the cardboard box, it must be able to estimate the pose of

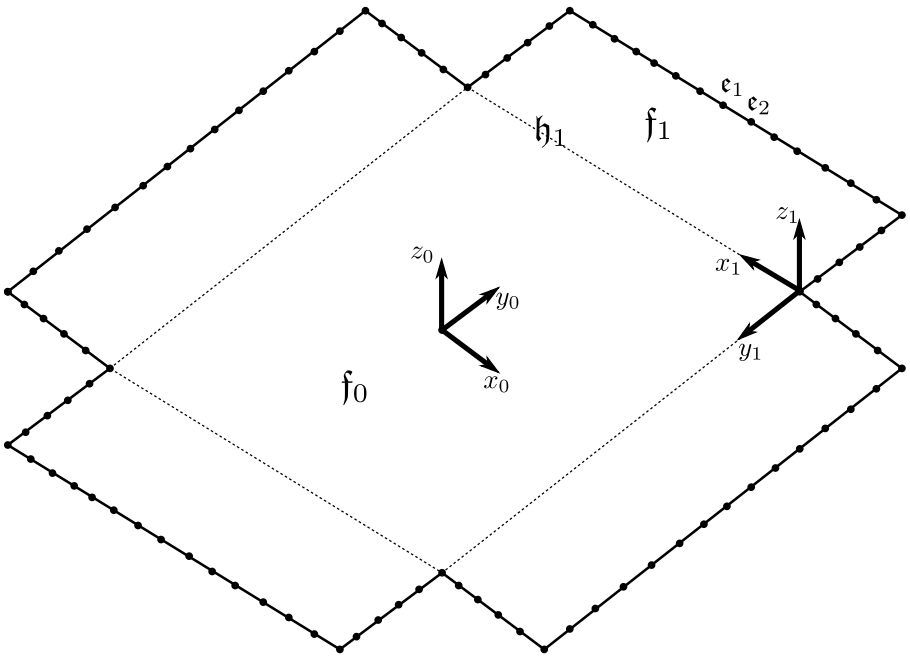


Figure 6.3: An example of how edge points and hinges form a box model, and how the coordinate frames of each face are positioned. For clarity, only a few elements of each type are labelled.

the box along with the angle of its hinges from a series of rectified images $\mathtt{I}_t$ that are retrieved from a camera at regular time steps $t = 1, 2, ..., T$.

The state of the box at time t is represented as

$$\mathbf{b}_t = (\mathbf{E}_t, \boldsymbol{\theta}_t), \tag{6.1}$$

where $\mathbf{E}_t$ is an Euclidean transformation matrix which represents the position and orientation of the box in camera space at time t and $\boldsymbol{\theta}_t \in \mathbb{R}^L$ is a vector of the hinge angles at time t.

6.4.2.1 Camera Model

Without loss of generality it is assumed that the camera matrix is equal to the identity matrix $\mathbf{I}$. A calibrated monocular camera system is used, and each image is rectified as it comes in to remove distortion. This allows the use of the standard pin-hole camera model. First the position of an edge point in model space $\mathbf{v}(\mathfrak{e}_i)$ is transformed to give its position in camera space $\mathbf{v}^{\mathcal{C}}(\mathbf{b}_t, \mathfrak{e}_i)$

$$\mathbf{v}^{\mathcal{C}}(\mathbf{b}_t, \mathfrak{e}_i) = \begin{pmatrix} x^{\mathcal{C}} \\ y^{\mathcal{C}} \\ z^{\mathcal{C}} \\ 1 \end{pmatrix} = \mathbf{U}(\mathbf{b}_t, f(\mathfrak{e}_i))\mathbf{v}(\mathfrak{e}_i) \tag{6.2}$$

where the function $\mathbf{U}$ returns the homogeneous transformation matrix that gives the pose of the face $f(\mathfrak{e}_i)$ in camera space. Camera space positions are projected into image space according to

$$\mathbf{v}'(\mathbf{K}, \mathbf{b}_t, \mathfrak{e}_i) = \mathbf{K} \begin{pmatrix} \frac{x^{\mathcal{C}}}{z^{\mathcal{C}}} \\ \frac{y^{\mathcal{C}}}{z^{\mathcal{C}}} \\ 1 \end{pmatrix} \tag{6.3}$$

where $\mathbf{K}$ is the matrix of intrinsic camera parameters

$$\mathbf{K} = \begin{pmatrix} l_u & 0 & u_0 \\ 0 & l_v & v_0 \end{pmatrix}. \tag{6.4}$$

Here l_u and l_v represent the focal length in pixels, and (u_0, v_0) is the principle point of the camera.

Similar calculations are performed to project edge point normals into image space to give $\hat{\mathbf{n}}'(\mathbf{K}, \mathbf{b}_t, \mathfrak{e}_i)$. For clarity, for the rest of this chapter the extra parameters are dropped and image space positions and normals are referred to as $\mathbf{v}'(\mathfrak{e}_i)$ and $\hat{\mathbf{n}}'(\mathfrak{e}_i)$ respectively.

6.4.2.2 Evaluating Box States with a Fitness Function

The extent to which a potential box state $\mathbf{b}_t$ matches a given image $\mathtt{I}_t$ is evaluated using a fitness function $boxfit(\mathbf{b}_t, \mathtt{I}_t)$ where

$$boxfit(\mathbf{b}_t, \mathtt{I}_t) = (1-\lambda)edgefit(\mathbf{b}_t, \mathtt{I}_t) + \lambda colourfit(\mathbf{b}_t, \mathtt{I}_t). \tag{6.5}$$

Here $edgefit(\mathbf{b}_t, \mathtt{I}_t)$ is a fitness function that measures how well the edge points projected into image space match with detected edge points, $colourfit(\mathbf{b}_t, \mathtt{I}_t)$ is a fitness function that measures how well the colour of the projected box matches the image, and $\lambda \in [0,1]$ is a parameter that determines the relative weights of the fitness functions. In all experimental work λ was set to 0.3, as this was found to give the best trade-off between the edge and colour fitness functions. The fitness functions $edgefit(\mathbf{b}_t, \mathtt{I}_t)$ and $colourfit(\mathbf{b}_t, \mathtt{I}_t)$ are chosen to return values in the range $[0,1]$ and so this means that $boxfit(\mathbf{b}_t, \mathtt{I}_t)$ returns 1 if the state $\mathbf{b}_t$ perfectly explains the data seen in $\mathtt{I}_t$.

Edge Fitness Function - The edge fitness function $edgefit(\mathbf{b}_t, \mathtt{I}_t)$ is based on a variant of the classical Chamfer Matching algorithm of Barrow et al. (1977)

called Directional Chamfer Matching which was introduced by Liu et al. (2010). To evaluate the fitness function, the image gradient $\nabla \mathtt{I}_t$ is calculated along with the edge image $\mathtt{E}_t$ which is derived from $\mathtt{I}_t$ by applying a Canny filter. $\mathtt{E}_t$ gives detected edge locations, and $\nabla \mathtt{I}_t$ give the orientation of the edges.

The fitness function is evaluated by projecting edge points of the model into image space and then computing an average of the dot product between each projected edge point and the closest corresponding edge pixel (edgel) in the edge image $\mathtt{E}_t$. The search for correspondences is conducted along projected edge normals in order to make it efficient. The fitness function is calculated as follows

$$edgefit(\mathbf{b}_t, \mathtt{I}_t) = \frac{1}{V} \sum_{i=1}^{V} \left| (1 - \frac{|d_i|}{D_i}) \hat{\mathbf{n}}'(\mathfrak{e}_i)^\intercal \hat{\mathbf{o}}_i \right| \tag{6.6}$$

where V is the number of edge points that are projected into $\mathtt{I}_t$, d_i is the distance to the corresponding edgel, D_i is the maximum search distance along a projected normal, and $\hat{\mathbf{o}}_i$ is the normal of the corresponding edgel. The value of D_i is calculated for each edge point individually and in the experimental work conducted was set to correspond to 2.5cm in model space. Figure 6.4 shows the matching process along with examples of each of the quantities used to calculate the fitness function.

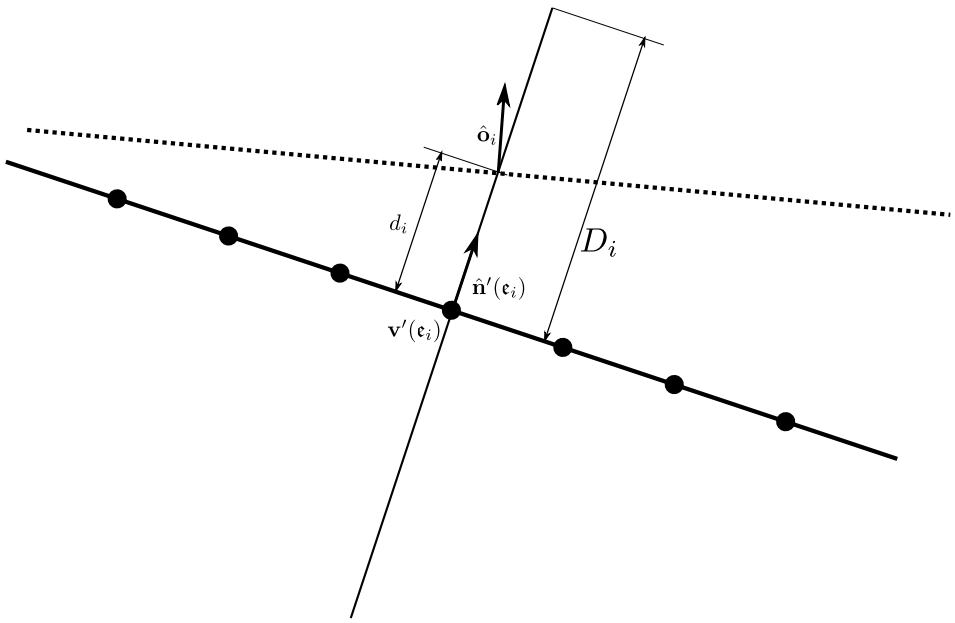


Figure 6.4: The process used to match model edge points, projected into image space with edge pixels.

Colour Fitness Function - The colour fitness function $colourfit(\mathbf{b}_t, \mathtt{I}_t)$ draws

inspiration from the work of Taiana and Nascimento (2008) and Panin et al. (2008) by evaluating how well the colours within and around the projection of the box model into the image, match against a pre-built colour histogram of the box.

Given a histogram $\mathscr{B}$ that describes the colour of the tracked box, histograms $\mathscr{I}$ and $\mathscr{E}$ are built that respectively describe the distribution of internal colours and external colours relative to the projection of the box model in image space. Colours are sampled from inside the box using a set of sample points that are projected into image space. Sample points outside the box model are found by sampling from points along the edge point normals projected into image space. An example of a typical set of internal and external sample points are shown in Figure 6.5.

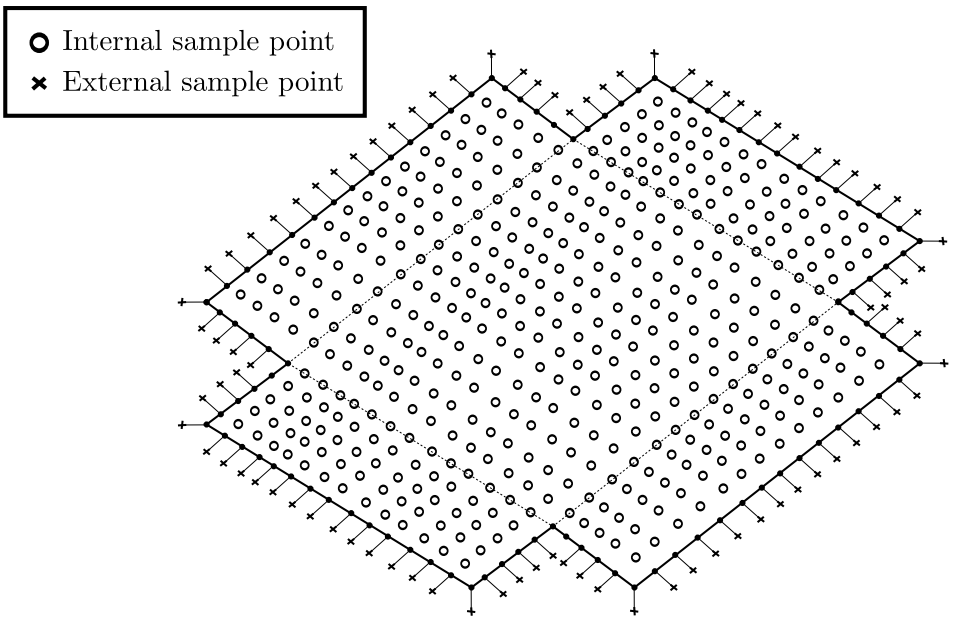


Figure 6.5: A typical set of internal and external sample points used to build histograms of the colour inside and outside of the box model when projected into image space (some sample points have been omitted for clarity).

Intuitively, the aim with the colour fitness function is to try to find box states where the internal colour histogram $\mathscr{I}$ matches the model histogram $\mathscr{B}$ well, and where the external colour histogram $\mathscr{E}$ is distinct from the model histogram.

The colour fitness function $colourfit(\mathbf{b}_t, \mathtt{I}_t)$ is therefore defined as

$$colourfit(\mathbf{b}_t, \mathtt{I}_t) = \frac{1 - d(\mathscr{B}, \mathscr{I}) + \eta d(\mathscr{B}, \mathscr{E})}{1 + \eta} \tag{6.7}$$

where η controls the relative weights of the different elements of the fitness function (it was found that $\eta = 1.5$ gave good results for this work). The function

$d(\mathscr{H}_1, \mathscr{H}_2)$ is the normalised Hellinger distance between 2 histograms (Nikulin 2011) defined as

$$d(\mathscr{H}_1, \mathscr{H}_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{\mathscr{H}}_1 \bar{\mathscr{H}}_2 B^2}} \sum_b \sqrt{\mathscr{H}_1(b) \cdot \mathscr{H}_2(b)}} \tag{6.8}$$

where

$$\bar{\mathscr{H}} = \frac{1}{B} \sum_b \mathscr{H}(b) \tag{6.9}$$

and B is the total number of histogram bins.

The histograms for this work were constructed in the HSV colour space using the Hue and Saturation components as these have been shown to be resilient to changes in illumination.

6.4.2.3 Frame to Frame Tracking

To track the box from time t to time $t+1$ the system starts at the state $\mathbf{b}_t$ and seeks to find the new state $\mathbf{b}_{t+1}$ such that

$$\mathbf{b}_{t+1} = \arg\max_{\mathbf{b}_{t+1}} boxfit(\mathbf{b}_{t+1}, \mathtt{I}_{t+1}) \tag{6.10}$$

The state $\mathbf{b}_{t+1}$ is related to the previous state $\mathbf{b}_t$ by

$$\mathbf{E}_{t+1} = \mathbf{E}_t \mathbf{M} \tag{6.11}$$

and

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \dot{\boldsymbol{\theta}} \tag{6.12}$$

where $\mathbf{M}$ is an Euclidean transformation matrix representing the interframe motion and $\dot{\boldsymbol{\theta}}$ is the change in hinge angles between frames.

The Euclidean transformation matrices representing the pose of the box at specific points in time are members of the 6-dimensional Lie group $SE(3)$, and the interframe motion $\mathbf{M}$ is a member of the Lie algebra $se(3)$. The generators of $SE(3)$ (which together form the a basis for $se(3)$) are represented by the matrices

$$\mathbf{G}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{G}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{G}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$
$$\mathbf{G}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{G}_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{G}_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.13}$$

Therefore $\mathbf{M}$ can be obtained from a 6-dimensional parameter vector $\boldsymbol{\alpha} \in \mathbb{R}^6$ by using the exponential map

$$\mathbf{M} = \exp(\sum_{i=1}^{6} \alpha_i \mathbf{G}_i) \tag{6.14}$$

Combining $\boldsymbol{\alpha}$ and $\dot{\boldsymbol{\theta}}$ there are $6+L$ parameters to optimise in order to find the new state $\mathbf{b}_{t+1}$. Now, Equation 6.10 is non-linear and contains local minima, but it is possible to get good tracking results by keeping the interframe movements small. For this work the inter-frame movements can in most cases be kept small as the Baxter robot used to move the box is capable of slow controlled movements.

To minimise Equation 6.10 the implementation of DIRECT-L algorithm (Gablonsky and Kelley 2001) from the NLopt library (Johnson 2010) was used. The DIRECT-L algorithm is a global search algorithm that systematically subdivides the search area into smaller and smaller hyper-rectangles. Due to the small size of the expected interframe motion, the search range was limited in order to prevent convergence to a distant minima.

6.4.3 Initialising Tracking

If a box model is available, then tracking can be initialised using an exhaustive search method, whereby a range of different possible box states are evaluated in order to find the maximum value of the fitness function c.

Due to the high dimensionality of the search space, an exhaustive search over all 6 degrees of freedom, plus the hinge angles, for the pose of the box model would take an excessively long time. To overcome this problem, it is assumed that the hinge angles and approximate orientation of the box model is known, and a 'template matching' style search is conducted in camera space. Specifically, during the search the x, y, and z position of the box is allowed to vary in camera space along with the rotation around the camera axis. This gives a 4 dimensional search space.

In order to perform the search efficiently, an image pyramid is formed by successively scaling down the input image so that higher levels contain smaller versions of the input image and so provide a reduced search space. In order to test for the presence of the box a search is performed at the top level, and all possible locations where the value of $c \geq c_{min}$ are tracked to lower levels of the pyramid where a higher resolution exhaustive search is carried out.

Once the best possible location has been found using exhaustive search, the pose of the box is refined by performing a round of tracking as described in Section 6.4.2.3.

6.4.4 Autonomously Building a Cardboard Box Model

A cardboard box model, as described in Section 6.4.1 is derived from an image of a cardboard box, opened up and flattened out to lie upon a planar surface. The image of the cardboard box is taken using a calibrated camera aligned so that its image plane is parallel to the cardboard box.

The box is placed on a background consisting of a single, solid colour and so it is easy to automatically segment it from the background using a simple thresholding method in order to form a binary mask image showing the extents of the box.

The edges of the box are found by applying a Canny edge detecting filter to the binary mask and then searching for the contour which represents the outline of the box using the contour finding algorithm of Suzuki and Abe (1985). At the moment it is assumed that the cardboard boxes correspond to a simply connected 2D space, i.e. they contain no holes. Therefore boxes are rejected if more than 1 contour is found in the image. The image points which make up the contour become the edge points $\mathfrak{E}$ with coordinates being translated from pixels to metres using the focal length of the camera. Normal vectors $\hat{\mathbf{n}}(\mathfrak{e}_i)$ are calculated by using a vector perpendicular to the direction to a neighbouring edge point.

6.4.4.1 Identifying Potential Hinges

Initially the model of the box does not contain any hinges, as the box is completely flat. Hinges can be specified however as an ordered pair of edge points as defined in Section 6.4.1.

There are a number of ways in which potential hinges could be identified. Hinges could be detected using visual cues such as looking for lines in the internal regions of the cardboard box, or looking for regions of high curvature on the outline of the box, as these often correspond to the end points of hinges. Alternatively, visual identification can be ignored altogether and potential hinges can be chosen by simply choosing pairs of random edge points (perhaps subject to

the constraint that they are a certain minimum distance apart). The process of testing a hinge and subsequently refining the hinge position presented in the next section is robust to errors in the in the initial placement of the hinge so it does not matter if proposed hinges do not align precisely with real world hinges.

For the experiments conducted for this work, the choice was made to specify potential hinge locations manually. In this way the performance of hinge testing and refinement method, which is the main thrust of this work, can be evaluated independently from any particular hinge identification method.

6.4.4.2 Testing Potential Hinges

Once a potential hinge has been found, the robot moves the box to evaluate the potential hinge in order to try to determine if a hinge is actually present or not. A hinge is tested by playing back a prerecorded test motion with the right arm of the robot, whilst the left arm of the robot holds the box in place. Using a prerecorded hinge test motion means that the potential hinge needs to be placed at a set position and orientation in the workspace of the robot.

Figure 6.6 shows the *ideal hinge location* for the test motion, marked with a red line. At the moment, the ideal hinge location is set manually by examining the prerecorded test motion that will be used and defining a line segment that coincides with the axis of rotation induced by the test motion.

Assuming that the current position and orientation of the box is known then it is possible to calculate the ideal position and orientation of the box that will align the potential hinge with the ideal hinge location. The robotic system is able to keep track of the movement of the box by periodically moving its left arm up and viewing the workspace from above using the camera in its left wrist. Figure 6.6 shows the measured box position and orientation in green, the potential hinge location in orange and the position and orientation of the box that is required to test the hinge in blue.

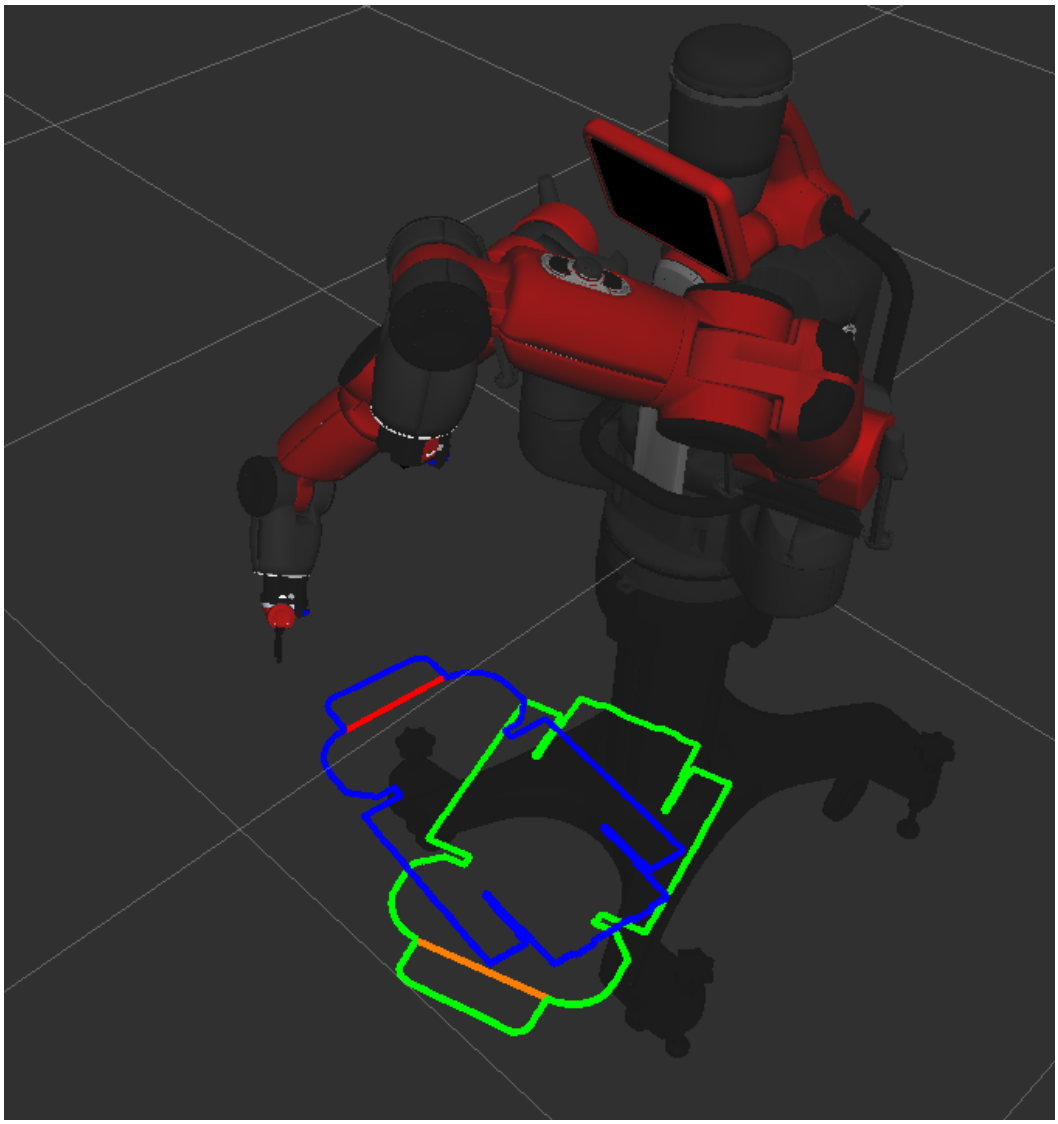


Figure 6.6: The measured box pose is shown in green, and the potential hinge is shown in orange. The ideal hinge position (shown in red) is defined when setting up the system and allows the ideal pose of the box (shown in blue) to be calculated.

The left arm of the robot, which is fitted with a vacuum gripper, is used to rotate and translate the box in 2D so that the position of the potential hinge aligns as best as possible with the ideal hinge position.

At this point, tracking of the box switches to the camera mounted on the chest of the robot. As the hinge exploration motion plays out with the right arm, the presence or absence of the potential hinge is evaluated using both a haptic evaluation method and a visual evaluation method.

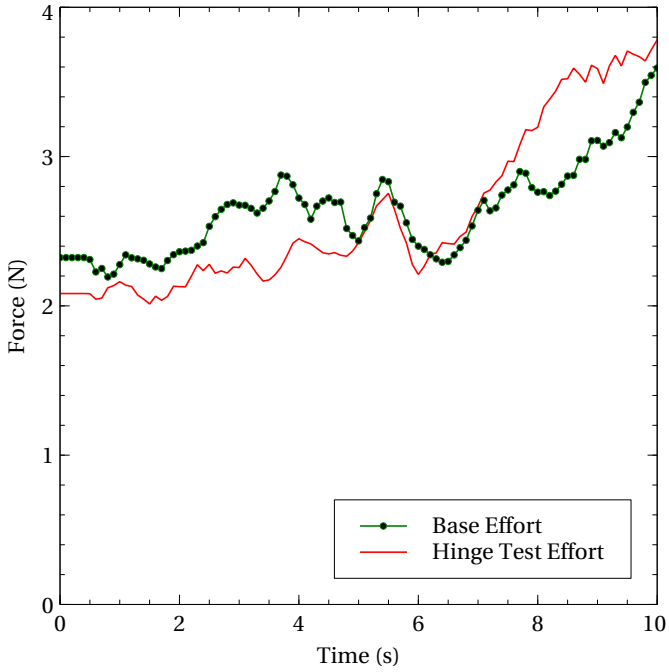
6.4.4.3 Haptic Evaluation

The Baxter robot is able to provide an estimate of the effort that its motors are exerting on its end effectors. The effort is represented as a wrench (force and torque) in the coordinate frame of the robot's base joint. By looking at the magnitude of the Cartesian force vector, it is possible to obtain an approximate estimate of the effort being applied to the end effector. It seems reasonable to assume that when a hinge is not present, then more effort will be applied to the end effector as the robot's motors seek to overcome the resistance of the cardboard.

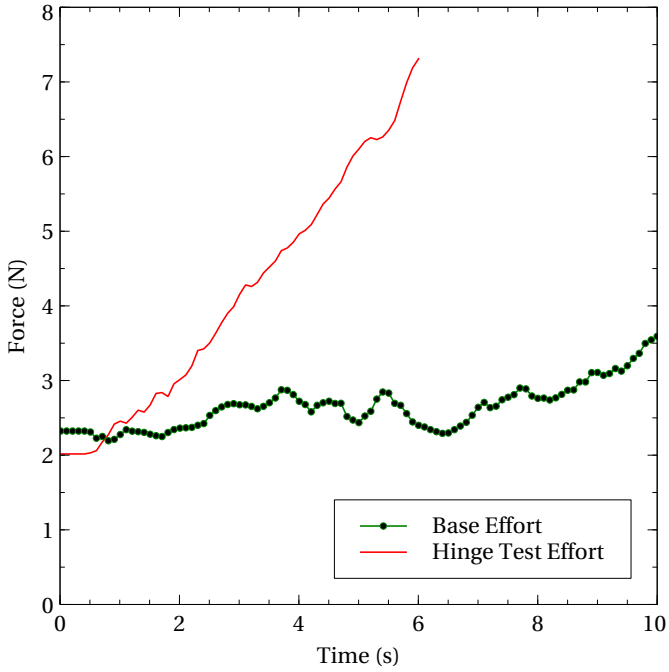
To take advantage of this fact, an effort profile is pre-recorded whilst running the hinge exploration on a known hinge. For future exploratory motions it is then possible to compare the effort applied with the recorded motion. If the applied effort exceeds that of the known 'hinge present' effort by more than a given threshold ζ_h, then it is assumed that a hinge is not present and the exploratory motion is aborted in order to avoid damaging the box. The value of ζ_h was set to 4N for all of the experimental work presented later on.

The effort reported by the robot is provided at a rate of 100Hz, and is very noisy. To reduce the effects of noise, a low pass filter in the form of a moving average filter with a window size of 50 is applied to the effort signal.

Figure 6.7 shows a pre-recorded force profile for the hinge exploration movement along with 2 different profiles. The first recorded when a hinge was present and the second recorded when a hinge was not present.



(a) Hinge present



(b) Hinge not present and test aborted

Figure 6.7: The effort exerted by the right end effector as the exploratory hinge move is made is compared to a pre-recorded baseline effort profile in order to determine the presence or absence of a hinge.

6.4.4.4 Visual Evaluation and Refining the Location of a Hinge

The haptic evaluation may answer the question of whether a hinge is present or not, but it does not give much information about how closely the hypothesised position and orientation of the hinge matches the real world position and orientation of the hinge.

Visual evaluation provides a way to answer that question, as by using the visual tracker that was described earlier in the chapter, the movement of the box can be tracked as the hinge is bent, and the fitness function $boxfit(\mathbf{b}_t, \mathtt{I}_t)$ can be evaluated at each time frame to determine how well the tracked pose of the box matches the image that is retrieved from the camera mounted onto the front of the robot.

As the hinge evaluation starts, a visual tracker for the camera mounted onto the front of the robot is initialised to track the movement of the box, with the initial pose set using the last pose of the box in world space that was measured using the camera in the left wrist of the robot.

The initial pose of the box is refined with a round of tracking as described in Section 6.4.2.3. For all subsequent motion though, it is possible to take advantage of the fact that the movement of the box is going to be restricted as it is held in place by the left arm of the robot. This means that it is not necessary to track the full 6 degrees of freedom for the box pose. Instead, only movement in the plane of the box model needs to be considered (assuming that the box can slip past or twist around the point at which it is held) along with the angle of the hinge being tested. This reduces the tracking optimisation problem to a search in 4 dimensions (3 for the movement of the box, and 1 for the hinge angle).

The value of the fitness function $boxfit(\mathbf{b}_t, \mathtt{I}_t)$ over time is useful for confirming the result of the haptic evaluation. But if the value of the fitness function falls below a threshold ζ_v (set to 0.45 for the experimental work) then this is taken as an indication that although the haptic evaluation may say that a hinge is present, the position of the potential hinge that is being tested is wrong.

The thinking here is that the more accurate the model of the box is, in terms of how precisely a hinge is positioned in the box model, then the value of the fitness function $boxfit(\mathbf{b}_t, \mathtt{I}_t)$ will be consistently higher as the box is tracked from frame to frame. In these situations, once the exploratory motion used to test for the presence of the hinge has finished, a search is conducted to find the optimum hinge $\mathfrak{h}_{opt}$ to describe the observed motion. The hinge $\mathfrak{h}_{opt}$ is defined as

$$\mathfrak{h}_{opt} = \arg \max_{\mathfrak{h}=(\mathfrak{e}_j,\mathfrak{e}_k)} \sum_{t}^{T} boxfit(\mathbf{b}_t, \mathtt{I}_t) \tag{6.15}$$

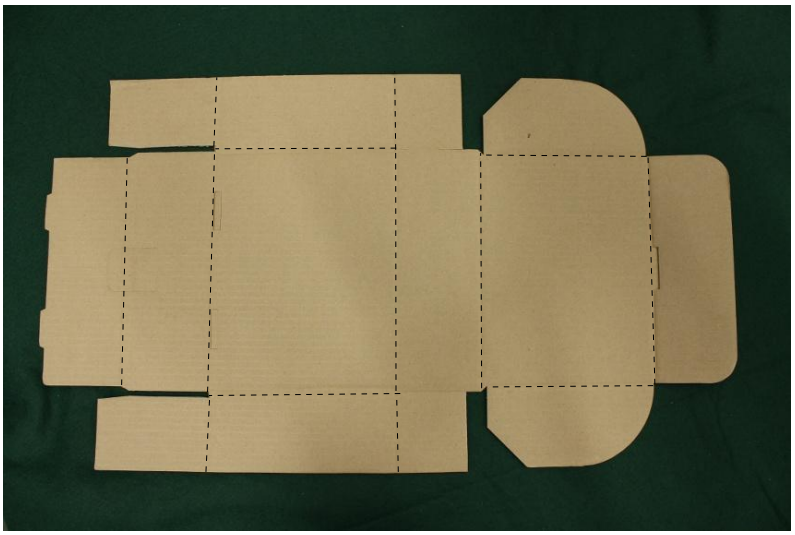
where T is the total number of images recorded by the camera during the hinge exploration movement.

Equation 6.15 is a non-linear objective function which is optimised by using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method (Nocedal and Wright 2006). The search space for the optimisation is 2 dimensional as modifying the position of a hinge involves moving the end points of the hinge to different potential edge points.

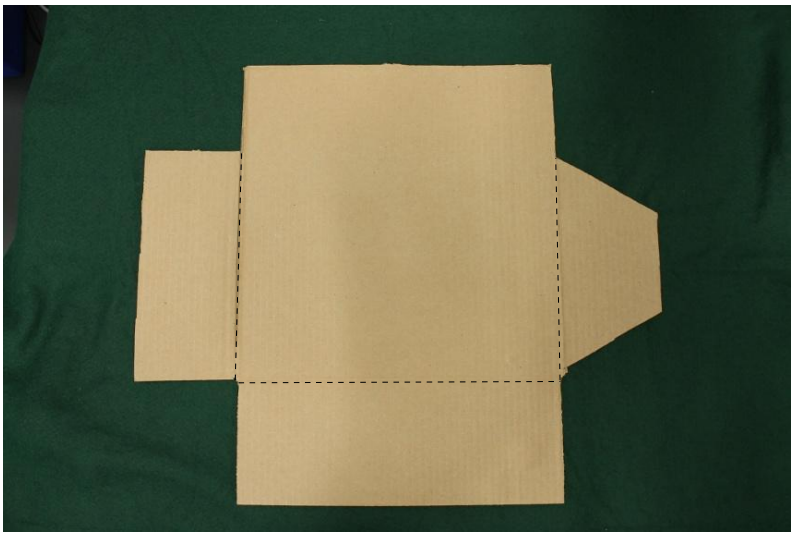
6.5 Experimental Evaluation

A number of experiments were performed to evaluate the performance of the system described above.

The system was tested using 2 different cardboard boxes, denoted Box A and Box B, shown in Figure 6.8. Box A has a number of hinges located at different orientations around the box. It also has a couple of internal edges (shown in Figure 6.9) which are not captured by the model building process. Box B is a test piece formed in the shape of an irregular cross that contains just 3 hinges.



(a) Box A



(b) Box B

Figure 6.8: The boxes used in the experiments. Hinges have been marked with dashed lines.

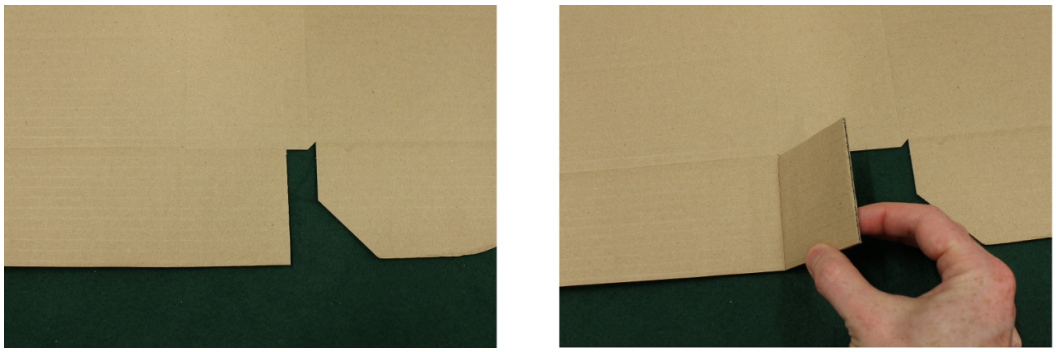


Figure 6.9: Internal edges are those which cannot deduced from the outline of the box.

Prior to testing the system, histograms were built for each the boxes, for use when evaluating the colour fitness function described in Section 6.4.2.2. The histograms were built by using a number of sample camera frames, in which the location of the boxes had been marked by hand.

The ability of the system to determine the presence of hinges was tested by getting it to build models of the 2 boxes. The initial outline of the boxes were constructed using the camera in the left wrist of the Baxter robot, and then potential hinges were specified manually for each of the boxes. The robotic system autonomously evaluated each of these potential hinges one after another; moving the box to align the potential hinge with the ideal hinge position, and then performing the prerecorded test move to classify the hinge. The robotic system was able to classify the potential hinges correctly as either existing on the real box or not.

Next, the ability of the system to accurately determine the position and orientation of a hinge was tested by specifying 10 incorrect hinges which were close to actual hinges. The robotic system was used to test if a hinge was present, and having determined that a hinge was present, the position and orientation of the hinge was refined using the visual tracker and the method given in Section 6.4.4.4. The accuracy of the refinement was assessed by comparing the position and orientation of the identified hinge to the ground truth. The average position error was 0.5cm and the average orientation error was 4 degrees.

In general, the system takes about 3 minutes to test each potential hinge. This includes moving the box into position, testing the potential hinge with the hinge exploration movement, and evaluating the results.

Detection of the presence or absence of the hinges was found to be robust, but the system was found to be vulnerable to problems that could occur due to failures as the Baxter robot tried to manipulate the cardboard boxes. For example, Box A shown in Figure 6.8 was awkward for the robot to manipulate when an attempt was made to test potential hinges on the long edge of the box. In this situation,

the hinge exploration movement had to be recorded carefully so that the box did not collide with the robot when it was manoeuvred into place. As it happened, it was possible to record a hinge exploration movement that worked for all hinges on both boxes (this gives the test hinge location shown by the red line in Figure 6.6) but the problems encountered indicate that the system would struggle to cope with a box that was much larger than Box A. There is no reason why the system could not be adapted to work with a robot that had a greater reach and range of motion however.

6.6 Summary

This chapter has presented a novel method for modelling and tracking the movement of flat pack cardboard boxes. It has also presented a robotic system that can investigate the location of potential hinges on flat pack cardboard boxes and shown how a haptic sensing system can be combined with a visual tracking system in order to provide more information about an object than would have been possible using either sensing system in isolation.

Overall, this work provides a valuable step towards a system that can autonomously learn the structure of flat pack cardboard boxes, and then go on to manipulate and fold them. Such a system would have a number of useful commercial applications as well as providing useful insights into how a robot might learn about novel objects by interacting with them in general.

The models of objects produced in this chapter have been 'physical' models, that represent the physical extents of the object as well as information about its kinematics. In the next chapter, focus shifts to look at object models which describe objects in terms of the text written on them. Even though the form of the models built will be very different to the models built in this chapter, the advantages gained from using exploratory actions to investigate the objects still hold and the next chapter presents a number of exploratory actions which can be used to efficiently find and identify text on objects.

Chapter 7

Object Text Models

Text is everywhere in the modern world and provides a valuable store of information. A key skill for any future robot operating in a human environment will be the ability to read. This would allow the robot to extract information from previously unseen objects which could otherwise be hard to obtain.

In an assisted living application, it seems reasonable to assume that an ability to read objects, i.e. being able to find and understand any text that might be on them, will be a highly useful skill for any general purpose robot. It is infeasible to equip a robot with knowledge about every possible object that it might encounter in a house, as the objects may change on a regular basis; however, any text on the objects will give valuable clues to the robot about the purpose and contents of the object. The ability to read would give the robot the ability to successfully obey orders such as 'get me the medicine' or 'bring me a packet of pasta', whilst having no prior knowledge about the visual appearance of the target object.

As an alternative to trying to read the text on an object, Klingbeil et al. (2011) simply perform a search for a barcode on the object. Locating and reading a barcode is much easier than locating and reading arbitrary text, and could in theory allow all text on an object to be retrieved, if that text was stored in a database alongside the barcode. However, in reality, manufacturers are unlikely to update their barcodes with every packaging revision, and some information, such as the use by date, is not contained in the barcode. More importantly, not all objects with text on them have a barcode, e.g. letters, brochures, and objects which have been annotated with hand written text. In all these situations, a general purpose text reading system would be needed.

A robotic system is presented which can manipulate and explore small household objects in order to detect and read any text which might be on them. The system combines text detection techniques with intelligent manipulation, and as a result of this, it is able to detect and read more text on the objects than would be possible using just static images of the objects taken from a fixed view point.

Other people have built robotic systems which can read text in human environments, but these have tended to focus on getting the robot to read text on signs or walls, e.g. (Mirmehdi et al. 2003), (Posner et al. 2010), (Case et al. 2011), or text on an object that is on a table or a shelf, e.g. Ramos-Garijo et al. (2003). This system, to the best of the author's knowledge, is the first system that attempts to give to a robot the general purpose skill of reading an object in its hand, in a form that would be recognisable to a human.

Reading text from objects held in the hand of a robot is much more challenging than the task of reading text as it might be presented to a desktop scanner. The problem is made harder by the inconsistent or varying illumination found in the natural world. Text items are also often not aligned with the image plane of the camera, may only be partially visible, and may be deformed on non-rigid objects.

A variety of techniques involving manipulating the object and adjusting the viewpoint and zoom level of the camera observing the object, are used in order to try to overcome these challenges. It is possible to extract enough text to be able to successfully classify the objects in a large test set, and to distinguish between objects of different type. This points to text detection and reading as being a useful addition when trying to identify and/or build models of objects with a robotic system.

The rest of the chapter proceeds as follows, Section 7.1 describes existing works and systems, Section 7.2 provides an overview of the robotic text reading application that has been constructed, Section 7.3 describes the key methods used to build the system, and Section 7.4 describes experiments carried out to evaluate the performance of the system along with the results obtained. Finally, Section 7.5 discusses the system as a whole, and looks at areas where future work could be carried out.

7.1 Background and Related Work

The field of text detection is an active one (Liang et al. 2005), with a large number of techniques proposed (Epshtein et al. 2010; H. Chen et al. 2011), and a number of datasets for evaluating their relative efficiency such as (Lucas et al. 2003) and (Yao et al. 2012).

The majority of text detection work focuses on passively identifying text from isolated images. Work has been done on detecting text in image sequences, e.g. Tanaka and Goto (2008) and Merino-Gracia et al. (2012) present wearable systems that continuously capture images and read text in the real world as their users move around. However, the images presented to these systems are still ultimately chosen by the human user. By contrast, the aim here is to build a system which actively searches for text itself, and autonomously adjusts the viewpoint and camera settings

for the pictures it takes, in order to increase its chance of finding readable text.

Recently, more attention has been given to the benefits that might be obtained from introducing a robot into text detection systems in order to actively seek out text. This is an example of *active vision* as defined by Bajcsy (1988), as the robot is potentially able to position the camera intelligently and adjust parameters such as the zoom, in order to reduce ambiguity when reading text. Therefore, more information can be extracted than would otherwise have been possible using just images that had been collected passively.

Examples of systems which use robots in this way include the work of Mirmehdi et al. (2003) and Ruiz et al. (2013) where a pan-tilt-zoom camera is used to actively search for regions of text in a scene. Also the work of Létourneau et al. (2004) and Tanaka and Goto (2007), which both presented mobile robots that could drive round an enclosed area, reading printed signs. A more capable system was presented by Posner et al. (2010) which read text from outdoor images. In addition, Case et al. (2011) presented a robotics system that combined text detection with a Simultaneous Localization and Mapping (SLAM) system to annotate a map with text from signs that the robot had found around the office space.

Research has been done looking at using robots to automate libraries, but any text reading that has been done seems to have focused on reading the covers of the books whilst they are on the shelf (Ramos-Garijo et al. 2003), and not when they have been picked up by the robot.

The approach taken to looking for text on an object is to view it as a form of *active visual search*; this was studied extensively by Tsotsos (1992), and recognises that the visual search for something is in worst case exhaustive, but uses constraints on the possible location of the item searched in order to speed up the search. For example, Wong et al. (2013) and Aydemir and Sjoo (2011) both used spatial constraints on the relative position of objects in order to speed up the task of a robot searching for the objects. In this work, constraints on the likely location of text on the surface of an object are used to speed up the search for text.

7.2 Overview of Application

The test application involves using the Bristol and Elumotion Robotic Torso (BERT) robot (Figure 7.1) that has been used in other parts of the thesis. The Microsoft Kinect mounted as its head provides a 3D vision system for looking at the shape of objects, and this is augmented with a high resolution Canon Powershot camera which is used for detecting and reading text. Objects are handed to the robot which then presents the objects to its vision system. The robot takes high resolution pictures of the object which are then processed by a text

detection algorithm. Regions of the image identified as text are passed to an off the shelf Optical Character Recognition (OCR) program - Tesseract (Smith 2007) - and any text output from the OCR program is the text 'read' from the object. Such text can be used to classify the object, or to infer information about the contents of the object.

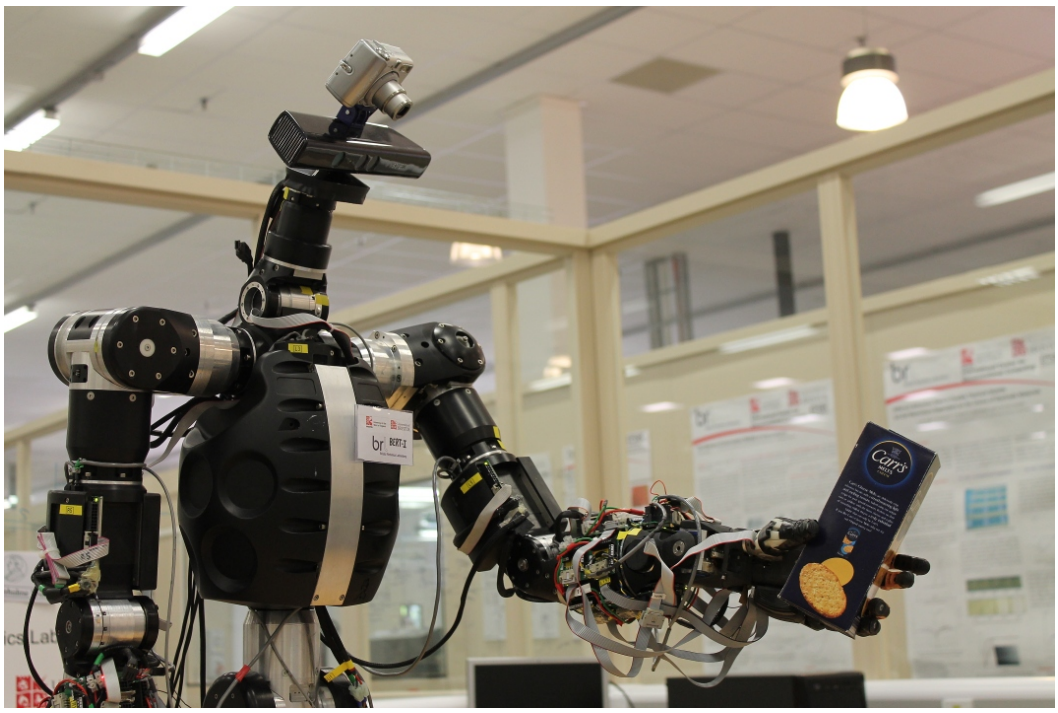


Figure 7.1: The BERT robot with added high resolution camera.

This work focuses on the active vision strategies that are exploited by the robot to choose the poses and camera settings it uses when taking high resolution images of the objects.

7.3 Methods

7.3.1 Exploration Process

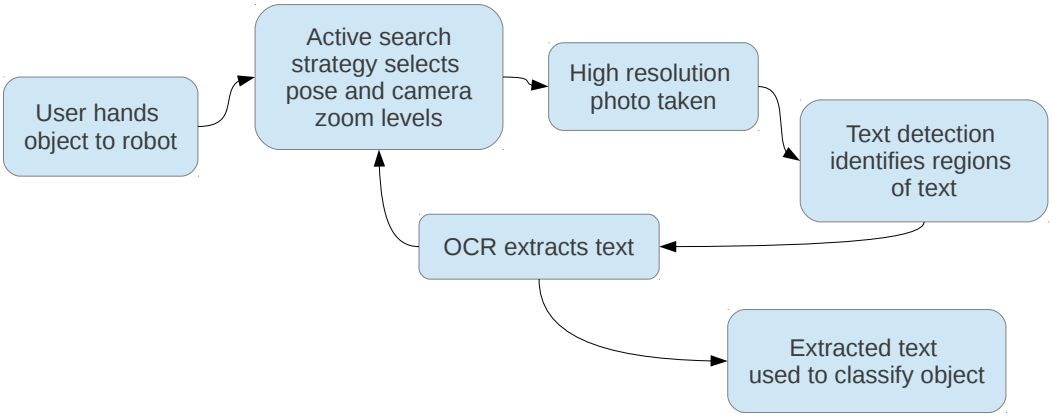


Figure 7.2: The exploration process used to read text from an object.

The exploration process (shown in Figure 7.2) is started when a user hands an object to the robot. A simple depth filter is applied to the output from the Kinect, allowing the robot to sense when somebody is waving an object in front of its vision system. At this point, a prerecorded animation is played out whereby the robot reaches out its hand, the user places an object in its hand, and the robot closes its hand, grasping the object.

The robot then moves through a sequence of (pose, zoom) pairs $(\boldsymbol{\theta}, z)$ where $\boldsymbol{\theta}$ is a vector of joint angles for the robot, and z is a zoom level chosen from the set $Z = \{z_1, z_2, z_3, z_4\}$. Each zoom level in Z represents a different focal length for the high resolution camera, with higher zoom levels being more zoomed in.

As each pose is reached, the corresponding zoom level is set, and a high resolution image is taken of the object. Text detection and recognition is run on these images in order to read the text on the object. The (pose, zoom) pairs are chosen by the active search strategy. Depending upon the search strategy used, the data obtained from the vision system at one pose may affect the next (pose, zoom) pair that is chosen.

7.3.1.1 Object Filtering

The depth camera provides a point cloud $\mathfrak{P} = \{\mathfrak{p}_1, \mathfrak{p}_2, ..., \mathfrak{p}_N\}$ where each point $\mathfrak{p}_i \in \mathfrak{P}$ has a colour $c(\mathfrak{p}_i)$ and a 3D position $\mathbf{v}(\mathfrak{p}_i)$ in camera space. First, a depth filter is applied to this point cloud in an attempt to remove background points, by discarding all points where $\mathbf{v}(\mathfrak{p}_i)_z$ is greater than a threshold λ_{depth}. Then a

kinematic model of the robot, coupled with joint angles from encoders, is used to work out the current pose of the robot's arm in camera space. All points which are within a distance of λ_{model} from a geometric model of either the robot's arm or hand are filtered out from the point cloud. In this way, just the points from the object of interest are left. For the arm, a simple cylinder model sufficed. For the hand, a model of the hand built using the techniques presented in Chapter 4 was used. The hand model only represented the hand in its open position, and so there were inaccuracies in the filtering when the hand closed to grip an object. However, it was found that this simplified filtering scheme was good enough for the purposes of the text reading system. An example of the filtering process and resulting object point cloud can be seen in Figure 7.3. For the experimental work done in this chapter the values $\lambda_{depth} = 0.8$m and $\lambda_{model} = 0.1$cm were used.

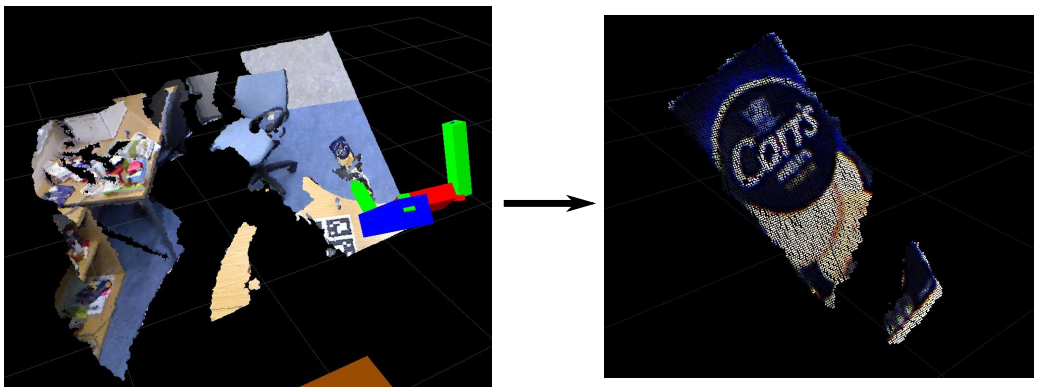


Figure 7.3: By combining a depth filter, a kinematic model of the robot, and a model of the robot's hand, the point cloud from the Kinect is filtered to leave just the object.

7.3.1.2 Gaze Control

A gaze control module was implemented to ensure that the high resolution camera is always pointed towards a target point on the object of interest. The gaze of the robot is set by choosing a pair of neck angles $\boldsymbol{\theta}^{neck} = \{\theta^{pan}, \theta^{tilt}\}$. Given a robot pose defined by a set of joint angles $\boldsymbol{\theta}$, this module chooses neck angles $\boldsymbol{\theta}^{neck}$ so that a target point $\mathbf{t}$ in hand space is moved as close as possible to $\mathbf{c}$, the centre of the image plane of the high resolution camera. Formally this is expressed by Equation 7.1

$$\boldsymbol{\theta}^{neck}_{opt} = \arg\min_{\boldsymbol{\theta}^{neck}} \left\| \mathcal{S}(\boldsymbol{\theta}, \boldsymbol{\theta}^{neck}, \mathbf{t}) - \mathbf{c} \right\|^2 \tag{7.1}$$

where $\mathcal{S}$ is a forward kinematics function which transforms the point $\mathbf{t}$ from hand space to the image space of the high resolution camera. The point chosen for $\mathbf{t}$ is

determined by the active search strategy, and the neck angles $\boldsymbol{\theta}^{neck}$ are found using non-linear optimisation.

7.3.1.3 Text Detection and Recognition

The text detector that was used in this system is a Maximally Stable Extremal Regions (MSER) based text detector, similar to the one presented by Merino-Gracia et al. (2012) and developed by Chris Beck as part of his PhD thesis work (Beck 2015). The text detector first identifies likely letters, and then seeks to group the letters into paragraphs and lines. The output from the text detector is a set of quadrilateral regions Q that identify regions of the image that are likely to contain text. In order to read the text in the quadrilaterals, every $q \in Q$ is mapped to a horizontal rectangle, and passed to an OCR program to be read. The rectangle is passed to the OCR program normally, and then a second time rotated 180°. In this way it is possible to read text which is viewed upside down by the system. Once text has been identified by the OCR program every word is checked with a spellchecker, and words that pass through the spellchecker are classified as the *readable* words found on the object.

7.3.2 Search Strategies

In total, 3 search strategies were developed for the system, ranging from a simple baseline strategy which uses preprogrammed poses, to a deliberative strategy which attempts to systematically locate and then examine each piece of text on an object. Developing a range of strategies allows the performance of a number of approaches to be compared. Also, the system can easily be extended by implementing new search strategies.

7.3.2.1 Preprogrammed Poses

The Preprogrammed Poses (PP) search strategy is a very simple baseline search strategy that can be used to assess the relative performance of the other search strategies. As its name suggests, it consists of moving the robot through a number of preprogrammed poses in order to take the high resolution images. Choosing which poses to use and how many to use, is necessarily a rather arbitrary process, and the set of poses that is optimal in some sense may vary depending upon the objects being examined. For the experiments conducted in this chapter, a set of poses were chosen that gave a good baseline performance on the objects in the dataset, to minimise the possibility that other search strategies could outperform the preprogrammed poses search strategy through pure chance. For gaze control, this strategy sets $\mathbf{t}$ to be a fixed position in hand space that was determined

experimentally to be good for a large number of objects. The zoom level is fixed at z_1, the most zoomed out level, so that as much of the object as possible can be seen in each image.

7.3.2.2 Planar Alignment

The Planar Alignment (PA) search strategy is inspired by the common observation that in general, text is easier to detect and read, if it is presented head on to the camera, so that any effects of perspective distortion are minimised.

To implement this, initially the robot was made to move to the poses defined for the PP search strategy. Before taking an image however, a search is conducted for planes in the object point cloud using the RANSAC algorithm. Planes are only considered if they have support above a threshold λ_r set in the conducted experiments to be 100 points. If one or more suitable planes are found, then the plane whose normal is furthest away from the camera plane is chosen.

An attempt is then made to align the chosen plane with the camera plane. Intuitively, the thinking is that aligning the plane that needs to be moved the furthest, will reveal the most new information.

Non-linear optimisation is used to look for the wrist joint angles $\boldsymbol{\theta}^{wrist}$ that most closely align the chosen plane with the image plane of the high resolution camera plane. The wrist on the BERT robot is a spherical joint so $\boldsymbol{\theta}^{wrist} = \{\theta^{roll}, \theta^{pitch}, \theta^{yaw}\}$. For the plane that is to be aligned r, the normal $\mathbf{n}(r)$ of the plane calculated, and its position $\mathbf{v}(r)$ is defined to be the average position of all the points assigned to the plane as part of the RANSAC step. The goal is then to minimise Equation 7.2

$$\boldsymbol{\theta}^{wrist}_{opt} = \arg\min_{\boldsymbol{\theta}^{wrist}} \mathcal{T}(\boldsymbol{\theta}, \boldsymbol{\theta}^{wrist}, \mathbf{n}(r)).\mathbf{z} \tag{7.2}$$

where $\mathcal{T}$ is a forward kinematics function that transforms $\mathbf{n}(r)$ into camera space, and $\mathbf{z}$ is the z-axis of the high resolution camera.The aim is to minimise the dot product of the transformed plane normal and the camera normal, as this corresponds to them pointing in opposite directions so that the plane directly faces the camera.

Optimisation is only performed for the wrist angles, as it was found that optimising for more angles would often lead to the optimisation process getting stuck in local minima. Once the wrist angles have been found, $\mathbf{t}$ is set equal to $\mathbf{v}(r)$ for gaze control. As with the PP strategy, the zoom level is held at z_1 for all poses.

7.3.2.3 Text Centring and Zooming

The Text Centring and Zooming (TCZ) search strategy is similar to the PA strategy, but whereas that strategy only made decisions informed by the geometry of the object of interest, the TCZ strategy also chooses poses and zoom settings based on the location and size of image regions that the text detection algorithm determines to be text.

As with the PA strategy, the TCZ strategy initially uses the poses determined by the PP strategy. Upon reaching a pose, the TCZ strategy takes a photo at zoom level z_1 and runs the text detector, but the quadrilaterals Q are then *not* passed onto the OCR program. Instead, a pose for each $q \in Q$ is calculated that will centre it in the field of view of the high resolution camera, and which will align the surface of the object in the quadrilateral with the image plane of the high resolution camera. This is done by first fitting a plane r to the surface of the object at the centre of q. The position $\mathbf{v}(r)$ is set by casting a ray from the centre of q into the object point cloud $\mathfrak{P}$ and using the position $\mathbf{v}(\mathfrak{p}_i)$ of the point $\mathfrak{p}_i$ with the smallest depth that is also within a distance λ_{ray} of the ray. Principal Component Analysis (PCA) is then applied to the set of points $O \subseteq \mathfrak{P}$ that are within a distance λ_{ray} of $\mathbf{v}(r)$, and $\mathbf{n}(r)$ is set to be the principle component of O which has the smallest covariance.

For all of the experimental work, $\lambda_{ray} = 1$cm. Due to the fact that the Kinect returns a noisy point cloud that may contain holes, it is possible that either no intersection point will be found for $\mathbf{v}(r)$, or that the number of points in O will be too small to reliably estimate $\mathbf{n}(r)$. In these situations $\mathbf{v}(r)$ is set to be a point which has the same depth as the robot's hand, and which projects onto the centre of q. The normal is set so that $\mathbf{n}(r) = -\mathbf{z}$ i.e. it is assumed that q lies on a plane that is already head on to the camera.

Equation 7.2 is used to find a pose that aligns r with the camera, and Equation 7.1 with $\mathbf{t} = \mathbf{v}(r)$ to centre q in the image plane of the high resolution camera. The maximum zoom level $z \in Z$ for the high resolution camera that allows q to be fully seen is then chosen. In this way a set S of (pose, zoom) pairs $(\boldsymbol{\theta}, z)$, is obtained. One for every quadrilateral $q \in Q$.

Now, it would obviously be possible to move through every element of S, and take an image for every q, but for objects which have lots of regions of text on their surface, this would be very time consuming. It could also be very wasteful, as for every given (pose, zoom) level pair $(\boldsymbol{\theta}, z)$, there are likely to be many quadrilaterals which are visible and adequately covered. Therefore, an attempt is made to minimise the number of (pose, zoom) level pairs that are visited by calculating which quadrilaterals are covered by each (pose, zoom) level pair, and then searching for the smallest $S' \subseteq S$ that captures all $q \in Q$. This is an example of the Set Covering Problem (SCP), a classical problem in combinatorics which is known to be NP-hard (Korte and Vygen 2007). However, the greedy algorithm for

Figure 7.4: The TCZ strategy in action - the top image shows the text quadrilaterals identified at the first pose. Initially 10 (pose, zoom) pairs are generated, one for each quadrilateral. These are pruned back to 2 (pose, zoom) pairs which can each see half of the quadrilaterals, and the bottom two images were obtained from those (pose, zoom) pairs.

solving this problem, whereby at each point the (pose, zoom) level pair that covers the most quadrilaterals that have not been covered so far is chosen, gives good results. So it is by this means that an attempt is made to minimise the number of (pose, zoom) level pairs that are visited. Figure 7.4 shows the result of applying this algorithm.

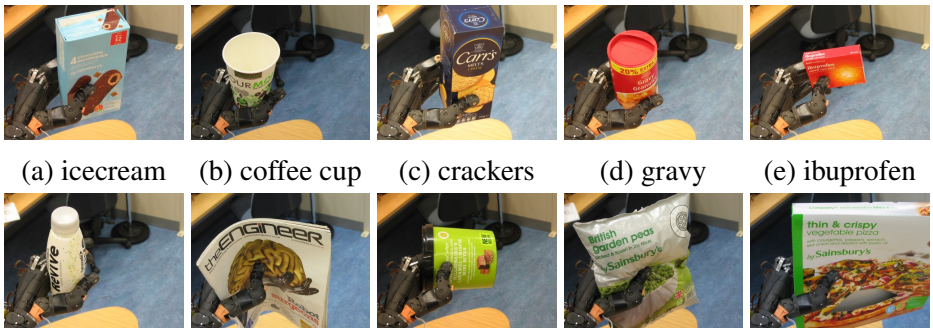
7.4 Experiments

To evaluate the performance of the system, a number of experiments were conducted to determine how well the different search strategies were able to recover valid text from a variety of different objects. The experiments were conducted with a set of 10 objects, as shown in Figure 7.5. An attempt was made to choose objects that were representative of objects that might be found around the home, with a particular emphasis on packages of food, because identifying food items would be a key part of the assisted living application outlined earlier. A variety of object shapes were included in the dataset, which also included 2 non-rigid objects, (a magazine and a bag of peas), in order to demonstrate the wide range of object types which the system can deal with. All of the food packages were emptied prior to the experiments and the contents of the bag of peas were replaced with small bits of foam packaging material. The ground truth text on the objects was manually copied into a text file so that it was possible to evaluate the performance of the system.

With regards to the robotic system used, the kinematic model used for the BERT robot was built using the techniques presented in Chapter 4 along with the hand model built as part of that process.

A series of experiments were conducted in which each object was presented to the robotic system 10 times, giving 100 experiments in total. The objects were put into the robot's hand in a variety of different orientations to reflect the fact that the robot would not always pick up objects in the same way. For each grasp, the robot explored the object using each of the 3 search strategies described in Section 7.3 in turn. In this way it was ensured that no search strategy would be favoured by perhaps having a more fortunate set of grasps. If the robot dropped the object, or if it shifted significantly in the grasp during an experiment, then the experiment was aborted and re-run. A total of 6 poses were chosen for the PP strategy, which means that both the PP strategy and the PA strategy took 6 images for each experiment. The TCZ strategy, which tries to align and correctly focus on every piece of text it finds, took on average 3 images for every 1 image taken by the PP strategy.

The BERT robot was programmed to make its moves slowly, and paused before each image was taken in order ensure that clear images were obtained. In general it took about 20 seconds for the robot to take and process each image looking for



(a) icecream (b) coffee cup (c) crackers (d) gravy (e) ibuprofen

(f) lucozade (g) magazine (h) mini bites (i) peas (j) pizza

Figure 7.5: The 10 objects used in the experiments along with their names.

text, so the overall time taken to examine each object varied depending upon the search strategy involved. The system took about 2 minutes to examine each object using the PP and PA strategies, and about 6 minutes to examine each object using the TCZ strategy.

The averaged results from these experiments are shown in Table 7.1. The TCZ strategy is clearly the best strategy in terms of the recall achieved, as being able to focus in on the text on an object allowed the text detection and recognition pipeline to identify many more readable words. Although obviously, a trade-off may need to be made between the improved recall, and the extra time needed to gather all of the images for the TCZ strategy. None of the strategies achieved particularly good precision, which seems likely to be due to the fact that it is very easy for the OCR program to get a few letters wrong, but then still return a word that passes through the spellchecker. Also, a lot of the incorrect words were substrings of correct words i.e. if the system saw ‘rice’ instead of ‘price’. Combining these substrings as was done by Case et al. (2011), coupled with other error correction methods could help to improve the precision of the system.

The ability of the system to uniquely identify objects by using words found on the objects using the TCZ strategy was also explored. A standard multinomial Naive Bayes classifier (Manning et al. 2007) was trained using the ground truth text from the objects. For each experiment, the words found on the object were passed to the classifier, and the most likely class was then chosen as the classification of the object. The result of object classification is shown in Table 7.2, and shows that the system is able to use the readable text it finds to clearly distinguish between objects in the test set.

Object	Search Strategy	Avg. Num. Correct Words	Avg. Num. Incorrect Words	Precision	Recall
icecream	PP	32.9	30.2	0.57	0.15
	PA	38.1	27.1	0.63	0.17
	TCZ	60.1	69.9	0.50	0.27
coffee cup	PP	4.2	8.0	0.42	0.07
	PA	5.4	7.5	0.43	0.10
	TCZ	12.7	20.6	0.42	0.23
crackers	PP	12.3	30.1	0.28	0.05
	PA	17.9	32.9	0.37	0.08
	TCZ	52.4	93.5	0.36	0.22
gravy	PP	5.9	7.6	0.45	0.04
	PA	7.3	9.7	0.45	0.05
	TCZ	35.7	38.9	0.48	0.27
ibuprofen	PP	6.4	9.9	0.35	0.04
	PA	11.4	13.2	0.43	0.08
	TCZ	58.0	61.3	0.49	0.39
lucozade	PP	3.1	8.0	0.26	0.02
	PA	6.5	11.4	0.43	0.04
	TCZ	27.8	50.0	0.38	0.15
magazine	PP	27.8	28.7	0.50	0.32
	PA	28.4	24.2	0.56	0.32
	TCZ	40.5	53.7	0.45	0.46
mini bites	PP	19.6	21.7	0.48	0.07
	PA	28.8	29.7	0.48	0.10
	TCZ	97.6	131.1	0.43	0.33
peas	PP	27.6	27.5	0.50	0.12
	PA	23.9	27.9	0.45	0.10
	TCZ	59.0	135.1	0.30	0.26
pizza	PP	88.0	62.6	0.53	0.34
	PA	84.1	68.5	0.52	0.33
	TCZ	113.4	103.4	0.52	0.44

Table 7.1: Table showing the result of each strategy averaged over 10 experiments for each object. The best strategy for each object in terms of the recall achieved is highlighted.

	icecream	coffee cup	crackers	gravy	ibuprofen	lucozade	magazine	mini bites	peas	pizza
icecream	10	0	0	0	0	0	0	0	0	0
coffee cup	0	10	0	0	0	0	0	0	0	0
crackers	0	0	10	0	0	0	0	0	0	0
gravy	1	0	0	9	0	0	0	0	0	0
ibuprofen	0	0	0	0	10	0	0	0	0	0
lucozade	0	0	0	0	0	10	0	0	0	0
magazine	0	0	0	0	0	0	10	0	0	0
mini bites	0	0	0	0	0	0	0	10	0	0
peas	2	0	0	0	0	0	0	0	8	0
pizza	0	0	0	0	0	0	0	0	0	10

Table 7.2: Confusion matrix obtained when attempting to classify each of the objects using the words obtained from the TCZ strategy.

7.5 Summary

A robotic system has been presented which can manipulate and explore objects presented to it in order to detect and read any text that may be on the objects. Experiments have shown that the system is able to identify large amounts of text on a variety of objects, and that the text obtained is sufficient to be able to classify and identify objects. A variety of search strategies have been implemented for the system, and experimentally, it has been shown that the strategies which actively consider where text may be located, and how it may be oriented, are much more likely to be able to locate readable text on the object.

The text obtained from an object is an important modality for understanding both what the object is, and how the object can be used. The classification results indicate that the readable text found by the system could be used to build models of previously unseen objects, or to augment models formed from more traditional features, such as visual features.

Chapter 8

Conclusion

This final chapter looks back at, and discusses, the work presented in the thesis. The main contributions of the thesis are restated, and the chapter concludes with a discussion of possible future directions that could be looked at in order to take this work further.

8.1 Thesis Summary

This thesis has looked at how a robotic system can use vision and exploratory actions in order to progressively build models of itself, and of objects that it encounters in the world around it.

Following the introduction which framed and motivated this work, Chapter 2 provided background by reviewing relevant areas of the literature. The review looked at evidence that has been found to support the existence of internal models within biological systems, especially for motor control. The review continued by looking at the classical methods used to calibrate kinematic models for robotic systems, as well as looking at a number of existing works which have presented methods for building models of robotic systems and of objects in the world around a robot. Chapter 2 concluded by making the argument that this thesis stands alone and is original because of the way in which it seeks to enable robotic systems to progressively build models of themselves, and of objects in the world around them using a layered approach. Throughout the thesis this has been done by devising a series of exploratory actions which allow a robotic system to obtain the information required to build those models.

Chapter 3 described the robotic systems which were used for the experimental work in this thesis. An overview was given of the software which was developed to support this work, and an assessment was made of the repeatability and accuracy of the robotic systems to allow the performance of any models built of the robots

to be evaluated.

In Chapter 4 it was shown how a robot can start from a state of very limited knowledge about itself, essentially just an ordered list of the actuators that make up its body, but can then perform a series of actions that give it the information required to build a series of models describing itself. The system presented was able to identify the robot's hand using a waving motion, and then, having found the hand, construct a 3D model of the hand using a Microsoft Kinect depth camera. Having a 3D model of the hand allowed its motion to be tracked as exploratory motions were made with each of the actuators in turn. Chapter 4 further showed that by tracking the motion of the hand in camera space, it was possible to construct a kinematic model of the robot.

The models that a robot builds of itself enable the robot to perform actions that were not possible before, and in Chapter 5 it was shown how the kinematic model built by the robot could be used to autonomously home its joint angles upon start up, using just vision.

Moving on from building models of itself, Chapter 6 demonstrated how the same concept of performing exploratory actions could be used by a robot in order to build articulated models of flattened cardboard boxes. Hypotheses about the presence and location of hinges can be tested by having the robot attempt to exercise the hinge, and then visual and haptic feedback can be used to extract information from the experience to improve the model of the box.

Finally, Chapter 7 looked at how a robot could use exploratory actions in order to locate and read any text that might be present on an object. This last piece of work provides a good example of how models of a robot or of an object do not have to be restricted to 'spatial' models such as a 3D representation, or a kinematic model, but in fact can be any collection of characteristics that describe and define the object in some way.

8.2 Contributions

The main contributions of this research are as follows:

- A method for enabling a robotic system to build a kinematic model of itself using a series of exploratory actions was described and evaluated. The work is novel due to the fact that, compared with other methods, the robot can start from a more limited state of knowledge. Also, the method does not require any fiducial markers to be attached to the robot.

- A novel method was presented for homing the joints of a robot using just vision, i.e. traditional methods such as absolute encoders or limit switches were not required.

- The concept of an exploratory joint space that a robot may safely move in whilst attempting to home itself was described.
- A novel edge based tracker was presented that incorporates colour cues, and which can be used to track the motion of articulated flat pack cardboard boxes.
- A hinge exploration strategy that combines both visual and haptic feedback in order to evaluate the presence or absence of a hinge was described and evaluated.
- A novel robotic text reading system was described; it was shown how exploratory actions could be used to find and read any text which may be on an object.

8.3 Future Work

There are number of ways in which the work of this thesis could be extended and taken further.

Chapter 4 presented a method that enables a robotic system to autonomously build a kinematic model of itself, starting with only a small amount of prior information. One of the more exciting applications of this work is that it makes progress towards software that could dramatically reduce the time and effort required to commission a new robot. Rather than a group of system designers having to produce models specific to the new robot, an advanced software agent could instead be loaded onto the robot. It would then undertake a learning process to discover the capabilities of the robot, possibly mimicking the process that a human uses to learn the capabilities of its body as a child.

It is interesting to consider what might be the minimal amount of information and set of sensors that a robot would require in order to build a kinematic model of itself. For the BERT robot used in Chapter 4, the main problem is that it is not possible to make unconstrained exploratory motions with the robot. This is because for some possible motions, there is a risk that the robot's arms and hands will collide with each other or with the robot's torso, damaging the robot. Therefore, it is always necessary to provide the robot with a 'safe' range of angles that it can drive its actuators through. This requirement could be avoided if the robot was equipped with a touch sensitive skin, which would allow it to detect self collisions. Projects such as ROBOSKIN (Schmitz and Maiolino 2011), show that this type of technology is readily available. Incorporating a sense of touch with the vision based techniques presented here could provide a range of interesting options for the autonomous kinematic identification of robotic systems.

The visual homing system presented in Chapter 5 does show a lot of promise, but it can take a while for the system to view the hand of the robot in enough positions to accurately home the robot. In future work, ways to simplify and speed up the homing process, (whilst maintaining or increasing the accuracy of the system), will be looked for.

Chapter 6 provided a method by which a robotic system could autonomously build an articulated model of a flat pack cardboard box placed in front of it, by systematically investigating the location of potential hinges and by attempting to manipulate them. In the future, the hope is to extend this work by improving the speed and accuracy of the tracking system, and by improving the ability of the robot to fold and manipulate the cardboard boxes. In its current configuration, it does not seem to be possible to get the Baxter robot used for this work to fold cardboard boxes using vacuum grippers or parallel jaw grippers due to some of the complex moves required. An interesting line of research could be to investigate and design some more capable manipulators that better allow the robotic system to exploit the information it has gained from its exploratory motions.

Finally Chapter 7 presented a robotic text reading system which described exploratory actions in the form of search strategies which could be used to efficiently locate text on the surface of an object held in the robot's end effector. In future work the aim is to explore a number of different search strategies to improve the performance of the system. A wide variety of approaches are possible, and it is possible to imagine search strategies which seek to avoid interference from specular reflections formed by nearby light sources, or which attempt to use bimanual manipulation in order to explore previously unseen parts of the object of interest.

So in conclusion, it is hoped that this thesis stands well as an exploration of different methods that may be used to incorporate autonomous model building into robotic systems. Clearly this is a field with a lot of exciting directions for future research, and it is a field in which mush progress must be made if the dream of fully autonomous robotic systems is to be realised.

Bibliography

ABB (2012). *IRB 120 Datasheet*. https://www.abb.com/ROB0149EN_D_LR.pdf.

Angelova, A. et al. (2006). Slip prediction using visual information. In: *Robotics: Science and Systems*.

Asfour, T. et al. (2006). ARMAR-III: An integrated humanoid platform for sensory-motor control. In: *IEEE-RAS International Conference on Humanoid Robots*, pp. 169–175.

Atkeson, C. G. et al. (1997). Locally weighted learning for control. In: *Artificial Intelligence Review* 11, pp. 75–113.

Aydemir, A. and K. Sjoo (2011). Search in the real world: Active visual object search based on spatial relations. In: *IEEE International Conference on Robotics and Automation*, pp. 2818–2824.

Bajcsy, R. (1988). Active perception. In: *Proceedings of the IEEE* 76.8, pp. 966–1005.

Ballard, D. H. (1991). Animate vision. In: *Artificial Intelligence* 48.1, pp. 57–86.

Barrow, H. G. et al. (1977). Parametric correspondence and chamfer matching: Two new techniques for image matching. In: *International Joint Conference on Artificial Intelligence*, pp. 659–663.

Beck, C. (2015). Text localization and understanding to aid efficient extraction. PhD thesis.

Beck, C. et al. (2014). Text Line Aggregation. In: *International Conference on Pattern Recognition Applications and Methods*, pp. 393–401.

Bellamy, M. C. (2006). Wet, dry or something else? In: *British Journal of Anaesthesia* 97.6, pp. 755–757.

Bennett, D. J. et al. (1992). Kinematic calibration by direct estimation of the Jacobian matrix. In: *IEEE International Conference on Robotics and Automation*, pp. 351–357.

Bernardini, F. and H. Rushmeier (2002). The 3D model acquisition pipeline. In: *Computer Graphics Forum* 21.2, pp. 149–172.

Besl, P. and N. McKay (1992). A method for registration of 3-D shapes. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2, pp. 239–256.

Birchfield, S. (1998). Elliptical head tracking using intensity gradients and color histograms. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 232–237.

Bongard, J. et al. (2006). Resilient machines through continuous self-modeling. In: *Science* 314.5802, pp. 1118–1121.

Botvinick, M. and J. Cohen (1998). Rubber hands 'feel' touch that eyes see. In: *Nature* 391.6669, p. 756.

Brooks, R. A. (1991). Intelligence without representation. In: *Artificial Intelligence*. Artificial Intelligence 47.1. Ed. by D Kirsh, pp. 139–159.

Brooks, R. A. et al. (1999). The Cog project: Building a humanoid robot. In: *Computation for metaphors, analogy, and agents*, pp. 52–87.

Broun, A. and M. Studley (2011). Low cost automatic object segmentation by detecting a signature motion within an optical flow signal. In: *IEEE International Conference on Industrial Technology*, pp. 342–347.

Broun, A. et al. (2012). Building a kinematic model of a robot's arm with a depth camera. In: *Towards Autonomous Robotic Systems*, pp. 105–116.

Broun, A. et al. (2013). Robotic manipulation strategies for detecting and reading text. In: *ICRA Interactive Perception Workshop*.

Broun, A. et al. (2014a). Bootstrapping a robot's kinematic model. In: *Robotics and Autonomous Systems* 62.3, pp. 330–339.

Broun, A. et al. (2014b). Visual homing of an upper torso humanoid robot using a depth camera. In: *Towards Autonomous Robotic Systems*, pp. 114–126.

Broun, A. et al. (In preparation). Active visual search strategies for robotic text reading. In:

Caflisch, R. E. (1998). Monte Carlo and quasi-Monte Carlo methods. In: *Acta Numerica* 7, p. 1.

Carlson, T. A. et al. (2010). Rapid assimilation of external objects into the body schema. In: *Psychological science* 21.7, pp. 1000–1005.

Case, C. et al. (2011). Autonomous sign reading for semantic mapping. In: *IEEE International Conference on Robotics and Automation*, pp. 3297–3303.

Chen, H. et al. (2011). Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In: *International Conference on Image Processing*, pp. 2609–2612.

Chen, Y. and G. Medioni (1991). Object modeling by registration of multiple range images. In: *IEEE International Conference on Robotics and Automation*, pp. 2724–2729.

Choi, C. and H. I. Christensen (2012). Robust 3D visual tracking using particle filtering on the special Euclidean group: A combined approach of keypoint and edge features. In: *International Journal of Robotics Research* 31.4, pp. 498–519.

Christiansen, A. D. et al. (1990). Learning reliable manipulation strategies without initial physical models. In: *IEEE International Conference on Robotics and Automation*, pp. 1224–1230.

Cignoni, P. et al. (2008). Meshlab: an open-source 3d mesh processing system. In: *ERCIM News* 73, pp. 45–46.

Dearden, A. and Y. Demiris (2005). Learning forward models for robots. In: *International Joint Conference on Artificial Intelligence*, pp. 1440–1445.

Drummond, T. and R. Cipolla (2002). Real-time visual tracking of complex structures. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7, pp. 932–946.

D'Souza, A. et al. (2001). Learning inverse kinematics. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 298–303.

Edsinger, A. L. (2007). Robot manipulation in human environments. PhD thesis.

Epshtein, B. et al. (2010). Detecting text in natural scenes with stroke width transform. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2963–2970.

Fitzpatrick, P. and G. Metta (2003). Grounding vision through experimental manipulation. In: *Philosophical transactions* 361.1811, pp. 2165–2185.

Gablonsky, J. M. and C. T. Kelley (2001). A locally-biased form of the DIRECT algorithm. In: *Journal of Global Optimization* 21.1, pp. 27–37.

Griffith, S., J. Sinapov, et al. (2009). Toward interactive learning of object categories by a robot: A case study with container and non-container objects. In: *IEEE International Conference on Development and Learning*, pp. 1–6.

Griffith, S., V. Sukhoy, et al. (2012). Object categorization in the sink : Learning behavior – grounded object categories with water. In: *ICRA Workshop on Semantic Perception, Mapping, and Exploration*.

Guennebaud, G., B. Jacob, et al. (2010). *Eigen v3*. http://eigen.tuxfamily.org.

Hartenberg, R. S and J. Denavit (1955). A kinematic notation for lower-pair mechanisms based on metrics. In: *Transactions of the ASME. Journal of Applied Mechanics* 22, pp. 215–221.

Hart, J. W. and B. Scassellati (2011). A robotic model of the ecological self. In: *IEEE-RAS International Conference on Humanoid Robots*, pp. 682–688.

Hayati, S. and M. Mirmirani (1985). Improving the absolute positioning accuracy of robot manipulators. In: *Journal of Field Robotics* 2.4, pp. 397–413.

Henry, P. et al. (2010). RGB-DMapping: Using depth cameras for dense 3D modeling of indoor environments. In: *International Symposium on Experimental Robotics*, pp. 477–491.

Hersch, M. et al. (2008). Online learning of the body schema. In: *International Journal of Humanoid Robotics* 5.2, pp. 161–181.

Hinterstoisser, S., C. Cagniart, et al. (2012). Gradient response maps for real-time detection of textureless objects. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.5, pp. 876–888.

Hinterstoisser, S., S. Holzer, et al. (2011). Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In: *IEEE International Conference on Computer Vision*, pp. 858–865.

Hoffmann, M. et al. (2010). Body schema in robotics: A review. In: *IEEE Transactions on Autonomous Mental Development* 2.4, pp. 304–324.

Hollerbach, J. M., L. Giugovaz, et al. (1993). Screw axis measurement for kinematic calibration of the Sarcos dextrous arm. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1617–1621.

Hollerbach, J. M. and D. M. Lokhorst (1995). Closed-loop kinematic calibration of the RSI 6-DOF hand controller. In: *IEEE Transactions on Robotics and Automation* 11.3, pp. 352–359.

Hollerbach, J. M. and C. W. Wampler (1996). The calibration index and taxonomy for robot kinematic calibration methods. In: *International Journal of Robotics Research* 15.6, pp. 573–591.

Hutchinson, S. et al. (1996). A tutorial on visual servo control. In: *IEEE Transactions on Robotics and Automation* 12.5, pp. 651–670.

ISO 9283 (1998). *Manipulating industrial robots performance criteria and related test methods*. International Standardization Organisation.

Jain, J. R. and A. K. Jain (1981). Displacement measurement and its application in interframe image coding. In: *IEEE Transactions on Communications* 29.12, pp. 1799–1808.

Jin, Y. and M. Xie (2000). Vision guided homing for humanoid service robot. In: *International Conference on Pattern Recognition*, pp. 511–514.

Johnson, S. G. (2010). *The NLopt nonlinear-optimzation package*. URL: `http://ab-initio.mit.edu/nlopt`.

Kajita, S. and B. Espiau (2008). *Springer handbook of robotics*. Springer. ISBN: 978-3-540-23957-4.

Katz, D. and O. Brock (2008). Manipulating articulated objects with interactive perception. In: *IEEE International Conference on Robotics and Automation*, pp. 272–277.

Kawato, M. (1999). Internal models for motor control and trajectory planning. In: *Current Opinion in Neurobiology* 9.6, pp. 718–727.

Kennedy, J. and R. Eberhart (1995). Particle swarm optimization. In: *International Conference on Neural Networks*. Vol. 4, pp. 1942–1948.

Kijewski, W. (1999). *SI units, conversion & measurement skills*. ISBN: 0620340584.

Klein, G. and D. Murray (2006). Full-3D edge tracking with a particle filter. In: *British Machine Vision Conference*, pp. 1119–1128.

Klingbeil, E. et al. (2011). Grasping with application to an autonomous checkout robot. In: *IEEE International Conference on Robotics and Automation*, pp. 2837–2844.

Korte, B. and J. Vygen (2007). *Combinatorial optimization: Theory and algorithms*. 2nd. Springer.

Krainin, M. et al. (2011). Manipulator and object tracking for in-hand 3D object modeling. In: *International Journal of Robotics Research* 30.11, pp. 1311–1327.

Lepetit, V. and P. Fua (2005). Monocular model-based 3D tracking of rigid objects. In: *Foundations and Trends in Computer Graphics and Vision* 1, pp. 1–89.

Létourneau, D. et al. (2004). Autonomous mobile robot that can read. In: *EURASIP Journal on Advances in Signal Processing* 2004.17, pp. 2650–2662.

Liang, J. et al. (2005). Camera-based analysis of text and documents: A survey. In: *International Journal on Document Analysis and Recognition* 7.2, pp. 84–104.

Liu, M. Y. et al. (2010). Fast directional chamfer matching. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1696–1703.

Lowe, D. G. (1991). Fitting parameterized three-dimensional models to images. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.May 1991, pp. 441–450.

Low, K. L. (2004). *Linear least-squares optimization for point-to-plane ICP surface registration*. Tech. rep. University of North Carolina, pp. 2–4.

Lucas, S. M. et al. (2003). ICDAR 2003 robust reading competitions. In: *International Conference on Document Analysis and Recognition*, pp. 682–687.

Manning, C. D et al. (2007). *Introduction to information retrieval*. Cambridge University Press. ISBN: 9780521865715.

Maravita, A. and A. Iriki (2004). Tools for the body (schema). In: *Trends in Cognitive Sciences* 8.2, pp. 79–86.

Marjanovic, M. et al. (1996). Self-supervised visually-guided pointing for a humanoid robot. In: *From Animals to Animats 4: International Conference on Simulation of Adaptive Behavior*, pp. 35–44.

Meltzoff, A. N. and M. K. Moore (1997). Explaining facial imitation: A theoretical model. In: *Early Development and Parenting* 6, pp. 179–192.

Merino-Gracia, C. et al. (2012). A head-mounted device for recognizing text in natural scenes. In: *Camera-Based Document Analysis and Recognition*, pp. 29–41.

Miall, R. C. et al. (1993). Is the cerebellum a smith predictor? In: *Journal of Motor Behavior* 25.3, pp. 203–216.

Mirmehdi, M. et al. (2003). A non-contact method of capturing low-resolution text for OCR. In: *Pattern Analysis and Applications* 6.1, pp. 12–21.

Mooring, B. W. et al. (1991). *Fundamentals of manipulator calibration*. Wiley. ISBN: 0-471-50864-0.

Munoz-Salinas, R. and S. Garrido-Jurado (2013). *ArUco: a minimal library for Augmented Reality applications based on OpenCv*. URL: `http://sourceforge.net/projects/aruco/`.

Murphy, Robin (2000). *Introduction to AI robotics*. MIT Press. ISBN: 0262133830.

Murray, R. M. et al. (1994). *A mathematical introduction to robotic manipulation*. CRC Press. ISBN: 9780849379819.

Nguyen-Tuong, D. and J. Peters (2011). Model learning for robot control: A survey. In: *Cognitive Processing* 12.4, pp. 319–340.

Nikulin, M. S. (2011). Hellinger distance. In: *Encyclopedia of Mathematics*.

Nocedal, J. and S. Wright (2006). *Numerical optimization*. 2nd. Springer.

Oliphant, T. E. (2007). *SciPy: Open source scientific tools for Python*. URL: `http://www.scipy.org/`.

Panin, G. et al. (2008). Robust contour-based object tracking integrating color and edge likelihoods. In: *Vision, Modeling, and Visualization Conference*, pp. 227–234.

Pfister, H. et al. (2000). Surfels: Surface elements as rendering primitives. In: *Conference on Computer Graphics and Interactive Techniques*, pp. 335–342.

Posner, I. et al. (2010). Using text-spotting to query the world. In: *International Conference on Intelligent Robots and Systems*, pp. 3181–3186.

Ramos-Garijo, R. et al. (2003). An autonomous assistant robot for book manipulation in a library. In: *IEEE International Conference on Systems, Man and Cybernetics*, pp. 3912–3917.

RethinkRobotics (2015). *Baxter Datasheet*. http://www.rethinkrobotics.com/baxter/.

Rochat, P. (1998). Self-perception and action in infancy. In: *Experimental Brain Research* 123.1, pp. 102–109.

Ruiz, J. A. Á. et al. (2013). Active scene text recognition for a domestic service robot. In: *RoboCup 2012: Robot Soccer World Cup XVI*, pp. 249–260.

Rusinkiewicz, S. and M. Levoy (2001). Efficient variants of the ICP algorithm. In: *IEEE International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152.

Santos, J. et al. (2010). Sensor-based self-calibration of the iCub's head. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5666–5672.

Schaal, S. et al. (2002). Scalable techniques from nonparameteric statistics for real-time robot learning. In: *Applied Intelligence* 17.1, pp. 49–60.

Schmitz, A. and P. Maiolino (2011). Methods and technologies for the implementation of large-scale robot tactile sensors. In: *IEEE Transactions on Robotics* 27.3, pp. 389–400.

Scholz, J. et al. (2011). Cart pushing with a mobile manipulation system: Towards navigation with moveable objects. In: *IEEE International Conference on Robotics and Automation*, pp. 6115–6120.

Schulman, J. et al. (2013). Tracking deformable objects with point clouds. In: *IEEE International Conference on Robotics and Automation*, pp. 1130–1137.

Sinapov, J. et al. (2011). Interactive object recognition using proprioceptive and auditory feedback. In: *International Journal of Robotics Research* 30.10, pp. 1250–1262.

Smith, R. (2007). An overview of the tesseract OCR engine. In: *International Conference on Document Analysis and Recognition*, pp. 629–633.

Stone, H. W. et al. (1986). Arm signature identification. In: *IEEE International Conference on Robotics and Automation*, pp. 41–48.

Sturm, J., C. Plagemann, et al. (2009). Body schema learning for robotic manipulators from visual self-perception. In: *Journal of Physiology-Paris* 103.3, pp. 220–231.

Sturm, J., V. Pradeep, et al. (2009). Learning kinematic models for articulated objects. In: *International Joint Conference on Artificial Intelligence*, pp. 1851–1856.

Sudderth, E. B. et al. (2004). Visual hand tracking using nonparametric belief propagation. In: *Computer Vision and Pattern Recognition Workshop*, p. 189.

Suzuki, S. and K. Abe (1985). Topological structural analysis of digitized binary images by border following. In: *Computer Vision, Graphics, and Image Processing* 30.1, pp. 32–46.

Taiana, M. and J. Nascimento (2008). Sample-based 3D tracking of colored objects: a flexible architecture. In: *British Machine Vision Conference*, pp. 1–10.

Tanaka, M. and H. Goto (2007). Autonomous text capturing robot using improved DCT feature and text tracking. In: *International Conference on Document Analysis and Recognition*, pp. 1178–1182.

Tanaka, M. and H. Goto (2008). Text-tracking wearable camera system for visually-impaired people. In: *International Conference on Pattern Recognition*, pp. 1–4.

Tsotsos, J. K. (1992). On the relative complexity of active vs. passive visual search. In: *International Journal of Computer Vision* 7.2, pp. 127–141.

Vacchetti, L. et al. (2004). Combining edge and texture information for real-time accurate 3D camera tracking. In: *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 48–57.

Vignemont, F. (2009). Body schema and body image - pros and cons. In: *Neuropsychologia* 48.3, pp. 669–680.

Weise, T. et al. (2009). In-hand scanning with online loop closure. In: *IEEE International Conference on Computer Vision*, pp. 1630–1637.

Whitney, D. E. et al. (1986). Industrial robot forward calibration method and results. In: *Journal of Dynamic Systems, Measurement, and Control* 108.1, pp. 1–8.

Wolpert, D. M. et al. (1998). Internal models in the cerebellum. In: *Trends in Cognitive Sciences* 2.9, pp. 338–347.

Wong, L. L. S. et al. (2013). Manipulation-based active search for occluded objects. In: *IEEE International Conference on Robotics and Automation*, pp. 2814–2819.

Yao, C. et al. (2012). Detecting texts of arbitrary orientations in natural images. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1083–1090.

Zacharias, F. et al. (2007). Capturing robot workspace structure: Representing robot capabilities. In: *IEEE International Conference on Intelligent Robots and Systems*, pp. 3229–3236.