

Robotic Control for the Manipulation of 3D Deformable Objects

by

Stephen Rowlands

Thesis submitted in partial fulfillment of the requirements for the
M.A.Sc. degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

© Stephen Rowlands, Ottawa, Canada, 2021

Abstract

Robotic grasping and manipulation of three-dimensional deformable objects is a complex task that currently does not have robust and flexible solutions. Deformable objects include a wide variety of elastic and inelastic objects that change size and shape during manipulation. The development of adaptable methods for grasping and autonomously controlling the shape of three-dimensional deformable objects will benefit many commercial applications, including shaping parts for assembly in manufacturing, manipulating food for packaging and controlling tissues during robotic surgery. Controlling a deformable object to a desired shape requires first choosing contact points on the object’s surface. Next, the robotic hand is positioned in the correct position and orientation to grasp and deform the object. After deformation, the object is assessed to evaluate the quality of the shape control procedure. In many cases, this process is completed without knowing the object’s properties or behaviour before deformation.

This work proposes and implements the framework for a robotic arm and hand system to control the shape of a previously unseen deformable object autonomously. Significant original contributions are made in developing an original algorithm to plan contact points on a three-dimensional object for grasping and shape control. This research uses a novel object representation to reduce the dimensionality of the deformable object manipulation problem. A path planning algorithm guides the robot arm to the optimal valid grasp pose to deform the object at the determined contact points. Additional contributions include developing a multi-view assessment strategy to determine the quality of the deformation towards the desired shape.

The system completes the objectives using depth and colour images captured from a single point of view to locate and identify a previously unseen three-dimensional object within a robotic workspace. After estimating the unknown object’s geometry, initial grasp contact points are planned to control the object to the desired shape. The grasp points are used to plan and execute a collision-free trajectory for the robot manipulator to place the robotic hand in the optimal position and orientation to grasp and deform the object. After the deformation is complete, the object is moved to a variety of assessment positions to determine the success of the shape control procedure. The system is validated experimentally on a variety of deformable three-dimensional objects.

Acknowledgements

I would like to thank my co-supervisors, Dr. Pierre Payeur and Dr. Ana-Maria Cretu, and everyone else who has supported me throughout this process.

Table of Contents

List of Tables	vii
List of Figures	viii
List of Algorithms	xii
List of Acronyms	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Organization	3
2 Literature Review	5
2.1 Object Detection and Segmentation	6
2.1.1 Traditional Segmentation Techniques	6
2.1.2 RGB-D Augmented Segmentation Techniques	8
2.2 Object Completion	10
2.2.1 Symmetry and Primitive Object Methods	11
2.2.2 Object Recognition Methods	12
2.2.3 Learning-Based Methods	12
2.3 Robot Manipulator Motion Planning	15
2.3.1 Search-Based Methods	16
2.3.2 Sampling-Based Methods	17
2.3.3 Optimization-Based Methods	20
2.4 Robotic Manipulation of Deformable Objects	20
2.4.1 Grasping and Deformation Control of 3D Objects	21

2.4.2	Shape Estimation and Deformation Tracking	23
2.5	Summary and Further Development of Methodologies	25
3	Design Considerations	26
3.1	Proposed Approach	26
3.2	Equipment	29
3.2.1	Intel RealSense Depth Camera	30
3.2.2	CRS F3 Robot Manipulator	31
3.2.3	Barrett Hand	36
3.3	Objects	38
4	Robotic Control of Deformable Objects	40
4.1	Initialization	40
4.1.1	Coordinate System Alignment	40
4.1.2	Object Segmentation Using RGB-D Data	44
4.1.3	Object Completion	59
4.2	Planning	63
4.2.1	Grasp Selection	63
4.2.2	Robot Pose and Trajectory Planning	86
4.3	Execution	98
4.3.1	Shape Control	98
4.3.2	Quality Assessment	99
4.4	Summary	109
5	Experimental Results and Analysis	111
5.1	End-to-End System Results	111
5.1.1	Experimental Test Case 1	112
5.1.2	Experimental Test Case 2	116
5.1.3	Experimental Test Case 3	119
5.1.4	Experimental Test Case 4	121
5.2	Discussion	123
6	Conclusion	127
6.1	Summary	127
6.2	Contributions	128
6.3	Future Work	129

References	131
APPENDICES	142
A Supplementary Results	143

List of Tables

3.1	Denavit-Hartenberg parameters for the CRS F3 manipulator.	32
3.2	Denavit-Hartenberg parameters for all fingers of the Barrett Hand.	37
3.3	Description of the set of deformable objects.	39
4.1	Position data from camera (Q) and robot (P) during coordinate system alignment and the resulting transformed camera coordinates ($T_{cam/rob} Q$). .	43
4.2	Mean square error for least-squares fit of the segmented point cloud to primitive shapes.	61
4.3	Numerical results for object completions shown in Figure 4.11.	63
4.4	Variables for valid grasping configurations of the robotic hand.	80
4.5	Different criteria that can be used to select third finger position.	82
4.6	JT-RRT parameters used in experiments.	93
4.7	Time to plan path to goal using different methods and parameters.	96
5.1	Fingertip positional error ($n = 14$).	124
5.2	Mean quantitative results for shape control ($n = 13$)	124
5.3	Mean computation time for different components of the grasping system ($n = 14$).	125

List of Figures

2.1	Examples of traditional segmentation techniques.	8
2.2	Example of simple search-based motion planning algorithms.	16
2.3	Example of sampling-based motion planning algorithms.	19
3.1	Proposed framework for a global approach to manipulation of deformable objects.	27
3.2	Detailed framework for robotic control and manipulation of deformable objects.	28
3.3	Physical robot infrastructure used in the proposed solution.	29
3.4	System diagram for the proposed robotic manipulation system.	30
3.5	Images of the robot and workspace captured from the Realsense D435i camera.	31
3.6	The CRS F3 robot manipulator with joint axes identified.	32
3.7	The BH8-280 three-fingered gripper with joint axes identified.	36
3.8	Objects used in shape control experiments.	38
4.1	Colour and colourized depth images of the CRS F3 robot with a ball of known radius fixed to the palm of the Barrett Hand for coordinate alignment.	41
4.2	Circular masks and edge types for probability boundary detection.	46
4.3	Cropped colour and colourized depth images for objects in a grasping area.	49
4.4	Edge images and probability boundary map for cropped object images.	50
4.5	Example of polar image representation for fast level sets.	51
4.6	Switch-in and switch-out algorithms illustrated in three stages.	53
4.7	Polar filtered boundary probability with edge terms.	56
4.8	Object segmentation using fast level sets with different values of γ	57
4.9	Segmentation results for boundary probability augmented fast level sets.	58
4.10	3D object completion results (blue dots) for fitting a (a) box, (b) sphere and (c) cylinder.	61

4.11	3D object completion results using primitive shape fitting.	63
4.12	Hand configuration for grasping an object with initial contact points identified.	65
4.13	Example of an estimated initial undeformed 3D object geometry alongside the desired 3D shape with the distances from the centre of the object to the surface identified.	68
4.14	Transformations from 3D Cartesian space to a surface image representation.	70
4.15	Surface images for the initial undeformed 3D object, the desired 3D shape and the difference surface between them.	71
4.16	Examples of 2D contours that are valid and invalid.	72
4.17	Finding local maxima on a difference surface image.	73
4.18	Relevant references frames and transformations for grasp points selection.	74
4.19	The object coordinate system based on candidate grasping points identified from the difference surface image.	75
4.20	Possible third grasping point given two grasping points and the hand's kinematic constraints for $0 < \alpha \leq \frac{\pi}{2}$	78
4.21	Possible third grasping point given two grasping points and the hand's kinematic constraints for $\frac{\pi}{2} < \alpha < \pi$	79
4.22	Possible third grasping point given two grasping points and the hand's kinematic constraints for $-\pi < \alpha < -\frac{\pi}{2}$	79
4.23	Possible third grasping point given two grasping points and the hand's kinematic constraints for $-\frac{\pi}{2} \leq \alpha < 0$	80
4.24	Barrett Hand placed in the grasping pose to grasp and deform a 3D object, with coordinate systems and variables identified.	84
4.25	Additional examples of initial undeformed 3D objects, a desired 3D shape to deform towards and the resulting difference surface image between them.	86
4.26	A part model of the CRS F3 robot manipulator illustrated with and without virtual collision spheres.	88
4.27	Snapshots of the robot moving from an initial pose to a grasping approach position.	94
4.28	Joint trajectories for the robot manipulator moving from an initial pose to a grasping approach pose.	95
4.29	Motion pipeline flowchart for moving the manipulator from its home position to the grasping pose.	97
4.30	Example of the robot motion pipeline for deformable grasping and shape control.	98
4.31	Object grasping and deformation at initial contact locations.	99

4.32	Deformed object at each assessment pose with the segmentation contour identified.	102
4.33	Two views of the complete point cloud from multi-view assessment registration, with fingertip positions shown in red.	104
4.34	Visual assessment of deformation and shape control.	105
4.35	Estimated initial object shape and desired shape requiring significant deformation.	106
4.36	Object presented at all assessment poses with segmentation contour identified and complete point cloud with fingertips positions from two perspectives.	107
4.37	Visual assessment of deformation and shape control for desired shape requiring significant deformation.	108
5.1	Initial setup for grasping and deforming a rubber ball.	113
5.2	Results from the initialization phase of robotic grasping system for rubber ball experiment.	113
5.3	Results from the planning phase of robotic grasping system for the rubber ball experiment.	114
5.4	Images of the robot as it moves from the approach pose into the grasping pose and deforms the object.	115
5.5	Results from the execution phase of robotic grasping system for the rubber ball experiment.	115
5.6	Initial setup for grasping and deforming a rubber ball fixed to a vertical surface.	116
5.7	End-to-end results for grasping and deforming a rubber ball fixed to a vertical surface.	118
5.8	Initial setup for grasping and deforming a sponge.	119
5.9	End-to-end results for grasping and deforming a sponge placed on a tabletop surface behind an obstacle.	120
5.10	Initial setup for grasping and deforming a foam cylinder.	121
5.11	End-to-end results for grasping and deforming a foam cylinder placed on a tabletop surface behind an obstacle.	122
5.12	Example of unsuccessful grasp by robot.	123
A.1	Initial setup for grasping and deforming an object, supplementary results (a) to (c) for objects that require small deformations compared to the desired 3D shape.	143
A.2	Deformation assessment for supplementary results (a) to (c).	144

A.3	Initial setup for grasping and deforming an object, supplementary results (d) to (g) for objects that require large deformations compared to the desired 3D shape.	144
A.4	Deformation assessment for supplementary results (d) to (g).	145

List of Algorithms

1	Probability boundary detection	47
2	Switch-in algorithm	54
3	Switch-out algorithm	54
4	Boundary probability augmented fast level sets segmentation algorithm . . .	57
5	Find all possible locations for third grasping point	81
6	Potential grasp points selection method	85
7	Structure for the JT-RTT algorithm	90
8	Extend random	91
9	Extend Jacobian transpose	92
10	Find optimal assessment poses	101

List of Acronyms

2.5D Two and Half-Dimensional

CAD Computer Aided Design

CDBN Convolutional Deep Belief Network

CHOMP Covariant Hamiltonian Optimization for Motion Planning

CNN Convolutional Neural Network

DoF Degree of Freedom

EPN Encoder-Predictor Network

FEM Finite Element Method

FK Forward Kinematics

FoV Field of View

FPFH Fast Point Feature Histogram

GAN Generative Adversarial Network

GMM Gaussian Mixture Model

GNGN Growing Neural Gas Network

GRU Gated Recurrent Unit

HSL Hue Saturation Lightness

HSV Hue Saturation Value

IK Inverse Kinematics

JT-RTT Jacobian Transpose-directed Rapidly Exploring Random Tree

LSTM Long-Short Term Memory

MAP Maximum a Posteriori Probability

MRF Markov Random Field

MSD Mass Spring Damper

MSE Mean Square Error

NURBS Non-Uniform Rational B-Spline

PRM Probabilistic Roadmap

RBM Restricted Boltzman Machine

RGB Red Green Blue

RGB-D Red Green Blue-Depth

RMSE Root Mean Square Error

RRG Rapidly-Exploring Random Graph

RRT Rapidly-Exploring Random Tree

SDF Signed Distance Function

STOMP Stochastic Trajectory Optimization for Motion Planning

SVD Singular Value Decomposition

Chapter 1

Introduction

1.1 Motivation

Research in robotic control and manipulation has achieved many advances across multiple fields. Some of the primary uses for robotic manipulation are in scenarios where it is unsafe or impossible for humans to operate, for example, space exploration, underwater manipulation, and bomb disposal. Additionally, robots can provide greater speed and higher precision for many tasks. Some examples include welding, assembling and surgery. Research problems in these areas have been solved with a high degree of success, but many problems require further investigation.

Research in robotics strives towards increasing the automation capacity of robots to work in flexible environments without human intervention or guidance. Therefore, it is essential to develop techniques for robots that resemble human skills. One of these skills that remains difficult and mostly unsolved for robots is the manipulation of deformable objects. Deformable objects also referred to as non-rigid objects, change shape and size as the result of manipulation. Humans manipulate deformable objects every day, whether it is tying shoelaces, opening a pencil case, shaping pizza dough, or folding laundry, but robots do not have flexible methods to accomplish all of these tasks. Determining intelligent and automated techniques for these problems will allow robots to automate a variety of these tasks and improve performance over humans when they require intense labour or speed and precision.

Specific robotic tasks that would benefit from flexible solutions to manipulating deformable objects include shaping parts for assembly in manufacturing, manipulating food for packaging, and robotic surgery. While robots play a role in these applications already, flexible and automated techniques still need to be developed.

In recent years there has been gradual progress in developing techniques to control and manipulate deformable objects. Many of these techniques are presented in [Chapter 2](#) of this thesis. Most of the research has been concerned with solving well-defined problems that involve handling cords and fabrics, leaving a need for task-independent solutions that can handle a variety of three-dimensional (3D) deformable and rigid objects. In many

cases, control and manipulation of these objects require precise models of the object being handled, limiting the ability to automate these tasks to novel objects. Other examples work on two-dimensional projections of three-dimensional objects rather than 3D objects. The research presented in this thesis fits with other ongoing research that explores sensing, modelling and control of 3D deformable objects. In particular, this work develops a solution for robotic manipulation of deformable objects and designs and implements a system to control the shape of a previously unseen three-dimensional deformable object using a robotic arm equipped with a three-fingered gripper.

Research in this specific area is confronted with numerous challenges that have yet to be fully solved. As mentioned previously, the issue of modelling objects presents a large challenge for performing manipulations and controlling the shape of an object. The challenge in modelling is twofold. Firstly, a method to measure and define a deformable object's configuration must be determined. This is especially challenging since deformable objects have infinite degrees of freedom and are therefore often approximated with a finite degree of freedom representation. In addition to the challenge in defining the object's configuration, there must also be an estimate of the deformation model, specifically how the object will deform to specific control actions.

Another challenge is getting accurate, precise, and complete 3D data from sensors. Sensors that capture 3D data are becoming cheaper and more widely available, but many of these sensors do not provide accurate and precise data. While this challenge is not unique to deformable object manipulation, it is a challenge to plan and assess deformations when the data is noisy and unreliable. In general, most manipulation tasks will deal with incomplete or partially observable objects. Even in rich sensor environments, when manipulating 3D objects, there will always be occlusions caused by the manipulator handling the object. This will leave gaps in the data that will need to be estimated for or result in only partial 3D models for tracking and assessing deformations.

Lastly, the problem of deformable object manipulation and shape control is severely underactuated. As deformable objects have infinite degrees of freedom, there is little to no guarantee that an object can be manipulated and controlled to the desired shape given a specific end-effector with strict kinematic constraints. This makes optimal shape control planning a challenge even when given an accurate representation and model of the object to be deformed.

1.2 Objectives

The main objective of the research presented in this thesis is to design a system to grasp and control the shape of a previously unseen three-dimensional deformable object using a robotic arm equipped with a three-fingered robotic hand. The system plans the grasp and manipulation to control an unseen object towards the provided desired shape. This work presents original contributions to deformable object manipulation to help develop flexible solutions to real-world problems in robotics.

The original contributions include an algorithm to distribute a robotic gripper's contact

points on the surface of a three-dimensional deformable object to optimally control it to a desired shape. This is done using an original representation of the object’s surface that reduces the dimensionality of the problem to a finite degree of freedom problem. Another contribution is the development of a collision-free path planning algorithm for the robot manipulator and hand to approach the object being grasped and place the end-effector contact points at the optimal collision-free grasp points from the candidates identified in the shape control algorithm. Path planning is done considering the kinematic constraints of the robotic system and any obstacles in the workspace that impact valid grasp positions for the arm and hand. Lastly, an original multi-view assessment phase is used to monitor the quality of the grasp and evaluate the object’s deformation. The quality of the deformation is assessed by comparing the 3D shape of the deformed object to the desired shape.

This thesis implements a full system to grasp and manipulate a deformable object that relies not only on the original contributions above but on some known solutions to standard problems in computer vision and robotics. The following list represents a step-by-step outline of the objectives achieved in this thesis, formulated as problems and solutions that make up the entire robotic grasping system.

- Identify and locate the object to be grasped using colour and depth images using known three-dimensional computer vision techniques to segment the object from the background.
- Estimate a three-dimensional model of the object using the segmented depth and colour images to enable more advanced grasp planning solutions by fitting primitive shapes to the segmented data.
- Identify the gripper contact points on the object to optimally control the three-dimensional deformable object to a desired shape using an original algorithm.
- Plan a collision-free path for the robot manipulator to approach the object being grasped and place the end-effector contact points at the calculated positions on the object.
- Execute the novel grasp and control strategy by controlling the robotic arm and hand to complete the planned manipulation of the object.
- Monitor the quality of the grasp and assess the deformation of the object by presenting multiple views of the deformed object to a 3D vision sensor.

This step-by-step process is implemented and experimentally validated on a variety of deformable objects.

1.3 Organization

The research presented in the remainder of this thesis is organized as follows:

- **Chapter 2** outlines the relevant literature associated with each part of the thesis. Specifically, Section 2.1 presents an overview of the state-of-the-art research for segmenting objects using traditional image processing techniques as well as more recent

techniques that incorporate colour and depth data. Section 2.2 presents relevant techniques used to generate three-dimensional objects from single view colour and depth data. Section 2.3 gives a description of the relevant robotic path planning algorithms in order to plan collision-free paths for the robot manipulator to guide the hand into the desired grasping position and orientation. Finally, Section 2.4 presents an overview of relevant planning and control algorithms for three-dimensional deformable objects as well as literature associated with tracking and assessing 3D deformations. This chapter presents a summary of a variety of research topics used by the solution designed in this thesis while highlighting some limitations of current technologies.

- **Chapter 3** starts by describing in detail a general framework for robotic manipulation of deformable objects and how this research fits into that framework. Section 3.2 describes the various pieces of computer vision and robotic equipment used to complete the research, as well as their associated technologies. Finally, Section 3.3 presents the objects that are used in the experimental validation.
- **Chapter 4** details the proposed robotic solution for shape control of 3D deformable objects. Section 4.1 discusses the methods used in the initialization phase of the system, including identifying objects and estimating their 3D geometry. Section 4.2 outlines the techniques used to plan control points on the objects to be deformed as well as manipulator path planning to bring the end-effector to valid grasping positions. Section 4.3 describes how the system executes the grasps and deformations developed in the planning phase and describes how the results are assessed.
- **Chapter 5** presents complete end-to-end system results on a diversity of cases for the techniques described in Chapter 4 alongside a comprehensive evaluation of the overall solution.
- **Chapter 6** reports a summary of the entire work as well as relevant contributions to the field and provides some proposals for future related work.

Chapter 2

Literature Review

This chapter provides an overview of relevant methodologies from the literature for all of the components that make up the robotic solution presented in this thesis. The components can be broadly grouped into four categories:

- Object Detection and Segmentation
- Object Completion
- Robot Manipulator Path Planning
- Robotic Manipulation of Deformable Objects

The object detection and segmentation section focuses on traditional and state-of-the-art methodologies for identifying an object in an image and removing it from the background. Object detection and segmentation is a preliminary step in the robotic grasping system for identifying the unseen object that is to be grasped by the robot.

The object completion literature review identifies relevant techniques to estimate an object's complete 3D geometry from incomplete data. These methodologies make it possible to plan complex 3D grasps on objects whose true 3D geometry is previously unknown.

Path planning algorithms allow a robot to move in a collision-free manner along a defined trajectory from an initial position to a desired end position. The literature review analyzes and discusses traditional and modern path planning algorithms. Manipulator motion planning is an essential step in moving the robot end-effector to a position where it can grasp and deform the desired object.

The final section of the literature review will discuss many current state-of-the-art techniques in 3D deformable object manipulation. These techniques include both grasp planning and control algorithms, as well as shape estimation and deformation tracking algorithms.

The scope of each of the topics and their potential applications are quite large, but they all play a crucial role in developing this research. In each section, specific attention is brought to techniques that have applications in 3D deformable object manipulation.

2.1 Object Detection and Segmentation

Image segmentation is the process of partitioning an image into multiple segments based on certain features associated with the pixels that make up each segment. This process is an essential task involved in many computer vision applications such as pose estimation, object detection, background differentiation and scene understanding. Traditional image segmentation approaches use Red Green Blue (RGB) colour or grayscale images and generally rely on thresholding or edge detection to determine boundaries between regions. While many techniques can solve this problem with a high degree of success, they are susceptible to failure when objects contain variable colours, textures or are exposed to uneven illumination. Recent technological advances have made depth sensors more accessible, allowing depth frames to be captured alongside colour images. The information provided by the depth data, specifically the geometry of the scene, can be used to improve traditional segmentation techniques. The following is a summary of image segmentation techniques used to detect one or more objects and segment them from the background.

2.1.1 Traditional Segmentation Techniques

One of the earliest methods of image segmentation is known as thresholding. The simplest thresholding method looks at a grayscale image and binarizes the image, replacing each pixel in the image, $I(i, j)$, with a black pixel if the pixel intensity is less than some threshold, T , and a white pixel if the pixel intensity is greater than that same threshold, $I(i, j) > T$. Thresholding can be done by manually choosing a threshold value or automatically by having an algorithm select the threshold based on some criteria. One of the first and most popular automatic thresholding techniques was presented by Otsu [1]. This method bins all pixel intensities into a histogram and iterates through all possible threshold values calculating the variance of the two groups to find a threshold that minimizes the sum of the foreground and background variances. This method can easily be extended to multi-channel (RGB) images. The algorithm can be implemented in a very computationally efficient manner and has been shown to work well when there is a significant background to foreground contrast but fails when there are uneven lighting conditions. To compensate for uneven lighting conditions, images can be transformed to different colour models (Hue Saturation Value (HSV), Hue Saturation Lightness (HSL), or YUV), which may significantly improve the results in certain cases. Various other thresholding techniques have been proposed and can be reviewed in the survey article [2].

Edge detection techniques [3] have been used as the base of several segmentation methods due to the close relation between edges and region boundaries. Boundaries between regions tend to have sharp changes in pixel intensity or colour, which can be easily identified using an edge detector (Sobel, Canny, Laplacian, etc.). The detected edges will often be disconnected and must be connected to form contours that identify closed regions. Connecting edges is done using a variety of techniques that may rely on interpolation, parallelism, symmetry and curvature. Many advanced segmentation techniques use edge detection methods as one step in a multi-step segmentation algorithm.

Region growing techniques assume that neighbouring pixels within a region must share some properties, such as intensity, colour or texture. The technique compares pixels with their neighbours and adds them to the region if they satisfy predefined similarity criteria. The technique continues in this manner until all pixels are grouped into regions. Seeded region growing [4] is an algorithm that relies on a set of input seeds that identify the regions to be segmented. These seeds are then used to iteratively grow regions by comparing unallocated pixels to the seeds and adding them to the region that satisfies the similarity criteria. The segmentation results are highly reliant on the choice of seeds. A variety of region growing techniques have been proposed based on this fundamental concept. Some rely on manual inputs for starting regions, while others rely on automated seed selection algorithms.

Several statistical methods have been proposed that build on the foundations outlined by the thresholding methods but rely on more complex probabilistic models. A Gaussian Mixture Model (GMM) was first used for image segmentation to develop an efficient adaptive algorithm to remove the background from images [5]. This technique uses a probabilistic model that consists of a number of Gaussians and identifies the best fit for each pixel to segment the image. Various improvements have been made using GMM to successfully segment images into a specified number of regions using grayscale or colour images. Another probabilistic-based model known as JSEG [6] combines colour quantization and spatial segmentation to generate possible boundaries and closed regions. This image is then used with a region growing technique to finish segmenting the image.

The idea of active contours, or snakes, is used to segment using a curve rather than working with individual pixels for segmentation [7]. Closed contours are defined by parameterized equations and are associated with a defined energy function. The algorithm segments the image by iteratively minimizing the energy function. In general, the energy function is composed of a smoothness term and an image gradient term. Given an initial curve, the contour iteratively moves towards segmenting the region until the energy term is minimized. Active contours work well to find the minimum contour to segment an object from the background, but the algorithm relies on a given initial contour and is susceptible to any edge artifacts. In addition, an active contour is restricted to segmenting a single simply connected region.

The level sets algorithm [8] uses the idea of active contours, but instead of parameterizing the curve, the whole image is discretized, and a function, $\phi(i, j)$, on the image is defined. The function is evolved iteratively and defines a set of contours where $\phi(i, j) = 0$ at the point that an energy function is minimized. Several implementations of the algorithm have been implemented that vary or extend the main idea. The energy function can be defined using either a region-based term or an edge-based term depending on the nature of the image being segmented. Shi and Karl introduce a real-time implementation of the algorithm for tracking contours referred to as fast level sets [9].

Some of the traditional segmentation techniques from the literature presented above have been implemented, and the results are shown in Figure 2.1. This section covers the main strategies in traditional image segmentation, but there exists a wide range of other advanced object segmentation algorithms. Object segmentation is one of the most

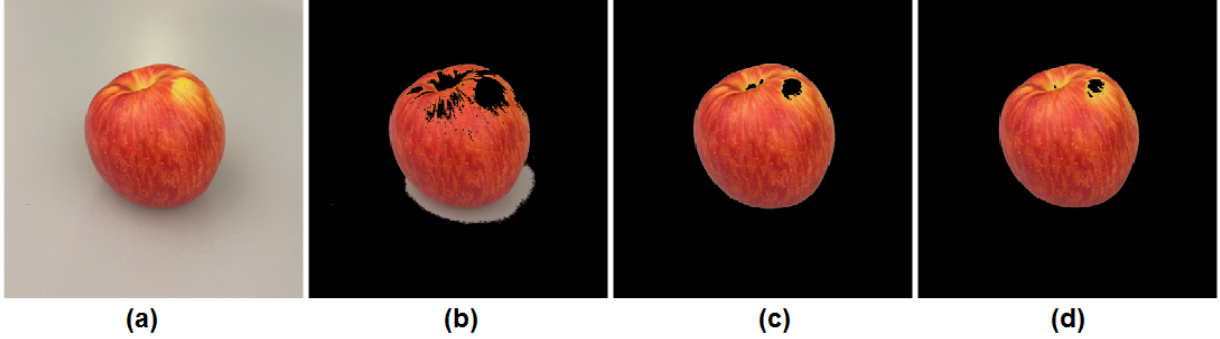


Figure 2.1: Examples of traditional segmentation techniques. (a) Colour image, (b) segmentation using grayscale thresholding, (c) segmentation using region growing, (d) segmentation using active contours.

widely studied problems in computer vision and image processing. It has therefore led to a variety of textbook chapters [10] and review articles that go beyond the scope of the articles reviewed in this section.

2.1.2 RGB-D Augmented Segmentation Techniques

A depth image is an image where each pixel’s intensity value represents the distance from an object’s surface to the camera. Depth images can be collected using various techniques, including stereoscopic vision, structured light, or active range-finding. Since the inception of the Kinect sensor and other cheap commercial RGB-Depth (RGB-D) sensors, there has been significant research into combining depth images with RGB images for image segmentation and object detection.

Early work proposed a segmentation technique for RGB-D video frames that combines visual information, depth information and prior knowledge of the scene to segment an object from the background [11]. This technique relies on the fact that the object to be segmented lies on a horizontal surface. A horizontal plane is fit to the depth data using the RANSAC algorithm [12] and then subsequently removed from the image. The next step is to perform vision-based background subtraction using a Gaussian mixture model to remove the background from the colour image. The two segmented images are combined, using the depth image as a starting point with pixels added from the RGB image if they are known to not be in the background or on the plane. Lastly, a filter is run on the segmentation mask to remove noise. This technique is effective at segmenting objects but relies heavily on prior knowledge of the scene. It is also problematic for small, dark or reflective objects due to the inherent noise in the depth images.

Another method [13] segments depth and colour images individually by first using a Sobel operator to detect edges in both images. Next morphological operators are used to enhance the detection of edges by combining erosion to reduce noise and dilation to fix gaps between edges. Small objects are removed, and then erosion is applied again to separate objects from each other. Finally, connected component labelling is performed,

and related components are merged. Once the process is completed for RGB and depth images, the results of segmentation are merged. In general, the depth image segmentation is used, but components segmented in the RGB image are included if they meet a criteria of being sufficiently large and compact.

Another way to leverage the depth data is to determine the type of boundary at an edge detected on a colour image by using the geometry deduced from the depth data. One method [14, 15] that leverages this idea begins by generating a binary edge map and edge direction using an edge detector on a colour image. Next, the geometry around the edges is used to define a probabilistic boundary edge map that defines whether the edge is likely a boundary between two objects. With the boundary probability defined, segmentation proceeds by assigning boundary ownership to the edge and then using a traditional technique by obtaining closed curves using the fixation-based segmentation method [16] to determine the segmented objects. Another technique [17] performs the same boundary probability estimate to segment RGB-D images while also implementing a novel trilateral filter and final segmentation using Graph Cuts [18]. Both of these methods work well but struggle to segment objects stacked on top of each other or objects placed side-by-side.

An alternative technique proposes a hierarchical approach [19, 20] that starts by grouping pixels based on surface normals and depth values. It continues by fitting planes and B-splines to the sets of clustered points. Furthermore, surfaces are selected using a support vector machine that has been trained on labelled data to determine the final surfaces to be used for segmentation. Lastly, the Graph Cut approach is used to find a global optimum for segmentation. This method was compared to the previous method [14] on two datasets and was shown to be superior. Specifically, this technique can segment objects successfully in very cluttered scenes, including objects stacked on top of each other and objects that are side by side. Although very successful, one drawback of this technique is the reliance on labelled data for training.

Other algorithms [21, 22] also use surface normals and surface edges to partition a cluttered scene into surface patches and object edges. Segmentation is performed on an object-level basis by identifying support planes and background pixels and excluding them from the image. Lastly, surface patches are segmented into objects using a 2D region growing method combined with a threshold based on 3D Euclidean distance. After initial segmentation, objects are segmented again using surface patches from the first step. The additional segmentation is done to handle objects that are stacked on top of each other or standing in front of each other.

It can be seen that a variety of techniques can segment objects successfully, although some perform better than others and are better suited for more complex scenes. While there have been some direct comparisons between algorithms reported above, there has not yet been a large-scale comparison of different techniques. Choosing the appropriate segmentation algorithm depends on the task at hand and on prior knowledge of the objects and scenes that are to be segmented.

In addition to the methods presented above, there has been a recent trend in automatically segmenting scenes using deep learning approaches [23, 24, 25]. These methods

use a combination of colour and depth images to train networks to perform automatic scene segmentation. These methods were not critically reviewed but are presented as an alternative to the more traditional methods presented in this section.

2.2 Object Completion

Object completion, also known as shape completion, is the problem of estimating the complete geometry of 3D objects or scenes from partial observations. In the real world, it is almost always the case that 3D data is incomplete, resulting in only partial geometric information. The data can be incomplete because it is captured from a single point of view, the view of the object is obstructed or due to noise from the sensor capturing the data. Depth images are considered Two and Half-Dimensional (2.5D) images. They contain 3D data but are only captured from a single point of view. These images contain valuable information, but many applications require full 3D models, including robotics, animation, modelling and architecture. In all of these applications, object completion must be performed to obtain a complete 3D geometry to work with. In this work complete 3D models are required to identify stable grasp locations and contact points for shape control on the surface of the 3D object.

Using a segmented object’s depth data, the 3D shape can be estimated using a variety of techniques that can be classified into one of three approaches: symmetry-based, object recognition-based, and learning-based. The techniques reviewed below will mostly focus on 3D object completion techniques for robotic grasping applications.

The earliest approaches to object completion use symmetry and extrusion techniques to identify symmetry axes and regular structures. The geometry is then copied from observed parts of the object to unknown regions. These techniques are successful given well-observed objects whose overall geometry complies with that of the visible region. Objects that are made up of simple geometric shapes can also be completed in this manner, either through symmetry or by identifying parts of primitive shapes and completing them.

Object recognition-based approaches complete shapes by matching partial inputs to models from a database. These can be augmented by using an assembly of known shapes or objects and combine them to complete a new object. These techniques are effective but require a large database of known shapes, requiring expensive computation and optimization. Additionally, requiring the use of a database makes them ineffective for novel objects and challenging to use for deformable objects whose shape may change as they are manipulated.

Learning-based approaches complete shapes by applying a model that will directly map a partial input to a completed output. These approaches generally employ a Convolutional Neural Network (CNN) that is trained on a set of partial objects alongside corresponding completed objects. Once trained, the model can be applied to any partial object and output an estimate of its completion. These methods tend to scale better to novel objects, provided that the data used for training preserves a high degree of similarity to the novel objects. After training the network, it is able to perform object completion very quickly

compared to symmetry-based or object recognition-based techniques. The ability to scale to previously unseen views and novel objects and the speed at which completion can be executed makes learning-based approaches a strong solution when deployed on data that is similar to the data the model was trained on.

2.2.1 Symmetry and Primitive Object Methods

Symmetry and primitive object methods are good candidates for robotic grasping solutions as many grasp planners rely on primitive shapes to identify stable grasps. These techniques can further be grouped into a variety of categories. The first group can be defined as those that use stochastic-based techniques to identify primitive objects such as boxes, spheres, cylinders and cones. The RANSAC algorithm [12] has been used extensively in primitive object detection and completion methods to fit observed data to completed shapes. An iterative primitive detection method developed by Schnabel *et al.* [26] uses RANSAC to detect planes, spheres, cylinders and various other primitive 3D shapes from point clouds. The algorithm fits primitive shapes to the data and selects a completion based on the minimal error. Inlier points are removed from the point cloud, and the process continues iteratively until a stopping criteria is met. A variety of algorithms have been built on this framework with added constraints and optimizations to improve performance.

Parameter-based techniques rely primarily on quantizing the image pixels into bins and identifying samples that correlate to geometric elements. The Hough transform [27] is a feature extraction technique that relies on this parameterization to identify different geometric elements from noisy instances using a sample voting procedure. Examples of Hough-based shape detectors include [28] which identifies planar regions represented by sets of 3D points that are refined using singular value decomposition. These planes are then used to extract polygons with user assistance to specify boundaries.

Clustering methods generally involve region growing by assigning seeds to a data set and grouping neighbours with similar properties, similar to region growing techniques for image segmentation. Once the data is segmented into patches or other clustering structures, algorithms can identify primitive shapes. These algorithms may use Euclidean distance and normal orientation for plane fitting or simply use the mean squared error to identify the best fit for a set of primitive shapes.

Robotic grasping specific examples include the work done by Bohg *et al.* [29], which estimates the occluded and unknown parts of a sensed object by leveraging symmetries found in partial point clouds. The algorithm associates pairs of detected surface points that are matched by reflection on the opposite side of a symmetry plane. Assuming objects that are to be grasped possess symmetries perpendicular to the planar surface they are positioned on allows complete 3D shapes to be estimated.

Quispe *et al.* [30] exploit symmetries in objects to be grasped to create complete 3D objects from a single RGB-D image. First, an optimal symmetry plane is identified using the technique presented by Bohg *et al.* [29], and then the object is completed using either rotational or planar extrusions. The optimal extrusion is identified in a multi-step process that aims to minimize the dispersion of the points along the object’s surface.

2.2.2 Object Recognition Methods

Object recognition-based methods can be highly effective for obtaining high accuracy complete 3D models from 2D or 2.5D data. The main drawbacks are that they require a database of known objects and can be computationally expensive. Many object recognition methods rely in part on elements that may be classified as learning algorithms. Whether they rely more heavily on traditional recognition or learning methods, they may be summarized here or in Section 2.2.3.

The work by Papazov and Burschka [31] uses a RANSAC-like sampling strategy to compare pairs of oriented sample points to pairs of oriented model points. Object hypotheses are developed and assessed based on the similarity of the sampled and modelled points, and a model hypothesis and quality measurement are provided. The algorithm is robust to object orientation, partial occlusions and sensor noise.

Hinterstoisser *et al.* [32] present a method for detecting 3D objects from aligned colour images and dense depth maps. The algorithm first calculates the normalized gradient map from the colour image and the quantized surface normals from the depth map. These measures are then used in a similarity measurement to compare sensed data against object templates to identify the estimated 3D model.

Another method that uses a database of models is presented by Li *et al.* [33], where keypoint descriptors from known models are generated using a 3D corner detector and an unsigned distance metric. 3D data is recorded and converted to a Signed Distance Function (SDF). Keypoints are then computed directly from the SDF representation and correspondences with keypoints in the database are identified. A single-point RANSAC sampling technique is used to match database models with captured data to register the objects. Lastly, the maximal agreement between registered correspondences is searched for to generate the best fit 3D model.

Alexandre [34] performed a comparative evaluation of 3D descriptors to determine which 3D descriptor can be best used for object recognition tasks. A variety of descriptors were tested on a database of point clouds for both object and category recognition performance. It was determined that successful object recognition is correlated with the number of 3D descriptors used with the penalty of additional computation time. For individual 3D descriptors, the success varied significantly for the desired task.

2.2.3 Learning-Based Methods

Learning-based approaches as a solution to object completion almost always involve the use of 3D volumetric CNNs. CNNs have been used successfully in numerous image processing applications such as object recognition, object segmentation and edge detection. One of the most important features of the CNN is that the convolutional layers dramatically reduce the number of parameters to be trained, making them computationally more viable than fully connected neural networks. This is especially important when working with 3D data where the number of parameters for the network to learn can quickly become

computationally infeasible. The following publications all use 3D CNNs, either on their own or as part of a more complex neural network, to perform object completion.

The earliest work that applied deep learning to 3D shape representation, including reconstruction, was 3D ShapeNets [35]. It builds a framework to learn the 3D representation of shapes across many different object categories and multiple poses from a large database of Computer Aided Design (CAD) models. To predict the output geometry of the 3D object, a Convolutional Deep Belief Network (CDBN) was designed. A CDBN is a type of convolutional neural network composed of multiple convolutional Restricted Boltzmann Machines (RBM). The CDBN works by learning the joint probabilistic distribution over a voxel grid and class labels for ten 3D volumes. A voxel is simply a representation of a value on a regular three-dimensional space; it is the 3D version of a pixel. The model is then applied to the task of shape completion. The success of this model has led to more recent work that improves the results.

A method [36] that builds on the results produced by 3D ShapeNets combines data from multiple RGB images from different viewpoints. A 3D shape is modelled by a Markov Random Field (MRF) where the information is retrieved from a set of RGB images captured at multiple viewpoints. The MRF model structure is learned from 3D shape models using a CDBN, thus formulating the task as a Maximum a Posteriori Probability (MAP) inference problem. This method improves upon the results of 3D ShapeNets. These models are able to reconstruct objects with a high degree of success but are limited by the fact that they rely on objects belonging to certain labelled classes. More recent research attempts to solve this problem by performing object completion without any class or object labels.

VConv-DAE [37] proposes a method of shape interpolation without labels that can be used for classification, denoising and shape completion. This work proposes a fully convolutional volumetric auto-encoder that learns to predict shape geometry from a set of voxels. The training data corresponds to the voxel occupancy of a completed object. The CNN architecture consists of an initial dropout layer in order to augment the input data and avoid over-fitting. This is followed by two convolutional layers to extract features of various levels of detail. The feature maps are flattened and input into a fully connected layer. Finally, a deconvolution layer is used to output a volume with the same size as the input volume. When the network is trained and tested for object completion, it performs significantly better than 3D ShapeNets.

Varley *et al.* propose a method [38] that will also complete shapes without any labels using a partial voxel grid and a completed grid to train the model. The method implements a novel 3D CNN that takes a single point of view RGB-D image in the form of a point cloud and converts it to a voxel grid for training and testing. This work is unique because it can be applied to solve tasks where only a single point of view is available, making it a good candidate for use in robotic grasping systems. In order to train the network, the YCB [39] and Grasp Database [40] datasets are used, rotating the objects to create additional single point of view training data. This method performs well on training views and previously unseen views of objects in the dataset. It scales reasonably to completely novel objects, making it a suitable choice for tasks where previously unseen objects may be encountered. The authors conclude that this method outperforms many general geometry-

based approaches.

Choy *et al.* propose a recurrent network [41] that learns mappings from 2D object images to their 3D representations. The network does not require any labels, but it does require bounding boxes to identify the objects in the 2D images. The network works for single-view or multi-view images, with performance increasing with additional views. The network architecture consists of a deep 2D CNN, a 3D convolutional Long-Short Term Memory (LSTM) unit and a 3D deconvolutional CNN to reconstruct the 3D objects. An LSTM unit is a component of a recurrent neural network that controls the flow from input to output by using a memory unit and control gates, allowing a deep neural network to avoid the problem of the gradient getting too small to continue to learn (vanishing gradient problem). Specifically, an LSTM unit known as a Gated Recurrent Unit (GRU) is used, which solves the same problem while using less computation. The network was trained on a set of 127×127 input images, along with a completed voxelized reconstruction of size $32 \times 32 \times 32$. This network had success in testing on other datasets, including ShapeNets [35] and a set of real-world images. The results are mixed, with both successful and unsuccessful reconstructions on the datasets.

Dai *et al.* [42] and Han *et al.* [43] both propose a 3D Encoder-Predictor Network (EPN) that uses convolutional layers to determine completed shapes from partial input. The works differ in that Dai *et al.* use the 3D EPN to predict a coarse shape that is further refined using a 3D object database to determine the shape’s final geometry. Han *et al.* predict a coarse shape using a long short-term memorized context fusion module along with 3D fully convolutional module, which is then further refined using the 3D EPN. While both methods are used successfully, they differ in that Dai *et al.* require a database of models for the local refinement, making it an inferior choice if novel objects are encountered.

Yan *et al.* formulate the learning process for 3D object completion as an interaction between 2D and 3D representations [44]. The proposed encoder-decoder network takes an input image and predicts a 3D shape so that the perspective transformations of the predicted object match well with corresponding 2D images. This is uniquely done using a projection loss function, which allows the network to predict 3D shapes from 2D data without explicitly using 3D volumes as the ground truth when training the model. The model demonstrates the ability to generate 3D volumes from a single 2D image in three separate conditions: learning from a single class of objects, learning from multiple classes of objects, and testing on previously unseen object types. The model showed positive results and good generalization when the projection loss was implemented.

More learning-based methods exist that use more complicated neural network architectures. Some of these methods include [45, 46] that use a Generative Adversarial Network (GAN). A GAN is a model that is implemented by a system of two neural networks, a CNN to generate a completed shape and another neural network to evaluate whether the completed shape is an instance from the true data distribution or a candidate produced by the generator. These two networks learn together to produce completed shapes that cannot be distinguished from ground truth completions. Object completion using these networks has been shown to outperform previous methods. Specific comparisons show that the model proposed by Yang *et al.* [46] is able to outperform Varley *et al.* [38] in single-view shape

completion when the networks are trained and tested on a single category of objects.

Learning-based alternatives to 3D CNNs involve using a neural network directly on raw point clouds without any structural assumptions. Recent work proposes a method that uses another type of deep neural network that operates directly on point clouds, known as a Point Completion Network [47]. It combines pointwise multi-layer perceptrons with a symmetric aggregation function that is invariant to data permutation. This is necessary as point clouds are unordered sets. Because there is no local geometry in point cloud sets, it is not possible to use any convolution functions. The proposed decoder combines multiple techniques, using a fully connected layer to generate a coarse output and a folding-based decoder to generate a more detailed output based on the coarse output. These types of networks are attractive as they have the ability to produce results with much higher resolution while also reducing memory requirements and could potentially be used without any pre- or post-processing.

2.3 Robot Manipulator Motion Planning

Robot motion planning is solving the problem of moving a robot from a start pose through a sequence of valid configurations and arriving at a desired end pose while avoiding collisions with obstacles and conforming to the robot’s dynamic and kinematic constraints. Motion planning is one of the fundamental problems in robotics; consequently, there has been significant research into the problem. The main ideas of path planning and motion planning can be applied to mobile robotics and robotic arm manipulators. While many of the same principles apply to both cases, they also raise their own set of challenges.

This work focuses on motion planning for high Degree of Freedom (DoF) robot manipulators used in grasping applications. Redundant manipulators are generally used for complex grasping applications to ensure the manipulator has the required DoF to achieve the position and orientation required to complete the task.

In general, the robot and its workspace are described in three-dimensional space, while motion is defined in the robot’s configuration space. The configuration space, $\mathcal{C}$, of a robot contains all possible positions and orientations that the robot can attain. While $\mathcal{C}_{\text{free}}$ is the set of all configurations where the robot is not in collisions with an obstacle. The configuration space is defined using a set of generalized coordinates that uniquely define the system’s configuration relative to a reference or base configuration. The idea of using the configuration space of a robot for motion planning was first introduced by Udupa [48] and Lozano-Perez [49]. These early approaches were the first to attempt to exploit the properties of a robot’s configuration space to optimally plan motion while avoiding obstacles.

Since then, robot motion planning has garnered significant interest and generated a great deal of research. The research can be broadly broken up into three types of algorithms: search-based, sampling-based and optimization-based. While these approaches have major differences, they are not necessarily mutually exclusive, and many state-of-the-art motion planners use a combination of the above.

2.3.1 Search-Based Methods

Search-based motion planning, often referred to as combinatorial planning or graph-based planning, solves the problem of motion planning by fully mapping the robot's workspace and searching for collision-free paths from an initial position to a goal position. Once the space has been subdivided, search algorithms such as Dijkstra's [50] and A* [51] can be used to find an optimal, collision-free solution on the defined space. Variants of these search algorithms have also been developed to work on dynamic graphs [52].

Other methods attempt to capture the connectivity of the robot's configuration space and solve the motion planning problem by building road maps through the free space. This consists of placing nodes in free space, making collision-free connections between nodes and finding the shortest path to the goal. Visibility graphs [53] map nodes to the vertices of configuration space obstacles and draw straight lines between them to create a collision-free graph where an optimal path can be planned. Trapezoidal decomposition [54] uses obstacle vertices to map the configuration space into a set of trapezoidal cells where collision-free paths exist between adjacent cells. A variant adds additional processing to check the links between all cells and connects them if the paths are collision-free, reducing the length of the path. Examples of road map path planning algorithms for a simple 2D scenario are shown in Figure 2.2.

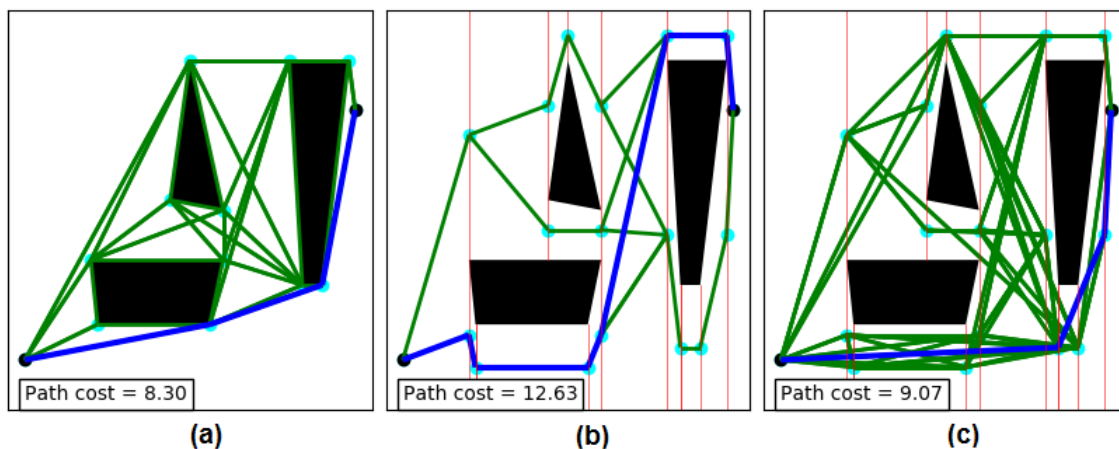


Figure 2.2: Example of simple search-based motion planning algorithms in 2D with obstacles in the workspace. For each algorithm the start and end nodes (black dots), potential path nodes (light blue dots), edges between nodes (green), obstacles (black shapes) and final collision-free path (blue) are illustrated. The algorithms are (a) visibility graph, (b) trapezoidal decomposition and (c) trapezoidal decomposition with all cells linked.

While these methods find optimal paths and are complete, meaning they will always return a solution if one exists and report if one does not exist, they are computationally very expensive. As the dimension of the configuration space increases, these algorithms' performance degrades, and they can become computationally infeasible, making them a poor choice for high DoF manipulators.

A search-based algorithm that effectively handles high dimension configuration spaces was achieved through the use of basic motion primitives and an anytime heuristic search [55]. Rather than building a complete graph of robot states, a lattice is constructed using pre-computed motion primitives of the robot. This reduces the number of nodes while ensuring the solution will be a feasible path. A cost function is derived that weighs minimizing the path length, maximizing smoothness and maximizing the distance from obstacles. The heuristic function guides the search in promising directions using a weighted cost of desired end-effector position and orientation. Lastly, the search is performed using the ARA* algorithm [56], where an initial possibly suboptimal solution is found quickly, and then the algorithm works to improve the solution for a limited amount of time. While this search-based motion planner is still less efficient than some sampling-based methods, it guarantees completeness and bounds the suboptimality of the solution.

2.3.2 Sampling-Based Methods

Sampling-based planning algorithms are concerned with searching the configuration space, $\mathcal{C}$, for a solution rather than completely characterizing the entire configuration space and finding an optimal path within it. While sampling-based algorithms are not complete, many are probabilistically complete, meaning that a solution will be found or no solution will be reported in a finite amount of time. One main aspect of sampling-based techniques is the introduction of a collision detection function. Since the entire configuration space is not mapped, each new configuration must be checked for collisions. This includes checking for robot self collisions and collisions with the environment before the robot configuration is included in a path for motion planning. The most popular types of sampling-based planning methods are mostly based on the Rapidly-Exploring Random Tree (RRT) algorithm and Probabilistic Roadmap (PRM) planner. Both of these techniques rely on the use of a graph structure, using nodes to represent robot configurations and edges between the nodes representing the motion from one configuration to another.

The probabilistic road map algorithm [57] is most commonly used when searching for multiple goals (multi-query solutions). The algorithm starts with a pre-processing phase with an empty graph and iteratively samples points from the configuration space, $q_{\text{rand}} \in \mathcal{C}$. These points are added to the graph as vertices, and connections are attempted between the new vertex and other vertices in the graph within a certain radius, r , of q_{rand} . If a connection does not cause a collision, a new edge is added to the graph connecting the vertices. A query phase then searches for paths that connect initial and final conditions.

Numerous implementations of the PRM algorithm have been designed, with the main difference being the choices for the set of vertices where edge connections are attempted to the new vertex. k -Nearest PRM [58] chooses the nearest k neighbours to the new vertex constructing a collision-free network as a random k -nearest graph. Bounded-degree PRM [58] uses a fixed connection radius, r , while attempting a number of connections proportional to the current number of vertices in the graph. When a large number of vertices are present, this can become computationally very expensive, so an upper bound k is also introduced. PRM* [59] differs from previous iterations of the algorithm by introducing a variable-length connection radius, r , proportional to the number of vertices in the graph.

The rapidly-exploring random tree (RRT) algorithms [60] are generally designed for single-query problems. The algorithm starts with an empty graph that includes the initial configuration of the robot as a vertex with no edges. The algorithm iteratively samples points from the configuration space, $q_{\text{rand}} \in \mathcal{C}$. The algorithm searches for the nearest vertex and attempts to connect the new point. If the connection is successful, q_{rand} is added to the graph, and an edge is established between the two vertices. RRT-Connect [61] is a variant of the algorithm that grows two trees, one from the initial configuration and one from the goal configuration to increase the speed of finding a collision-free path.

The Rapidly-Exploring Random Graph (RRG) algorithm [59] is somewhat of a combination between a PRM and an RRT. Every time a new point, q_{new} , is added to the graph, additional edges within a variable radius, r , are also tested for successful connections. This algorithm produces a vertex set that is equal to that of an RRT, but the RRG has an edge set that is a subset of that of the RRT. This leads to the possibility of finding shorter paths within the graph. RRT* [59] improves the efficiency of RRG by removing redundant edges that are not part of the shortest path. This ensures that all vertices are reached through a minimum distance path. RRT* adds points to the graph in the same way as RRT and RRG but only adds edges to vertices along a path with minimum distance, and edges that have higher cost are deleted.

Figure 2.3 shows examples of RRT, RRG and RRT* after 100, 250 and 500 iterations. It is clear to see that RRG and RRT* have clear advantages over RRT in terms of the cost of the path to the goal as the number of iterations increases. This is due to the remapping of edges as more iterations are performed to search for lower-cost solutions. Additionally, while RRG and RRT* have the same minimum path to the goal, RRT* contains significantly fewer edges, making it computationally more efficient for single-query problems. A more thorough comparison of a variety of sampling-based path planning algorithms can be found in [59].

Redundant manipulators are those that possess more degrees of freedom than required for a specific task. These manipulators pose an additional challenge for motion planning as there are numerous, if not infinite, configurations for each end-effector pose. Therefore, goal planning to determine the goal configuration adds an additional step before motion planning can proceed when using path planning approaches like those presented above. This extra step can be avoided by combining motion planning and goal selection for redundant manipulators by planning motion based on a goal position and orientation rather than a predefined endpoint configuration.

One solution [62] that combines motion planning and goal selection for redundant manipulators does so by planning motion based on a goal position and orientation constraints. This can be accomplished using an RRT combined with a heuristic goal function to bias the manipulator to move towards a goal a certain percentage of the time. Directing motion towards the goal is accomplished by keeping the graph vertices sorted with respect to distance to the goal using a defined goal function. Vertices are added to the graph if they move the current closest vertex closer to the goal. To avoid issues of local minima, cases where extending towards the goal results in collisions or are otherwise unfeasible are tracked, and the vertices are removed from the ranked list if they reach a certain thresh-

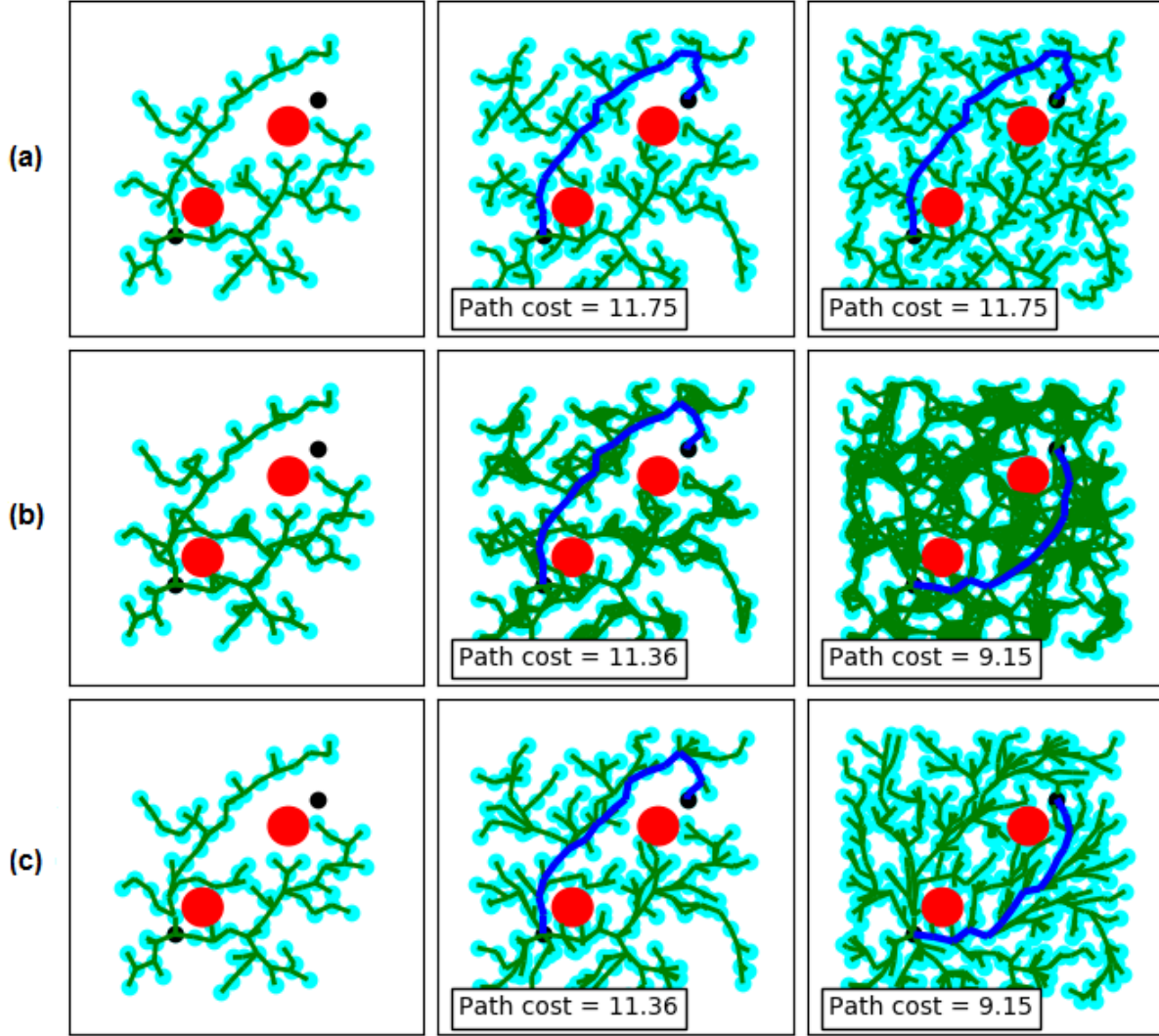


Figure 2.3: Example of sampling-based motion planning algorithms with nodes, edges and final path if goal is reached. For each algorithm the start and end nodes (black dots), potential path nodes (light blue dots), edges between nodes (green), obstacles (red shapes) and final collision-free path (blue) are illustrated. The algorithms are (a) RRT, (b) RRG and (c) RRT*. Number of iterations for each algorithm from left to right: 100, 250, 500.

old of failures. Additionally, this removes the need for a direct inverse kinematic solution because the solution is integrated directly into the planning process. While this algorithm is effective, the goal heuristic function can be difficult to tune when including orientation constraints for the manipulator.

An alternative method [63] builds upon this idea, but rather than extending arbitrarily from the node closest to the goal, the Jacobian of the manipulator is used to extend in the direction of the goal pose. This maintains the condition of not requiring explicit inverse kinematics as the Jacobian is calculated numerically at each configuration. Additionally, convergence time is improved by moving directly towards the goal pose a certain percentage

of the time. The idea of an RRT approach using a Jacobian Transpose-directed Rapidly Exploring Random Tree (JT-RTT) was extended [64] to include the idea of workspace goal regions, which allows the specifications of a continuous goal region of end-effector poses.

2.3.3 Optimization-Based Methods

Optimization-based path planners exist as both standalone motion planners and as post-processing planners to optimize redundant or jerky motions from paths generated from sampling-based methods. The concept of artificial potential fields [65] has been used in robot path planning to construct a smooth, attractive potential function that covers the entire configuration space. The potential function has lower values closer to the goal and higher values further away from it. Additionally, a repulsive potential function can be constructed that relates to the distance to the closest obstacle. The robot position can then be controlled by choosing the direction along the gradient of the potential field. This concept can be extended to real robots by choosing a set of control points on the robot and applying the potential field at each point.

Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [66] is a gradient descent technique that was initially used for motion planning of a 7-DoF manipulator. The algorithm minimizes a cost function that combines path smoothness and distance from obstacles to generate an optimal motion plan. Gradients are required for both elements of the cost function to generate the motion plan. A signed distance field is used to represent the environment to generate obstacle costs. The algorithm uses a Monte Carlo simulation to avoid converging to high-cost local minima and for probabilistic completeness.

Stochastic Trajectory Optimization for Motion Planning (STOMP) [67] uses a similar technique but does not require gradients for the cost functions to generate a motion plan. STOMP starts with a simple, not necessarily collision-free, path from an initial position to a goal position and iteratively modifies the path using the cost function of a set of noisy trajectories until an optimal collision-free path is found. By keeping the cost function general, the path can be optimized for a variety of constraints. Additionally, it has been shown that STOMP can overcome some local minima that other gradient-based methods get stuck in.

2.4 Robotic Manipulation of Deformable Objects

As research in robotics moves towards developing flexible, autonomous solutions that can be implemented without human intervention or guidance, there is an increased need for techniques that resemble human skills. One of these skills that remains difficult and mostly unsolved is the manipulation of deformable objects. Deformable objects are those that can change form as the result of manipulation. In recent years there has been gradual progress in techniques to control and manipulate deformable objects. The majority of this research is focused on handling 1D (linear) and 2D (planar) objects to complete predefined tasks. There has been some progress in manipulating and controlling 3D objects as well. A

variety of recent surveys document the current research involving robotic manipulation and modelling of deformable objects [68, 69, 70]. These surveys are large in scope and explore sensing, modelling, and manipulation of a variety of classes of deformable objects. The survey articles can be reviewed for a complete breakdown of state-of-the-art techniques up until the time they were published.

This section will specifically present relevant state-of-the-art research related to grasping and deformation control of 3D deformable objects, as well as 2D projections of 3D objects. This is followed by a review of literature related to 3D shape estimation and deformation tracking.

2.4.1 Grasping and Deformation Control of 3D Objects

One of the earliest works concerning the deformation of three-dimensional objects was proposed by Tokumoto and Hirai [71], in which they developed a forming machine for use in food processing. Feedback from a 2D image system was used to predict deformations of rheological materials, specifically dough, and simultaneously control the shape to a target.

Other early work by Shibata and Hirai [72] models deformable objects as a Mass Spring Damper (MSD) system and simulates simultaneous control of motion and deformation. Deformations in one dimension were modelled, and the analysis of dynamical behaviour was performed while taking into account the stability of the system.

The work by Higashimori *et al.* [73] proposes a method to actively shape rheological objects based on a determined deformation model. The objects' dynamic characteristics are defined using a model composed of the objects' elastic and plastic responses to a specified input force. The work proposes a two-stage implementation for shaping the objects. First, it determines the elastic response while avoiding over-deformation of the object. The next stage uses the calculated elastic deformation parameters to generate the desired deformation while actively managing the plastic deformation response. This work was shown to successfully deform 3D objects in a single dimension. Yoshimoto *et al.* [74] extend the previous work to two dimensions by actively modelling and deforming objects along two perpendicular axes. This method is still unable to handle general 3D shapes and has not been extended to a full three-dimensional model.

Smolen and Patriciu [75] studied surgical applications of deformable object modelling and deformation planning through simulation. The work used the reproducing kernel particle method to simulate the mechanics of a deformable object to manipulate complex objects at their boundaries with the goal of controlling internal points to specified positions. The path planning algorithm provided accurate results for tracking control point positions on the object in simulation.

Das and Sarkar have proposed a variety of techniques for manipulating 2D projections of 3D deformable objects. In [76, 77] an optimization-based planner that minimizes the error between an initial contour and final contour is used to determine the locations of manipulator contact points to control the shape. This technique was used on objects modelled as an MSD system and achieved good results through simulation. They also

designed a robust controller for the shape control task that can work even when there is uncertainty in the object model. In [78] target points inside of objects are manipulated by control of actuation points at the object’s boundary. The problem is formulated to minimize the force required to move the target points to a desired position. This is again achieved through modelling the object as an MSD and simulation.

The works by Navarro-Alarcon *et al.* [79, 80] developed a variety of control laws for manipulating control points on deformable objects. Adaptive control techniques are used in conjunction with 2D visual servoing to iteratively estimate a deformation model of the object by tracking feature points as the object is manipulated. This method allows deformation control without prior knowledge of the object’s deformable properties. Throughout the research, control points on the boundary of the object are manipulated to deform the object to meet a variety of criteria, including feature point position, distances between feature points and angles between feature points. The work was extended to 3D objects in [81] by using a 3D computer vision system for visual feedback.

In more recent work [82], instead of controlling a series of feature points, a 2D contour of the object is controlled using truncated Fourier series coefficients and 2D visual feedback. This technique gives greater control over the total deformation of the object. Again control is done in a model-free manner by iteratively estimating the deformation model throughout the control process.

Recent work by Delgado *et al.* focuses on the control and manipulation of objects relying on tactile feedback from a robotic hand rather than visual feedback. In [83] an object’s softness is estimated in an initial feature estimation phase, which is then used with an adaptable tactile servo control scheme for in-hand manipulation tasks, such as lifting, rotating or moving the objects. This control scheme is based on a minimal spring model for the objects. It is used to maintain forces at the fingers’ contact points during manipulation to maintain a stable grasp on a variety of deformable objects. This approach was augmented in [84] to include a wider range of deformable and rigid objects. In [85] a model-free approach was used for different manipulation tasks on deformable objects. Specifically, this work relies on tactile images obtained using a combination of dynamic Gaussians and uses tactile servo control to stabilize grasp points to avoid slipping as the object deforms.

Nadon and Payeur [86, 87] focus on grasp point selection for shape control of deformable objects with a three-finger robot hand. Given an initial and final 2D contour of an object, grasp points are selected to minimize the difference between the contours. This is done in a model-free manner using the concept of diminishing rigidity to place the contact points at locations that represent local maxima on the difference contour and that also satisfy the criteria for a stable grasp. Experiments were performed using a variety of deformable objects and desired contours with a three-fingered hand whose wrist was fixed to the top of a table.

Zaidi *et al.* [88, 89, 90] develop a grasping and manipulation strategy for robotic hands based on the physical constraints between the fingers and the surface of a deformable object. They split the grasping strategy into two parts. The first phase finds initial contact points on an object using grasping triangles to identify likely stable grasp points. In the second

phase, contact interactions between the fingers and the object are simulated using a mass-spring model to find stable grasps that satisfy the force equilibrium to ensure a stable grasp is obtained. The research simulates contact interaction and applies the results to a real implementation. The work succeeds in planning stable grasps on deformable objects for pick and place operations.

Hu *et al.* [91] tackle the problem of deformable object shape control by using a deep neural network-based controller to control the position and deformation of an object with unknown deformation properties. The network maps the end-effector’s movement and the object’s deformation measurement using an adaptive learning technique. Deformation measurement is done using keypoint features defined on the object’s surface using a combination of Fast Point Feature Histogram (FPFH) and principal component analysis to reduce the dimensionality. Simultaneous tracking and object estimation for the deformable object is used to compensate for the partial observability during the deformable object manipulation.

2.4.2 Shape Estimation and Deformation Tracking

In order to determine whether the manipulation of a deformable object is successful, an estimate of the deformed object must be calculated. Early work on estimating deformation was done by Khalil *et al.* [92] in which surface shape estimation is performed using stereoscopic vision and disparity maps. The research focuses on the real-time mapping of a single surface of a deformable object as it is deformed by a fixed robotic gripper.

Cretu *et al.* [93] track a segmented deformable object that is being manipulated by a robotic hand using a Growing Neural Gas Network (GNGN) to define a minimal enclosing contour every time a video frame is captured. The GNGN works by predicting and adjusting the network’s neuron position to optimally define the contour of the object as it deforms.

Valencia *et al.* [94, 95] also use a modified GNGN to represent a deformable object in a graph structure. The network is used to track an object as it is deformed by a robotic hand in real-time. Additionally, a neural network was trained to predict deformations of the graph structure based on the current graph and the displacement of the fingers of a three-fingered robotic hand.

Nadon and Payeur [86] and Hui *et al.* [96] both use an implementation of the fast level sets algorithm to segment and track the contour of a 3D object as it is manipulated by a fixed robotic hand. The contour is tracked using colour images collected by a sensor positioned directly above the hand. The object is segmented from the image, defining a 2D contour that encloses the 3D object. The resulting contours are used for shape control and deformation parameter estimation, respectively.

Cuiral-Zueco and López-Nicolás [97] present a novel RGB-D deformable object tracking algorithm which is also used for optimal next best view planning. The visible area of a previously unseen object is discretized using a supervoxel graph each time a frame is captured. The deformation estimate is done by defining metrics and an acceptance criteria

to match supervoxels from one frame to the next. The algorithm is robust against partial object occlusions as well as changes in lighting that occur between subsequent frames. The use of supervoxels makes the algorithm computationally inexpensive, allowing it to be run in real-time.

Real-time shape estimation and deformation tracking are also required to automate and assist surgeons during robotic surgery. Malti *et al.* [98] estimate the shape of soft tissue during surgery by using a shape-from-motion algorithm to estimate an undeformed 3D model template and then assess the deformations caused during surgery by computing the correspondence between real-time images and the template by estimating shearing and scaling parameters. An alternative approach is presented by dos Santos *et al.* [99] that uses surface matching instead of parameter estimation to assess the deformations.

Han *et al.* [100] describe a vision-based algorithm for surgical path definition that uses visual feedback of feature points to estimate unknown deformation parameters. An ongoing update of the feature points after deformation is used to minimize further deformation.

Outside of robotics, deformation estimation and assessment is of interest to the graphics and augmented reality communities as well. A few of the papers from that community are summarized below, although this is not a complete review of deformable surface estimation in computer graphics.

De Aguiar *et al.* [101] use feature point detection and tracking to find correspondences between subsequent images in a colour video. The tracked points are combined with several constraints on the mesh points to estimate and reconstruct deformations for 3D movies, games or augmented reality applications.

Jordt and Koch [102] track deformable objects from RGB-D data using a Non-Uniform Rational B-Spline (NURBS) function. NURBS functions are used readily in computer graphics as they can define various types of continuous surfaces. The NURBS function is used to find a first estimate of the initial surface of the shape and is then registered to a 3D mesh deduced from the RGB-D data. The mesh is then deformed by manipulating control points of the NURBS function, and the tracking is done by finding the set of NURBS control points that minimize the error between the function and the reconstructed mesh. This method allows deformations to be tracked without explicitly tracking feature points as well as using a low dimensional space to describe object deformations.

Ngo *et al.* [103] recover the deformed shape of surfaces by computing a 2D warp between a pair of images and infer the 3D shape from it. This technique uses the concept of Laplacian formalism, which is to represent a surface as the Laplacian of the object’s 3D mesh. With this reduction of dimensionality as well as imposing other linear constraints, the 3D reconstruction problem can be formulated as solving a non-degenerate linear system. Furthermore, it is shown that the least-squares problem can be further reduced by expressing combinations of vertices as a smaller number of control points.

2.5 Summary and Further Development of Methodologies

This chapter provides a review of the state-of-the-art literature associated with the many components of the robotic solution presented in this research. Introducing and discussing the relevant literature helps position the original contributions developed in this work.

Many of the works in the literature review are expanded on if they are used in the robotic system or if they inspire the original work developed in the thesis. Chapter 4 will present the robotic grasping solution developed for this thesis and expand on some of the relevant work first presented in Chapter 2.

Section 4.1.2 uses boundary probability detection [14, 15, 17] and the fast level sets algorithm [8, 9] to develop a novel technique relying on depth and colour image information to segment a previously unseen object from the background.

Section 4.1.3 implements an object completion method that uses a primitive shape detection algorithm to generate estimates of 3D objects from a single point of view RGB-D image. The completion algorithm relies heavily on the previous primitive shape object completion methods [26] to generate 3D shapes to plan grasps and deformations.

An original 3D grasp selection algorithm is developed in Section 4.2.1 that plans contact points for shape control in a model-free manner. This original algorithm builds on some of the work presented in 2.4.1 [86, 87, 90] and represents the most significant original contribution from this thesis.

Section 4.2.2 describes the motion planning algorithm to move the end-effector into the grasping position to optimally deform the object at the contact points planned in Section 4.2.1. The algorithm combines the JT-RTT algorithm [63] with the RRT* [59] algorithm and some original contributions to plan motion that brings the end-effector to the grasp points quickly and without collisions.

Chapter 3

Design Considerations

3.1 Proposed Approach

This research’s main objective is to design a system to control the shape of an unknown three-dimensional deformable object using a robotic arm and a three-fingered robotic hand. The outcomes of this work include several original contributions in the field of deformable object manipulation. The most significant original contributions, first detailed in Section 1.2, include:

- An algorithm to plan the contact points on a deformable object to optimally deform it towards a desired shape.
- A collision-free path planning algorithm to place the robotic hand in the optimal valid grasping pose.
- A multi-view assessment strategy to assess the quality of the grasp and deformation.

The proposed approach uses a series of modules that rely on some of the state-of-the-art techniques discussed in Chapter 2 alongside original contributions to the field. An additional outcome of this thesis is a general framework for the robotic manipulation of deformable objects based on visual and force sensing [104], presented in Figure 3.1. This framework details a broad and flexible approach that can be used to define solutions to a variety of deformable object manipulation tasks.

This framework divides the tasks into three main stages: namely the initialization, planning and execution phases. The initialization phase uses visual information from one or more sensors and general scene knowledge to identify the workspace, relative positions of the robot, objects to manipulate and any obstacles that need to be avoided. This stage also includes the task definition by determining the object to manipulate and the desired locations, shapes, or paths to reach the goal.

In the planning phase, the initial knowledge of the object, task and robotic system is used to derive initial grasping points and manipulator trajectories that are expected to complete the defined task. This stage also includes predicting the behaviour of the object as it is manipulated based on data output from the initialization stage.

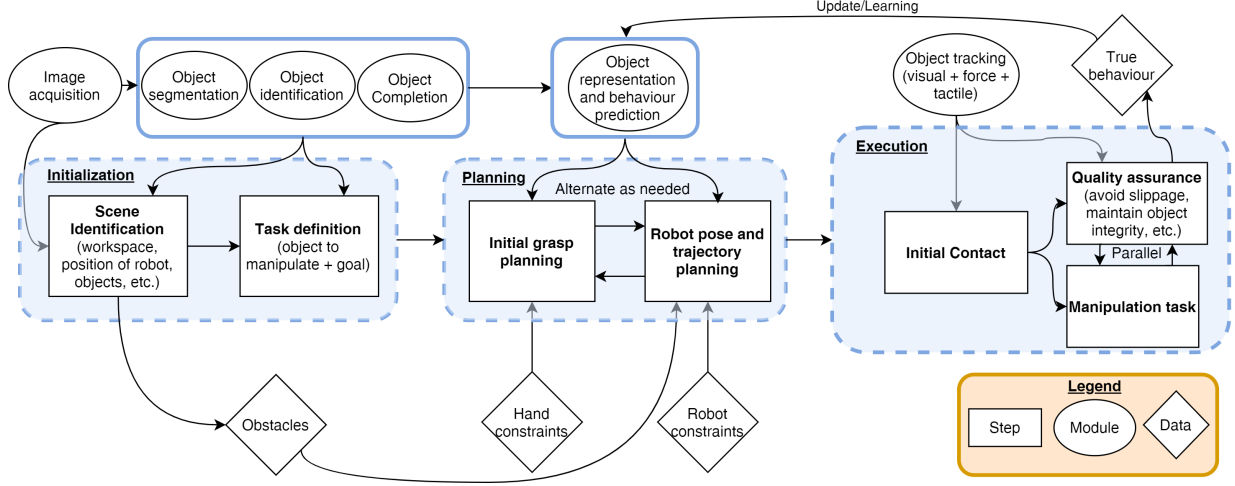


Figure 3.1: Proposed framework for a global approach to manipulation of deformable objects.

Lastly, the execution phase is when initial contact with the object is made, and the manipulation task is performed. This stage includes quality assurance to handle issues such as failed grasps, slippage and maintaining the structural integrity of the object throughout the manipulation. In addition, this stage is responsible for tracking the true behaviour of the object being manipulated and using this data to update any object models and behaviour predictions. These updates can then be used to adaptively replan the control based on new information. This will provide the framework to modify the grasp points or trajectories to give the system increased flexibility in handling objects with uncertain or unknown initial characteristics.

The framework presented in Figure 3.1 and described above is very general and can be refined and applied to a variety of deformable object manipulation tasks. Figure 3.2 gives a more detailed description of the proposed approach that this research follows while fitting into the general framework described above, while Figure 3.3 depicts the physical robotic infrastructure considered in the manipulation process.

This more refined framework is the basis for the remainder of the thesis. The components of each stage will be defined in detail in the subsequent sections.

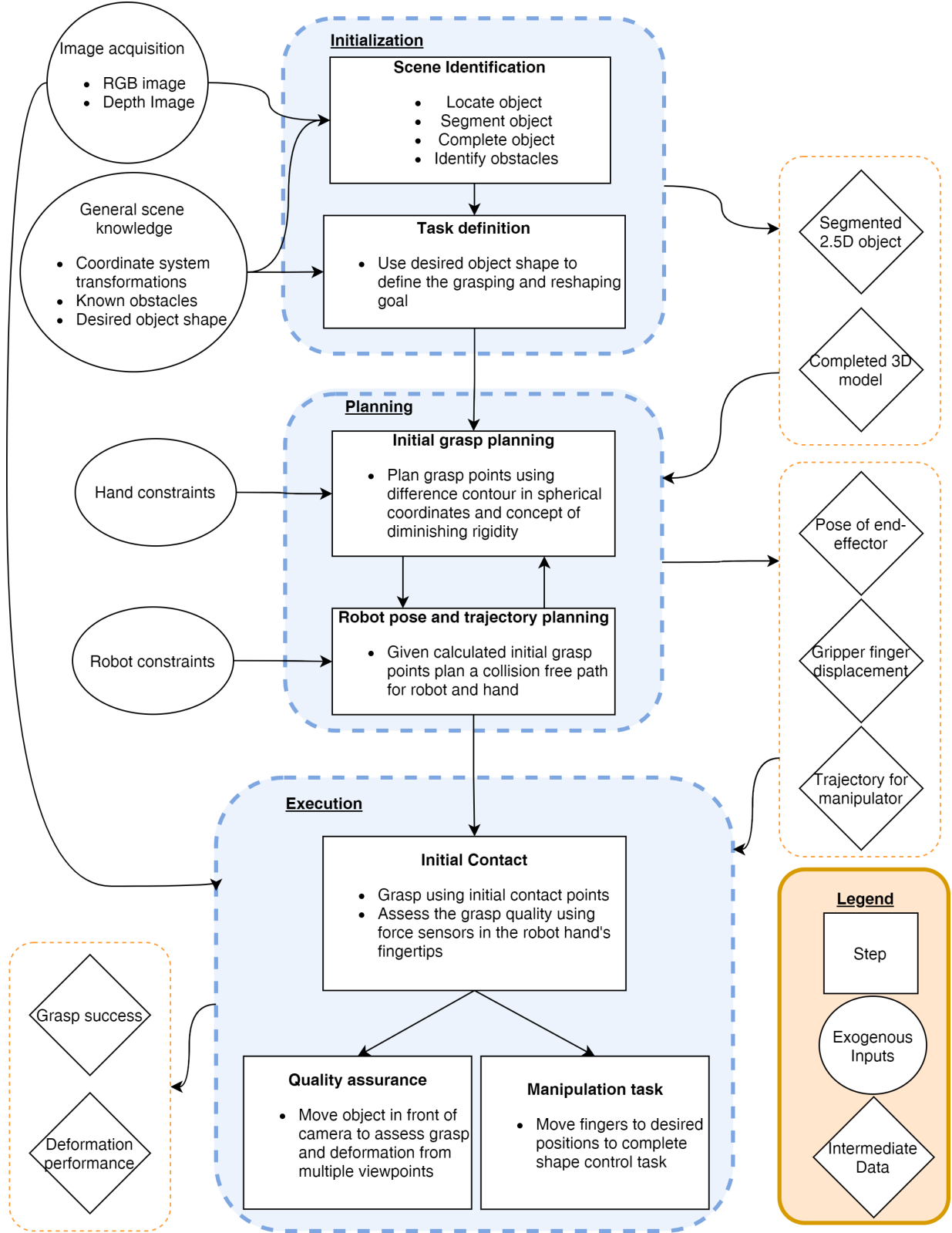


Figure 3.2: Detailed framework for robotic control and manipulation of deformable objects.



Figure 3.3: Physical robot infrastructure used in the proposed solution.

3.2 Equipment

Various tools are required to implement the system described in Section 3.1 and complete the goals outlined in Section 1.2. A 3D depth sensor and colour camera capture images used to locate objects and segment them from the environment. The segmentation results are then used to estimate the 3D geometry of the object. A robot manipulator is used to optimally guide a three-fingered robot hand along a collision-free path to a grasping pose, as seen in Figure 3.3. Once the robot hand is in position, it controls and deforms the object to the desired 3D shape. The manipulator moves the grasped object into a variety of assessment positions. At these positions, the 3D depth sensor and colour camera capture additional data to assess the quality of the deformation of the object. Specifically, the assessment evaluates the change in the similarity between the deformed object under manipulation and a predefined desired shape from the initial level of similarity between the undeformed object and the desired shape.

The system must integrate the tools into a single workspace so they can work together to sense objects and manipulate them. The equipment chosen for implementing the system includes the Intel Realsense D435i depth camera, the CRS F3 robot manipulator and the Barrett Hand. Figure 3.4 shows each component and how they are integrated into a single system.

A detailed description of each piece of equipment and their associated technologies are outlined in the following sections.

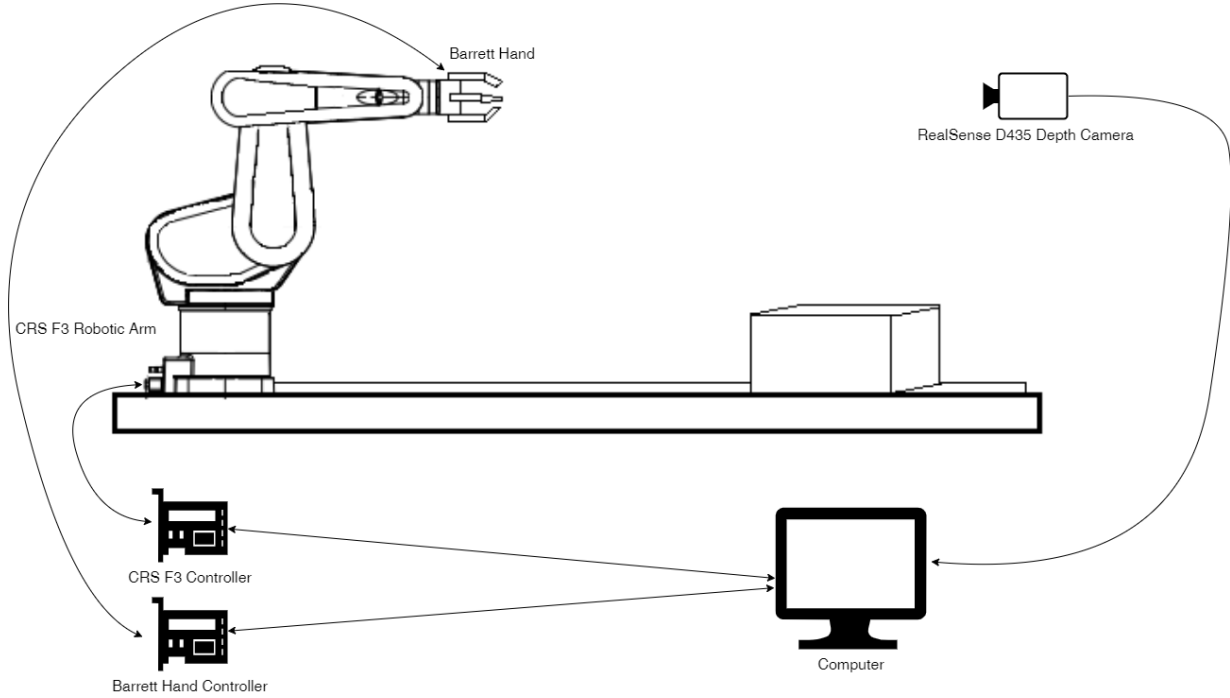


Figure 3.4: System diagram for the proposed robotic manipulation system.

3.2.1 Intel RealSense Depth Camera

The Intel RealSense D435i is a stereo depth camera that relies on two infrared (IR) imagers and an IR projector to generate depth images. The stereo camera takes the two images, one from each IR sensor, and compares features within the images and the known distance between the sensors to calculate the depth at each pixel location. The IR projector emits a random pattern to ensure that the depth camera will still work in low lighting conditions and on scenes with few features. This type of stereo depth camera has become the preferred technology for low-cost 3D depth as they can be used in dynamic scenes, and multiple cameras can be integrated together. Additionally, IR stereo depth sensors are full image sensors meaning the entire Field of View (FoV) is captured at the same time. The camera also includes an RGB module to capture standard colour images. The depth and colour images can be aligned to create 3D textured scenes. These attributes, as well as the low cost, make it a strong choice for research in robotics.

The technical specifications of the depth sensor include a depth FoV of $86^\circ \times 57^\circ (\pm 3^\circ)$ and an ideal range between 0.3 m and 3 m. One of the drawbacks of the low-cost depth sensors is the accuracy of the depth data. The Intel RealSense D435i has an accuracy of up to 2%, meaning the true depth value may differ significantly from the real value when working with small objects placed at a reasonable distance from the camera. Additional technical specifications of the sensor can be found at [105, 106].

An example of the robot workspace captured by the sensor is shown in Figure 3.5. Observing the colourized depth image, it is evident that there is a lack of resolution for many of the objects in the scene; this is due to the accuracy of the depth data. This lack of resolution poses a challenge for accurate object segmentation, 3D object completion and deformation assessment, all of which are discussed in subsequent sections.

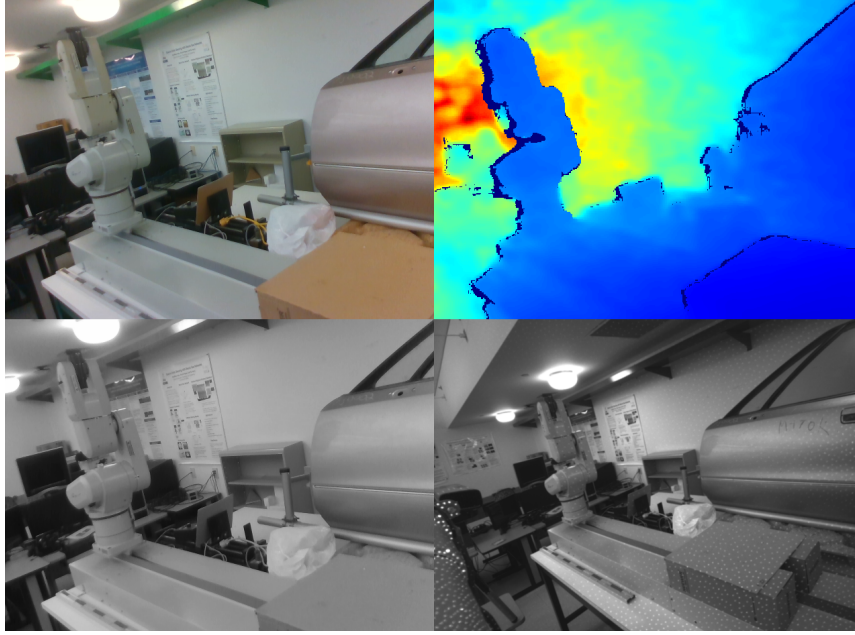


Figure 3.5: Images of the robot and workspace captured from the Realsense D435i camera. Clockwise: Colour, colourized depth, infrared and grayscale.

3.2.2 CRS F3 Robot Manipulator

The CRS F3 robot manipulator is a six-degree of freedom (DoF) robotic arm used to transport payloads and perform motion operations in space. In this system, the robot arm is secured to a linear track to add an additional DoF. Meaning the robot consists of one prismatic joint, D_0 , and six revolute joints, θ_1 through θ_6 . This setup can be seen in Figure 3.5. The CRS F3 robot's controller uses prescribed joint positions to control the manipulator movement. Figure 3.6 shows the robot manipulator's physical characteristics, identifying the joints and their axes of rotation. A reference frame for each link of the manipulator is specified following the Denavit-Hartenberg (DH) convention. The DH parameters associated with the robot are described in Table 3.1.

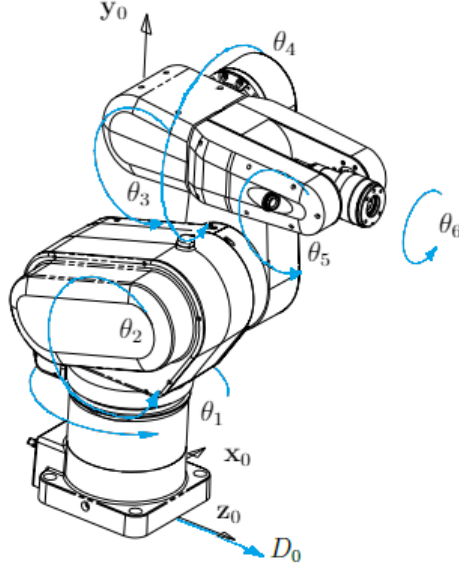


Figure 3.6: The CRS F3 robot manipulator with joint axes identified. Figure adapted from the CRS F3 user guide [107].

Table 3.1: Denavit-Hartenberg parameters for the CRS F3 manipulator ($a_1 = 100$, $d_1 = 350$, $a_2 = 265$, $d_4 = 270$, and $d_6 = 75$)

Joint (i)	a_i	d_i	α_i	θ_i
0	0	D_0	90°	180°
1	a_1	d_1	90°	$\theta_1 + 90^\circ$
2	a_2	0	0°	$\theta_2 + 90^\circ$
3	0	0	90°	$\theta_3 + 90^\circ$
4	0	d_4	90°	$\theta_4 + 90^\circ$
5	0	0	90°	$\theta_5 + 90^\circ$
6	0	d_6	0°	$\theta_6 + 90^\circ$

In this work, a three-dimensional coordinate frame's position and orientation are described using the special Euclidean group $SE(3)$. This group describes a 6D pose, $\mathbf{v}_6$, as a displacement in 3D plus a 3D rotation defined by a set of Euler angles, in this case, the yaw (r_z), pitch (r_y) and (r_x) roll angles.

$$\mathbf{v}_6 = (x, y, z, r_z, r_y, r_x) \quad (3.1)$$

Any pose can also be described using a 4×4 homogeneous transformation matrix with the following structure:

$$T = \left[\begin{array}{ccc|c} & & & x \\ & R & & y \\ & & & z \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (3.2)$$

Where R is the 3×3 orthogonal rotation matrix described by the Euler angles.

The robot's Forward Kinematics (FK) are solved using the DH parameters to describe each link's position and orientation with respect to the base frame. The homogeneous transform matrix A_i describes the position and orientation of the frame associated with link i relative to link $i - 1$. Combining the transformations for all of the manipulator's joints, the matrix T_{07} describes the position and orientation of the end-effector frame with respect to the base link frame. It is given by the expression:

$$T_{07} = A_1 A_2 A_3 A_4 A_5 A_6 A_7 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

Given that the complete robot system is a 7-DoF manipulator, there is redundancy to the solutions of the robot's Inverse Kinematics (IK), meaning that the majority of end-effector poses have infinite joint position solutions. Therefore, traditional inverse kinematic solutions cannot be applied to solve for joint positions of desired end-effector poses. The technique used to solve inverse kinematics of the redundant manipulator is explored in Section 4.2.2, but first, the inverse kinematics for the closed-loop model with a set position of the track will be solved. Fixing the value of D_0 removes a DoF and allows the 6-DoF manipulator to have a closed-loop inverse kinematic solution that can be used to find precise joint positions for poses within the robot's current workspace. The precise joint positions from the inverse kinematic solution are used in Section 4.2.2 to move to exact configurations when the track is fixed.

This thesis develops a closed-loop inverse kinematic solution for the 6-DoF CRS F3 robot using techniques described in [108]. This involves determining the values of θ_1 through θ_6 using the forward kinematic model from Equation 3.3 and a desired end-effector position and orientation.

The first step in the inverse kinematic solution is to calculate the vector P_{06} , the position of the robot's wrist with respect to the base link.

$$P_{06} = T_{07} \begin{bmatrix} 0 & 0 & -d_6 & 1 \end{bmatrix}^T \quad (3.4)$$

Which leads to the following equations for the position of the wrist:

$$o_{x,06} = [a_1 - a_2 \sin(\theta_2) - d_4 \sin(\theta_2 + \theta_3)] \sin(\theta_1) \quad (3.5)$$

$$o_{y,06} = a_2 \cos(\theta_2) + d_1 + d_4 \cos(\theta_2 + \theta_3) \quad (3.6)$$

$$o_{z,06} = [a_1 - a_2 \sin(\theta_2) - d_4 \sin(\theta_2 + \theta_3)] \cos(\theta_1) \quad (3.7)$$

Solving for θ_1 yields:

$$\theta_1 = \text{atan2}(o_{x,06}, o_{z,06}) \text{ or } \theta_1 = \text{atan2}(o_{x,06}, o_{z,06}) \pm \pi \quad (3.8)$$

To solve for θ_2 and θ_3 the matrix T_{01}^{-1} is calculated. This matrix represents the position and orientation of the base link with respect to link two given a value for θ_1 . This matrix is used to generate the matrix T'_{16} which represents the position of the wrist in the orientation of the end-effector with respect to link two.

$$T'_{16} = T_{01}^{-1} T'_{06}$$

Which leads to the equations for the position of the wrist with respect to joint two:

$$o_{x,16} = -a_2 \sin(\theta_2) - d_4 \sin(\theta_2 + \theta_3) \quad (3.9)$$

$$o_{y,16} = a_2 \cos(\theta_2) + d_4 \cos(\theta_2 + \theta_3) \quad (3.10)$$

These equations can be manipulated to solve for θ_3 by first solving:

$$\cos(\theta_3) = \frac{o_{x,16}^2 + o_{y,16}^2 - a_2^2 - d_4^2}{2a_2d_4} \quad (3.11)$$

With $-1 \leq \cos(\theta_3) \leq 1$:

$$\sin(\theta_3) = \pm \sqrt{1 - \cos^2(\theta_3)} \quad (3.12)$$

Where the positive sign corresponds to the solution in the positive half plane and the negative sign corresponds to the solution in the negative half plane, leading to:

$$\theta_3 = \text{atan2}(\sin(\theta_3), \cos(\theta_3)) \quad (3.13)$$

The other joint variable from these equations can also be computed using the value of θ_3 and manipulating the equations further:

$$\sin(\theta_2) = -\frac{o_{x,16}(a_2 + d_4 \cos(\theta_3)) + o_{y,16}(d_4 \sin(\theta_3))}{o_{x,16}^2 + o_{y,16}^2} \quad (3.14)$$

$$\cos(\theta_2) = \frac{o_{y,16}(a_2 + d_4 \cos(\theta_3)) - o_{x,16}(d_4 \sin(\theta_3))}{o_{x,16}^2 + o_{y,16}^2} \quad (3.15)$$

$$(3.16)$$

$$\theta_2 = \text{atan2}(\sin(\theta_2), \cos(\theta_2)) \quad (3.17)$$

The remaining joint variables can be computed from the transformation matrix of the final three joints, T_{36} , while using the previously calculated joint variables to solve for the matrix T_{03} .

$$T_{36} = T_{03}^{-1} T_{06} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.18)$$

With the following terms fully defined:

$$r_{13} = \cos(\theta_4) \sin(\theta_5) \quad (3.19)$$

$$r_{23} = \sin(\theta_4) \sin(\theta_5) \quad (3.20)$$

$$r_{31} = \sin(\theta_5) \cos(\theta_6) \quad (3.21)$$

$$r_{32} = -\sin(\theta_5) \sin(\theta_6) \quad (3.22)$$

$$r_{33} = \cos(\theta_5) \quad (3.23)$$

Leading to solutions for the remaining joint variables:

$$\theta_4 = \text{atan2}(r_{23}, r_{13}) \quad (3.24)$$

$$\theta_6 = \text{atan2}(-r_{32}, r_{31}) \quad (3.25)$$

$$\theta_5 = \text{atan2}(r_{31}, \cos(\theta_6)r_{33}) \quad (3.26)$$

Depending on the manipulator pose, there may exist a number of possible solutions for the joint variables. In order to determine whether the solution is valid, it must be within the workspace of the robot and be within the joint limits for each joint. If there exists a solution, the result will be a set of allowable joint configurations that satisfy the inverse kinematic problem.

In the case where the track is included with the robot, a 7-DoF inverse kinematic solution can be developed using the Jacobian of the robot manipulator. The Jacobian of the robot is the time derivative of the forward kinematics of the manipulator, relating the rate of change of the joint angles and positions to the angular and linear velocities of the end-effector. Using the inverse of the Jacobian matrix, the joints can be changed at specified rates to move the end-effector in a prescribed direction or to a specific pose. This technique will be described in detail and used when solving for the inverse kinematics of the redundant manipulator in Section 4.2.2.

The Jacobian of the manipulator can be broken up into two components, linear and angular. Each component is a $3 \times n$ matrix, where n is the DoF of the robot, relating the rate of change of the joint angles and positions to the linear and angular velocities of the end-effector. The equation for the linear Jacobian, J_L :

$$J_{L_i} = \begin{cases} \mathbf{z}_{i-1} \times (\mathbf{o}_n - \mathbf{o}_{i-1}) & \text{if joint } i \text{ is revolute} \\ \mathbf{z}_{i-1} & \text{if joint } i \text{ is prismatic} \end{cases} \quad (3.27)$$

And the angular Jacobian, J_ω :

$$J_{\omega_i} = \begin{cases} \mathbf{z}_{i-1} & \text{if joint } i \text{ is revolute} \\ 0 & \text{if joint } i \text{ is prismatic} \end{cases} \quad (3.28)$$

Where $\mathbf{z}_i$ is the orientation of the z -axis of joint i in the base coordinate frame and $\mathbf{o}_i$ is the position of the origin of joint i in the base coordinate frame.

3.2.3 Barrett Hand

The Barrett Hand is a three-fingered programmable end-effector used to grasp target objects of a variety of different shapes and sizes. The end-effector is an impactful gripper using forces from each finger to impact against the surface of the object that is to be manipulated. The three fingers are labelled F1, F2 and F3, where F3 is fixed, and F1 and F2 can rotate around a fixed point on the palm. To model the forward kinematics of the hand, each finger is considered a robot manipulator with a reference coordinate frame at the centre of the wrist. The hand model used in this research is the BH8-280. Figure 3.7 shows the hand attached to the CRS F3 robot with the hand joint axes identified.

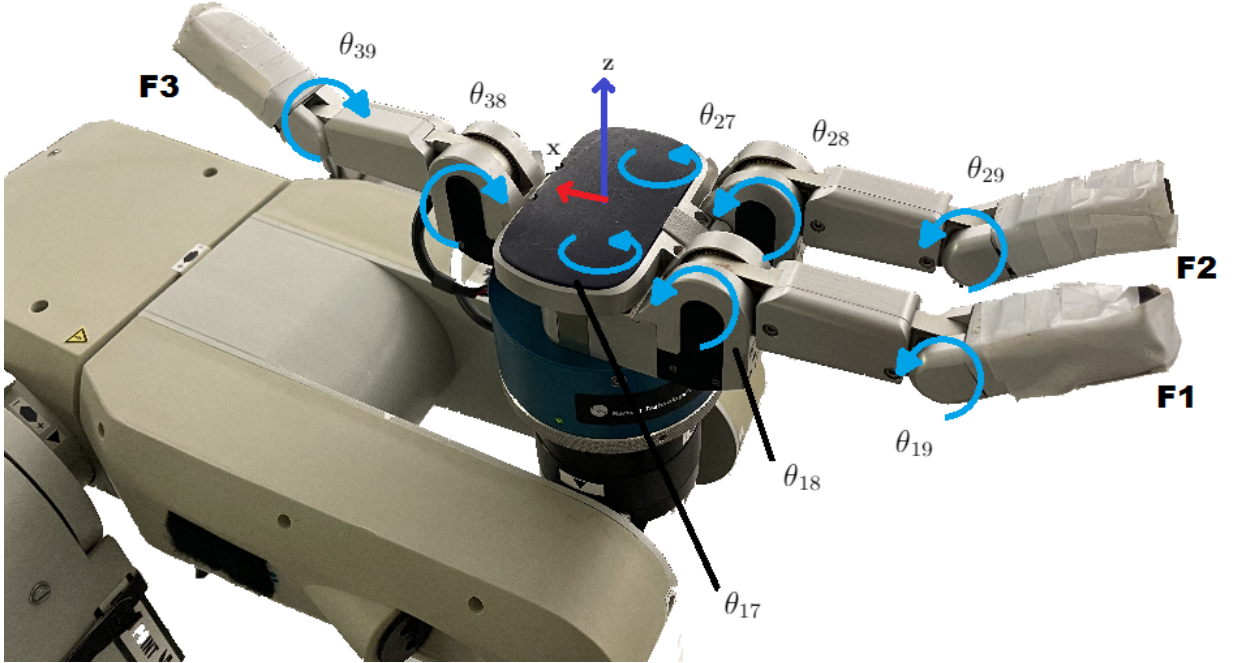


Figure 3.7: The BH8-280 three-fingered gripper with joint axes identified.

The forward kinematics are solved in the traditional way using the DH parameters in Table 3.2.

Where k indicates the finger of the hand, either F1, F2 or F3 for $k = \{1, 2, 3\}$. Variables $\mathbf{g}$, and $\mathbf{j}$ are parameters for each finger that define the location of the base of the finger with respect to the centre of the wrist, $\mathbf{g} = [-1, 1, 0]$ and $\mathbf{j} = [1, 1, -1]$.

The Barrett Hand has significant constraints for motion control of the fingers. Firstly, the spread of the fingers is controlled equally for both of the non-fixed fingers. This means that when the fingers are rotated, F1 and F2 move in equal and opposite directions by θ_7 .

$$\theta_7 = |\theta_{17}| = |\theta_{27}| \quad (3.29)$$

Table 3.2: Denavit-Hartenberg parameters for all fingers of the Barrett Hand ($a_0 = 25$, $a_1 = 50$, $a_2 = 70$, $a_3 = 50$, $d_0 = 84$, $d_3 = 9.5$, $\phi_2 = 0.4^\circ$, and $\phi_3 = 42^\circ$).

Joint (i)	a_i	d_i	α_i	θ_i
0	$g_k a_0$	d_0	0°	0°
1	a_1	0	90°	$g_k \theta_{k7} - j_k 90^\circ$
2	a_2	0	0°	$\theta_{k8} + \phi_2$
3	a_3	d_3	-90°	$\theta_{k9} + \phi_3$

The second constraint is that the joints used to close the fingers cannot be controlled independently. The joint position for each finger are controlled by a single encoder. The following equations relate the encoder position to the joint positions for θ_{k8} and θ_{k9} [109].

$$\theta_{k8} = \frac{e_k}{195000} \cdot 140 \quad (3.30)$$

$$\theta_{k9} = \frac{e_k}{195000} \cdot \frac{140}{3} \quad (3.31)$$

Where e_k is the number of encoder ticks for finger k . This value ranges from 0 to 195000 for the particular model in question. This results in joint motion limits for each finger joint to be $\theta_{k8} \in [0^\circ, 140^\circ]$ and $\theta_{k9} \in [0^\circ, 46.7^\circ]$. The complete configuration for the hand can be uniquely described by four variables: θ_7 , e_1 , e_2 and e_3 .

The forward kinematic model for each finger is solved using the DH parameters and the joint constraints to describe the position and orientation of each fingertip with respect to the centre of the wrist. The matrix T_{W9}^k describes the position and orientation of the fingertip frame of finger k with respect to the centre of the wrist. It is given by the expression:

$$T_{W9}^k = \begin{bmatrix} r_{11}^k & r_{12}^k & r_{13}^k & t_x^k \\ r_{21}^k & r_{22}^k & r_{23}^k & t_y^k \\ r_{31}^k & r_{32}^k & r_{33}^k & t_z^k \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.32)$$

Because of the constraints on the movements of the fingers, the end-effector may be underactuated for certain tasks. The major constraint that each finger is subject to is the relationship between the distance from the palm to the fingertip along the z -axis and the distance from the centre of the palm to the fingertip along a vector parallel to the xy -plane. This second distance will be referred to as the grasp radius, t_r^k .

$$t_r^k = \sqrt{t_x^{k2} + t_y^{k2}} \quad (3.33)$$

The encoder positions are uniquely defined to move the fingertip to the desired grasp radius. In turn, the encoder positions determine the z distance for each finger. The inverse

of this can also be utilized to find the encoder positions for a desired z distance, and the corresponding grasp radius.

There is no closed-form solution for the inverse kinematics of the Barrett Hand. Therefore the inverse kinematic solutions are solved numerically using the range values for the encoders. For a fixed value of θ_7 the encoder value, e_k to generate a desired radius, r :

$$e_k = \arg \min_{e_k} |(t_r^k(\theta_7, e_k) - r)| \quad (3.34)$$

Or the encoder value for a desired z distance:

$$e_k = \arg \min_{e_k} |(t_z^k(\theta_7, e_k) - z)| \quad (3.35)$$

The Barrett Hand also contains three force sensors that measure externally applied torques at the finger joint θ_{k8} . The sensor measures external forces over a range of ± 1 Nm [109] applied to the fingertips. The torque sensors are used to measure the forces applied by the deformable objects being grasped.

3.3 Objects

A set of deformable objects of various primitive shapes and sizes were chosen for use in the grasping and shape control experiments. The primitive shapes chosen in this research are spheres, cylinders and rectangular cuboids (boxes). The objects are shown in Figure 3.8. Each object deforms by varying amounts when they are handled, but the exact material properties of the objects are unknown. Table 3.3 presents a brief description of each object.

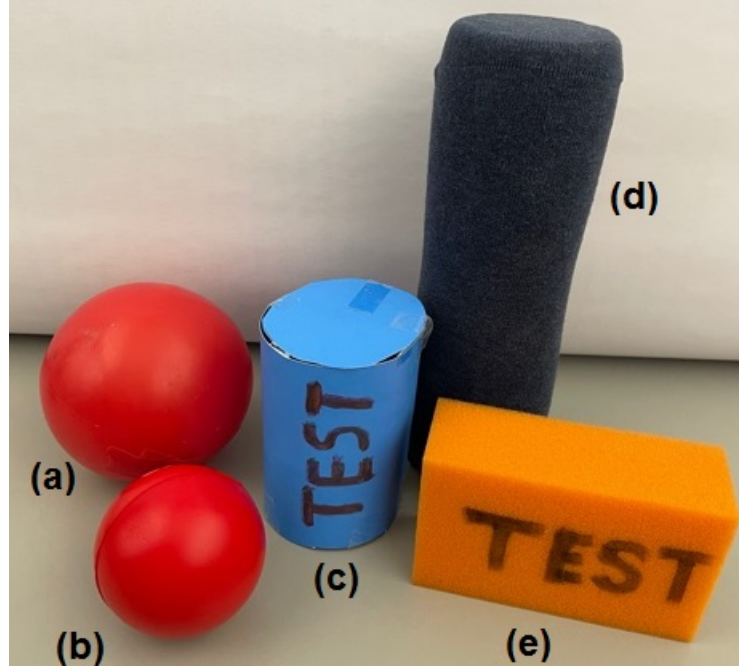


Figure 3.8: Objects used in shape control experiments.

Table 3.3: Description of the set of deformable objects.

Label	Name	Material	Dimensions [mm]
a	Large Ball	Foam	$r = 52$
b	Small Ball	Rubber	$r = 34$
c	Small Cylinder	Paper	$r = 35, h = 108$
d	Large Cylinder	Stiff Foam	$r = 37, h = 227$
e	Sponge	Foam	$l = 124, w = 44, h = 75$

Throughout the research, objects of various colours but identical dimensions and material properties were substituted to test the robustness of the image processing algorithms.

This chapter defines the general design consideration and physical constraints imposed by the robotic devices used in this research. For that matter, formal kinematic models for the CRS F3 manipulator and the BH8-280 Barrett hand are developed, which will be used to support the methodologies proposed in Chapter 4.

Chapter 4

Robotic Control of Deformable Objects

This chapter develops the proposed robotic grasping system module-by-module. Working off of the framework developed in Section 3.1 and illustrated in Figure 3.2, the modules are broken up into initialization, planning and execution phases. During the development of each module, one example is carried throughout the chapter to help develop the methodologies and algorithms used by the system. Additional supporting examples are presented where appropriate.

Once the system has been developed, Chapter 5 presents end-to-end examples of grasping and shape control on a variety of test cases.

4.1 Initialization

4.1.1 Coordinate System Alignment

In the system framework’s initialization phase, the data captured from the RGB-D sensor is used to identify objects in the robot workspace. For the system to plan robot motion to grasp the objects, their coordinates must be transformed from the camera coordinate system to the robot coordinate system. The coordinate system alignment is done by collecting data with known locations in both the camera and robot coordinate systems and then solving for the transformation that minimizes the error between the sets of points.

A set of known positions in the camera coordinate system are identified by attaching a ball with a known radius to the end-effector and recording the position of the centre of the ball with the 3D camera. The ball is fixed to the open palm of the hand with Velcro to keep it in position throughout the experiment. The fingers of the hand are opened fully to avoid occluding the ball from the camera. The ball’s centre is located by applying a mask to the colour image that selects pixels close to the same colour as the ball. A Hough transform is used on the mask to find a circle that identifies the ball’s location. The centre

of the ball is then projected into 3D space using the depth image, and the centre is offset by the radius of the ball. This procedure gives the 3D location of the centre of the ball in the camera coordinate system.

The matching set of points in the robot coordinate system is generated by calculating the end-effector position and orientation using the joint encoder positions and the robot's forward kinematic model. The end-effector position is offset by the radius of the ball to identify the 3D location of the centre of the ball in the CRS F3 robot coordinate system.

The robot is programmed to move to various predefined poses within the workspace visible to the 3D sensor. Once the robot and 3D sensor have recorded the data at all positions, the data is used to calculate the optimal coordinate system transformation. One of the poses used to align the coordinate systems can be seen in Figure 4.1.

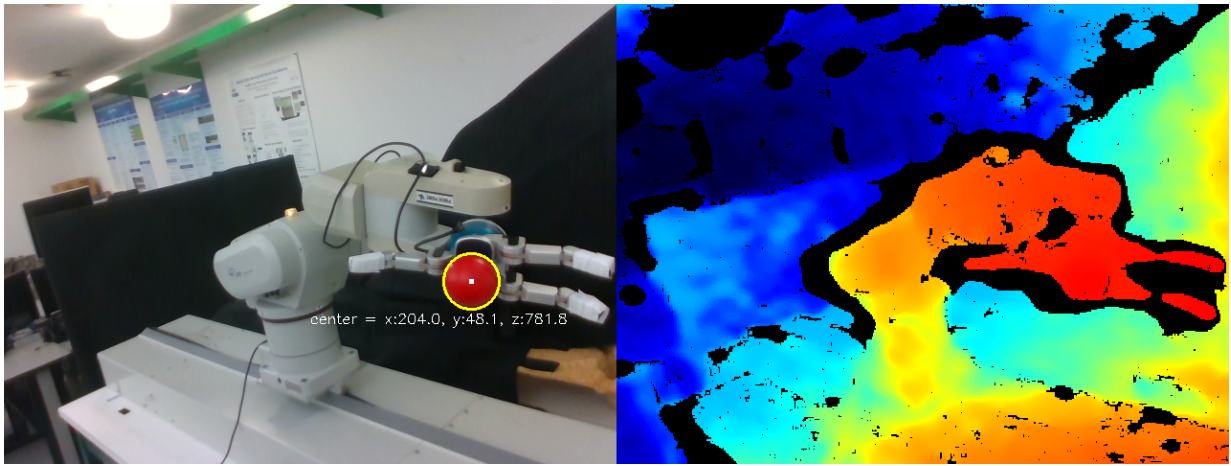


Figure 4.1: Colour and colourized depth images of the CRS F3 robot with a ball of known radius fixed to the palm of the Barrett Hand for coordinate alignment.

The Kabsch algorithm [110] is used to calculate the optimal rotation matrix that minimizes the Root Mean Square Error (RMSE) between two sets of paired points. Let the two sets of points be referred to as P , for points reported by the CRS F3 robot, and Q , for points calculated from the 3D sensor. The translation between the two sets of points is computed by translating each set of points by its centroid so that both sets align with the origin. Next, the covariance matrix, H , is calculated:

$$H = P^T Q \quad (4.1)$$

The optimal rotation matrix is calculated by taking the Singular Value Decomposition (SVD) of the covariance matrix. SVD breaks down any invertible linear transformation into a composition of three geometrical transformations, which generate the optimal rotation matrix.

$$H = U \Sigma V^T \quad (4.2)$$

U and V both represent a reflection or rotation, and Σ is the coordinate-by-coordinate scaling factor. Thus, the optimal rotation from the camera coordinate system to the robot coordinate system, R , can be calculated as:

$$R = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} V^T \quad (4.3)$$

Where d is $+1$ or -1 to ensure a right handed coordinate system.

The rotation matrix can be expressed as a homogeneous transformation matrix by adding the translation component between the two coordinate systems.

$$\mathbf{t} = \mathbf{p}_o - (R\mathbf{q}_o)^T \quad (4.4)$$

The terms $\mathbf{p}_o$ and $\mathbf{q}_o$ represent the centroids of the sets of points P and Q , respectively.

$$T_{cam/rob} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \quad (4.5)$$

This matrix, $T_{cam/rob}$, is used to transform points from the camera coordinate system to the robot coordinate system. Additionally, $T_{cam/rob}$ is used to check the error in the coordinate system alignment.

To align the coordinate systems, twelve positions for the robot were chosen where the fixed ball is fully visible to the camera. The positions were chosen in workspace regions where objects may be grasped or assessed in subsequent steps. The ball was identified in the camera and robot coordinate systems using the respective procedures outlined above. The alignment procedure was run five times, using the same 12 positions each time, to ensure consistency in the results. The data from one alignment experiment is shown in Table 4.1.

Table 4.1: Position data from camera (Q) and robot (P) during coordinate system alignment and the resulting transformed camera coordinates ($T_{cam/rob} Q$).

Position Number	Q [mm]			P [mm]			$T_{cam/rob} Q$ [mm]		
	x	y	z	x	y	z	x	y	z
1	200.9	48.6	789.9	-201.3	396.8	1507.8	-202.9	393.0	1507.4
2	227.3	50.3	756.5	-201.3	396.8	1547.8	-204.5	394.2	1550.0
3	225.6	135.9	663.6	-285.1	334.9	1620.0	-287.2	333.4	1623.7
4	172.6	-72.2	629.6	-282.3	551.1	1604.3	-288.3	549.9	1603.2
5	318.4	-117.5	918.9	14.5	500.7	1479.4	10.0	501.9	1477.7
6	339.2	31.9	905.8	-24.5	356.0	1509.5	-25.0	358.1	1509.4
7	247.9	29.0	1014.0	-24.5	356.1	1369.4	-24.7	352.9	1367.8
8	251.8	-165.2	996.5	30.3	538.9	1373.0	22.3	542.1	1372.9
9	375.6	-90.9	1048.9	126.4	434.2	1419.0	124.6	437.0	1418.4
10	378.5	-220.6	1033.3	147.9	567.1	1424.7	153.8	564.2	1424.9
11	-11.8	-3.2	749.0	-366.1	493.2	1396.1	-369.8	490.0	1396.4
12	7.3	48.3	691.9	-431.1	441.0	1455.6	-405.4	450.1	1454.9

The accuracy of the coordinate alignment is examined by looking at the RMSE between one set of coordinates and the other after it has been transformed. The error, $e_{cam/rob}$, is calculated as follows:

$$e_{cam/rob} = \sqrt{\frac{1}{N} \sum_{x,y,z} \sum_{n=1}^N (T_{cam/rob} Q_n - P_n)^2} \quad (4.6)$$

Where N is the number of positions, Q_n is the position of the ball from the 3D sensor at position n , P_n is the position of the ball from the robot at position n and $T_{cam/rob}$ is the optimal rigid transform calculated in Equation 4.5.

For the positional data in Table 4.1, the resulting RMSE was 9.28 mm. The mean RMSE of all five experiments was 9.93 mm with a standard deviation of 1.07 mm. The reported error is mainly from inaccuracies in the 3D sensor. The sensor’s documentation [105] shows that there may be up to a 2% error in the depth measurements, which increases with distance from the sensor. The average distance from the sensor during the alignment procedure was 891.2 mm, resulting in an average error of 1.11%, which is within the manufacturer’s tolerance. It is possible that some error is also a result of the positions reported by the robot as the precision and accuracy of the robot encoders have not been estimated. This amount of error may give rise to significant errors in estimating the shape and position of small objects, as well as assessing small deformations in the object during manipulation. For this reason, objects that are significantly larger than the error were chosen for the experiments in this research, as per Table 3.3.

4.1.2 Object Segmentation Using RGB-D Data

For a robotic system to perform any object manipulation task autonomously, it must be able to locate an object of interest within an undefined environment. In this part of the initialization phase, colour and depth images are used to identify and segment a previously unseen object from the background. Once this procedure is complete, further planning steps can proceed.

In this work, the 3D sensor acquires a single RGB-D image of the robot workspace that contains the object of interest. The sensor is positioned to capture the majority of the robot workspace but is placed close enough to the workspace such that the depth accuracy is not compromised. Capturing the majority of the workspace allows the system to grasp objects in various locations accessible to the robot. Although the sensor captures the majority of the workspace, the images are manually cropped to a region of interest by a user as there is significant undesired content in the periphery of the images. To fully automate the process, an additional step to locate a region of interest using visual attention could be introduced prior to object identification.

The identification and segmentation techniques leverage both the colour and depth images to handle objects that contain variable colours, textures or are exposed to uneven illumination, similar to many of the state-of-the-art techniques described in Section 2.1. The depth data is used to generate a boundary probability estimate to accurately determine whether edges detected in the images correspond to boundaries between objects. The algorithm is inspired by the techniques described by [14, 15, 17] and is presented in detail in Section 4.1.2.1.

The proposed segmentation method combines the calculated boundary probability with the colour image data to segment the object from the background using the fast level sets algorithm [9] with an original cost function. The boundary probability augmented fast level sets algorithm represents an original contribution from the thesis to segment objects using depth and colour images. The traditional fast level sets algorithm is described in detail in Section 4.1.2.2, followed by the novel approach in Section 4.1.2.3.

4.1.2.1 Boundary Probability Detection

The object segmentation procedure’s first step is to identify all edges in the colour and depth images. This is done by first converting the depth image to a grayscale image, where pixel intensity corresponds to the distance from the sensor. Edges are detected in the grayscale image and each channel of the colour image using a Sobel filter. For a single-channel image, the change in pixel intensity in the horizontal direction, G_x , is determined by convolving the image, I , with a horizontal Sobel kernel, K_x . Approximating the derivative with a kernel size of three is computed as:

$$G_x = I * K_x \quad (4.7)$$

$$G_x = I * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (4.8)$$

Similarly, the change in pixel intensity in the vertical direction, G_y , is determined by convolving the image, I , with a vertical Sobel kernel, K_y .

$$G_y = I * K_y \quad (4.9)$$

$$G_y = I * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (4.10)$$

The magnitude of the change in intensity, G , can be calculated by combining the horizontal and vertical derivatives.

$$G = \sqrt{G_x^2 + G_y^2} \quad (4.11)$$

The direction of the change in intensity, Θ , can also be calculated:

$$\Theta = \text{atan2}(G_y, G_x) \quad (4.12)$$

Once the gradient has been calculated, edges are identified by applying a threshold to the gradient magnitudes. For this particular application, a low threshold is used to keep the majority of the edges. This process is repeated for each image channel (R, G, B, D), and the edges from all channels are combined to generate a complete edge map. If edges are found in the same location in multiple channels, the highest magnitude edge and its direction are used. The resulting edge map is a binary image where edges, e , are represented by pixel values of one. These edges are then assessed further by taking advantage of depth measurements to determine whether they represent a true boundary between the object and the background.

The 3D point cloud generated from the captured depth image is leveraged to analyze the geometry of the scene at each edge pixel. The geometry on either side of an edge is used to determine the boundary probability, P_b , using the technique described in [14]. A circular mask is split into two sections and is positioned on the edge pixel along the direction of the edge to generate two distinct regions. Edge direction, Θ , is quantized into one of eight possible direction bins to simplify the process. Figure 4.2a shows the masks used for each possible edge direction. Placing the mask at the edge pixel, the pixels that

are in each section of the mask are grouped together, forming two clusters. The RANSAC algorithm is used to compute the plane that best fits the 3D data in each cluster.

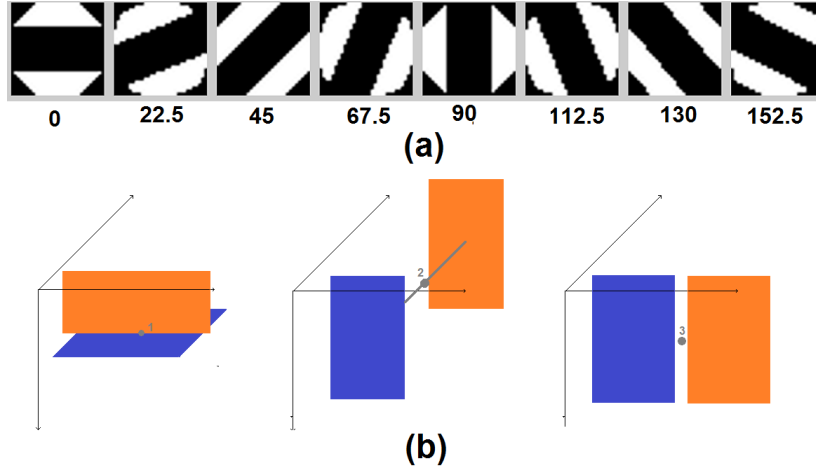


Figure 4.2: Circular masks and edge types for probability boundary detection. (a) All edge directions for the circular masks used to get the geometry on both sides of an edge point. (b) Different plane orientations for boundary edges, depth edges and internal edges respectively.

The relationship between these planes is used to determine the probability that an edge pixel represents a boundary between objects. Figure 4.2b shows a variety of ways the planes can be related. If the planes intersect with an angle between 65° and 115° , this is deemed a contact boundary between two objects, and a boundary probability of 1 is assigned to the edge. Otherwise, the mean distance between the two point clusters, $d_\perp$, is determined and used as input to a simple logistic function, L , to convert the distance to a boundary probability.

$$L(d_\perp) = \frac{1}{1 + e^{-\beta_1(d_\perp - \beta_2)}} \quad (4.13)$$

If the distance is large, this results in a probability close to one, as it most likely indicates a depth discontinuity. If the distance is small, then it most likely corresponds to an internal edge, and the resulting probability is close to zero. The parameters β_1 and β_2 were determined through trial and error and relate to the distance between clusters of points that should lead to a high probability of an edge representing an object boundary. The chosen parameters are robust to object shape but are specific to object size and the sensor's unit of measurement. For the robotic grasping application, the objects are all similar sizes (see Table 3.3), as they are all to be grasped by the gripper, so fixed values of $\beta_1 = 74$ and $\beta_2 = 0.08$ are sufficient. In other applications, where the objects vary in size significantly, fixed values of β_1 and β_2 may not be appropriate.

Following this process at each edge pixel, the boundary probability, P_b , is determined for the entire scene. Similar to the initial edges, a threshold is applied to P_b to remove

any elements that have a low probability of representing an object boundary. The steps as described above are outlined in Algorithm 1.

Algorithm 1: Probability boundary detection

Input:
 I_{RGB} - colour image
 I_D - depth image
Output:
 P_b - probability boundary

```

1  $G, \Theta \leftarrow \text{CalcGradient}(I_{RGB}, I_D)$ 
2  $G = \text{Threshold}(G)$ 
3 Initialize  $P_b = 0$ 
4 foreach edge  $e \in G$  do
5   Fit planes to each side of edge
6   if  $65^\circ < \text{angle between planes} < 115^\circ$  then
7      $P_b(e) = 1$ 
8   else
9      $d_\perp = \text{abs}(\text{mean distance between point cloud clusters})$ 
10     $P_b(e) = \frac{1}{1 + e^{-\beta_1(d_\perp - \beta_2)}}$ 
11   end
12 end
13  $P_b = \text{Threshold}(P_b)$ 

```

The probability boundary, P_b , is used in combination with other image segmentation techniques to segment the desired object from the background. The probability boundary is first used to find an estimated centre of the object that is being segmented by combining the detected boundary edges with some general scene knowledge.

The boundary edges of the map, P_b , identify the intersection of two surfaces that form an angle close to 90° (identified by $P_b(e) = 1$). In general, these edges represent the interface between an object and the surface that it sits on. For example, an object sitting on a table or shelf, or alternatively an object stuck to a wall or other vertical surface. Depending on the shape of the object, there may also be edges within the object that meet this criteria. For example, a rectangular sponge where the faces are at right angles to each other. The mean of all boundary edges is calculated to determine an estimate for the object's centre. The estimate can be improved by adjusting the centre to move further inside the object using the direction along the normal of the surface that the object is on. This technique was developed in [14] for objects that sit on tabletop surfaces, but it is general enough to work when objects are in a variety of positions. This technique fails when a boundary probability map does not contain any boundary edges. This may be the case for a spherical object that is suspended in the air. If no boundary edges exist, the centre is estimated using the mean of all non-zero elements of P_b .

With the estimated centre point identified, the probability boundary is filtered to better represent the edges in the image. The idea of filtering the probability boundary to better represent true boundaries was first presented in [16]. The concept is used in this work

but implemented in a more computationally efficient manner by filtering the edges with an asymmetric Gaussian filter.

The Gaussian filter, G_{σ_x, σ_y} , is generated with variances σ_x^2 and σ_y^2 :

$$\sigma_y^2 = \alpha \cdot \sigma_x^2 \quad (4.14)$$

Where α is a real valued constant greater than one. When applying the filter, it is rotated to match the direction of each edge in the probability boundary image. The filter is applied to all non-zero values of the probability boundary image using the indicator function $\mathbf{1}_{P_b > 0}$.

$$W = G_{\sigma_x, \sigma_y} * (P_b \cdot \mathbf{1}_{P_b > 0}) \quad (4.15)$$

This type of filter increases the number of edge points in the neighbourhood of a detected edge along its detected direction. The filter is useful to get a better definition of real edges in the image.

Figure 4.3 shows the colour and depth images of a single object placed in a grasping area in the robot workspace. These images are used to generate the probability boundary map using the techniques described above. Figure 4.4 shows the step-by-step results of the edge detection and boundary probability calculations.

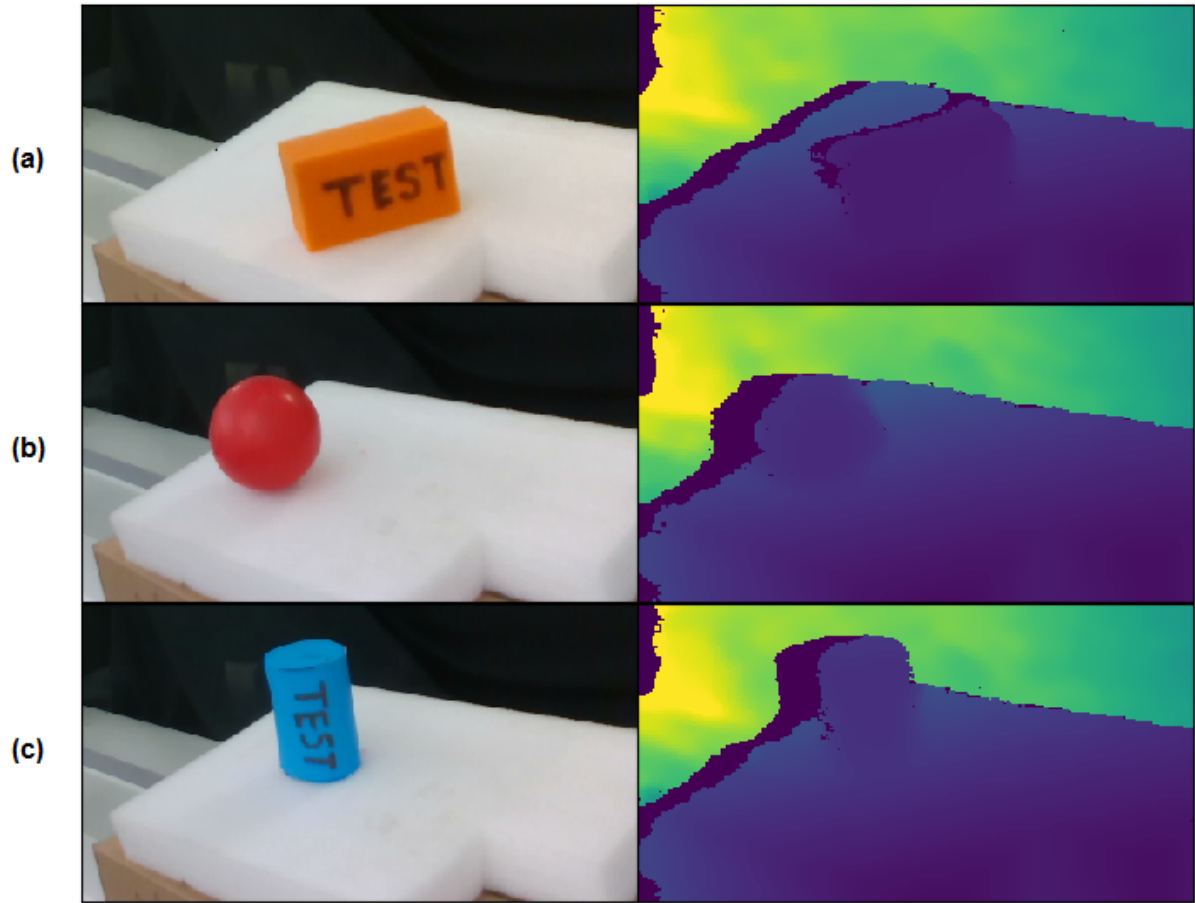


Figure 4.3: Cropped colour and colourized depth images for objects in a grasping area. (a) Box, (b) Large Ball and (c) Small Cylinder.

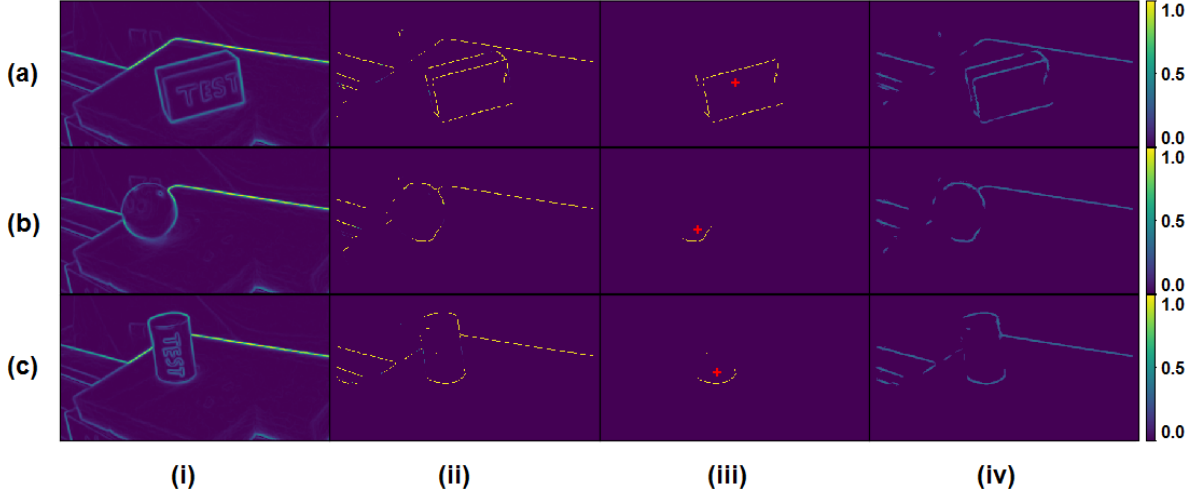


Figure 4.4: Edge images and probability boundary map for cropped object images. (i) Gradient magnitude of all identified edges from the colour and depth images (scaled 0 to 1), (ii) boundary probability map, (iii) boundary edges only with estimated object centre, and (iv) filtered boundary probability map.

Examining Figures 4.4aii and 4.4cii, it can be seen that the probability boundary map removes the internal edges that were present purely from changes in colour as they do not represent geometric boundaries. The majority of the remaining edges of the object are identified as boundaries with varying probabilities. Some edges that should be identified as boundaries are not. This can be in part due to the error associated with the depth image. When planes are being mapped to the depth data, they occasionally do not meet the criteria to be boundary edges when they would be if the data was more accurate. In Figure 4.4aiii, many of the boundary edges are identified correctly, including edges at the top of the object where a right angle is formed. The correct identification of the boundary edges allows a point inside the object to be used as the estimate for the centre of the object in all test cases. The filtered probability boundary smooths the edges so that they can be better utilized in the segmentation algorithm.

4.1.2.2 Traditional Fast Level Sets

As the system is looking for objects to grasp and manipulate, it is intuitive to use a segmentation technique that uses contours to segment objects from the background. The algorithm chosen to segment the image into object and background regions is the fast level sets algorithm [9], briefly presented in Section 2.1.

Using the estimated centre of the object, calculated in Section 4.1.2.1, the entire image, I , can be warped to a polar coordinate representation, $\tilde{I}$. This conversion simplifies the segmentation procedure as the contour is now represented by a path that divides the polar image vertically; this is effectively demonstrated in [87, 96]. Figure 4.5 depicts an image of an object with its estimated centre point alongside the image warped to a polar representation.



Figure 4.5: Example of polar image representation for fast level sets. (a) Image of object with estimated centre identified and (b) image warped to a polar representation.

The fast level sets algorithm works to quickly and iteratively divide each pixel in the polar image into four categories. The function $\Phi(u, v)$ indicates whether each pixel, (u, v) , is part of the inner boundary, l_{in} , the outer boundary, l_{out} , the object itself or the background. The format for the function Φ is:

$$\Phi(u, v) = \begin{cases} -1 & \text{if } (u, v) \in l_{in} \\ 1 & \text{if } (u, v) \in l_{out} \\ -3 & \text{if } (u, v) \in \text{object (left of } l_{in}) \\ 3 & \text{if } (u, v) \in \text{background (right of } l_{out}) \end{cases} \quad (4.16)$$

The algorithm starts by initializing Φ such that the leftmost n columns of the polar image representation are set to be associated with the object as they represent the pixels closest to the object's estimated centre followed by the inner boundary, the outer boundary and the remainder of the image is to belong to the background. Using the estimated centre from Section 4.1.2.1 removes the need for an operator to manually select an initial contour.

For the algorithm to correctly sort the pixels between the regions, a cost function, $F_{ext}(u, v)$, is defined. The cost function represents the likelihood that each pixel belongs to the object or the background. Positive values of the cost function represent pixels that are more likely to belong to the object, and negative values represent pixels that are more likely to belong to the background. The cost function can be constructed using edge-based techniques or region-based techniques to segment an object from the background. The traditional cost function is defined as:

$$F_{ext}(u, v) = -\lambda_1(\tilde{I}(u, v) - c_l)^2 + \lambda_2(\tilde{I}(u, v) - c_r)^2 \quad (4.17)$$

Where $\tilde{I}(u, v)$ is the intensity of pixel, (u, v) , in the image $\tilde{I}$, and c_l and c_r are the mean intensity values of the pixels on each side of the current object boundary defined by Φ .

$$c_l = \frac{\sum_{(u,v)} \tilde{I}(u,v) \mathbf{1}_{\Phi(u,v) < 0}}{\sum_{(u,v)} \mathbf{1}_{\Phi(u,v) < 0}} \quad (4.18)$$

$$c_r = \frac{\sum_{(u,v)} \tilde{I}(u,v) \mathbf{1}_{\Phi(u,v) > 0}}{\sum_{(u,v)} \mathbf{1}_{\Phi(u,v) > 0}} \quad (4.19)$$

The parameters λ_1 and λ_2 dictate whether the segmentation prioritizes moving the boundary outwards (right) or inwards (left).

After initialization, the cost function is calculated for each pixel in the image. The algorithm proceeds by examining all elements of Φ that belong to the outer boundary and switching them to the inner boundary if the cost function at that pixel is greater than zero. A cost function value greater than zero represents an element that is more likely to belong to the object itself rather than to the background. Thus the boundary contour is moved outwards to enclose the pixel in question. For any pixel that is moved from the inner boundary to the outer boundary, the neighbouring pixels must be adjusted as well. Neighbour pixels are defined as the pixels directly above, below, left and right of the pixel being assessed. If any of the neighbouring pixels currently belong to the background, they are switched to the outer boundary. Once this process is complete for all elements of Φ that have a cost function greater than zero, a cleaning process is completed. This involves going through all the pixels in the inner boundary and switching them to belong to the object itself if all neighbouring pixels have $\Phi < 0$, meaning they all either belong to the object or the inner boundary. Following this procedure ensures that the structure of Φ remains consistent even as the boundaries around the object change.

The opposite procedure is then performed to move any pixels from the inner boundary to the outer boundary if the pixel's cost function is less than zero. The same procedure as above is performed to move pixels that are more likely to belong to the background from the outer boundary to the inner boundary and adjust all neighbouring pixels. The two procedures, labelled switch-in and switch-out, are illustrated in Figure 4.6 and explained further in Algorithms 2 and 3.

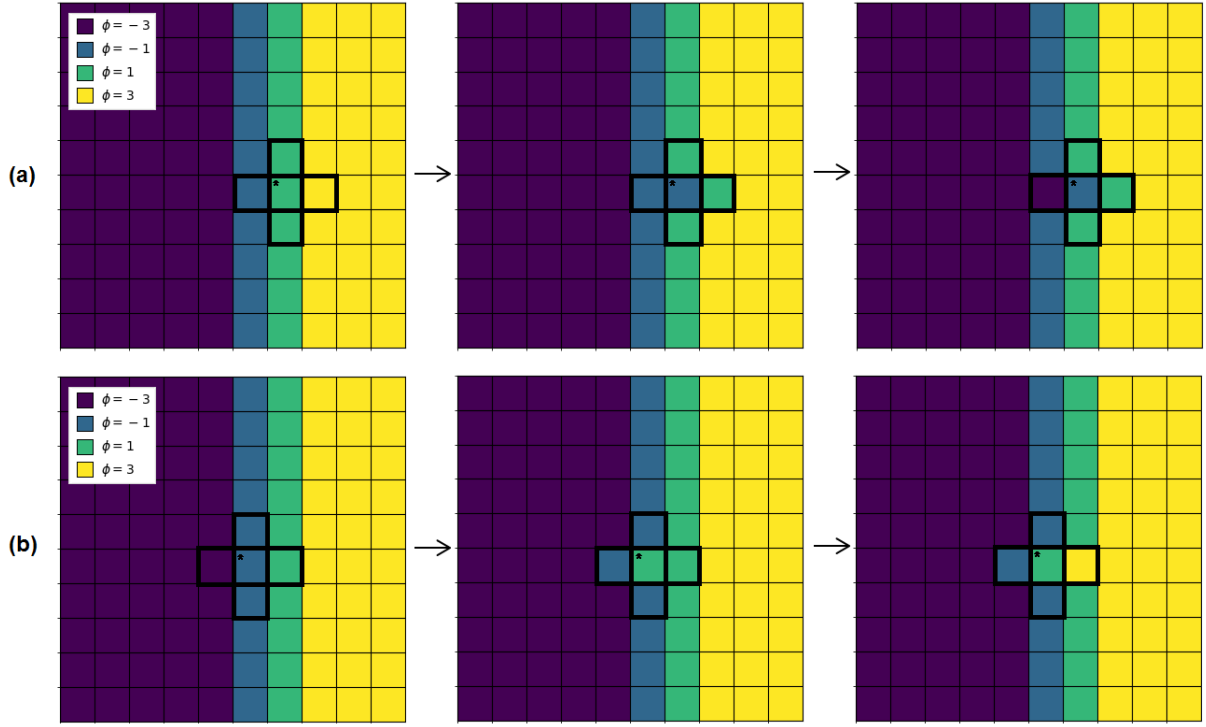


Figure 4.6: Switch-in (a) and switch-out (b) algorithms illustrated in three stages. Left is the original boundaries, centre is after the boundary pixel is swapped and right is after neighbouring pixels have been modified.

Algorithm 2: Switch-in algorithm

Input: F - cost function calculated at each pixel Φ - current pixel labels**Output:** Φ - updated pixel labels

```
1 foreach pixel  $(u, v)$  such that  $\Phi(u, v) = 1$  and  $F_{ext}(u, v) > 0$  do
2    $\Phi(u, v) = -1$ 
3   if Neighbours( $\Phi(u, v)$ ) = 3 then
4     Neighbours( $\Phi(u, v)$ ) = 1
5   end
6 end
7 foreach pixel  $(u, v)$  such that  $\Phi(u, v) = -1$  do
8   if AllNeighbours( $\Phi(u, v)$ ) < 0 then
9      $\Phi(u, v) = -3$ 
10  end
11 end
```

Algorithm 3: Switch-out algorithm

Input: F - cost function calculated at each pixel Φ - current pixel labels**Output:** Φ - updated pixel labels

```
1 foreach pixel  $(u, v)$  such that  $\Phi(u, v) = -1$  and  $F_{ext}(u, v) < 0$  do
2    $\Phi(u, v) = 1$ 
3   if Neighbours( $\Phi(u, v)$ ) = -3 then
4     Neighbours( $\Phi(u, v)$ ) = -1
5   end
6 end
7 foreach pixel  $(u, v)$  such that  $\Phi(u, v) = 1$  do
8   if AllNeighbours( $\Phi(u, v)$ ) > 0 then
9      $\Phi(u, v) = 3$ 
10  end
11 end
```

In addition to segmenting based on the image intensity, a smoothing cost function is also defined to ensure that the boundary contours are smooth and resemble real-life object edges. The internal cost function is defined as the convolution between the pixel labels, Φ , and a bilateral Gaussian filter, G_σ .

$$F_{in} = \Phi * G_\sigma \quad (4.20)$$

The internal cost function values are used to switch the pixels between boundaries,

following the procedures described in Algorithms 2 and 3. The result is that pixel labels are switched to the labels of the pixels that surround them. This stops very sharp edges, as well as small regions from being segmented away from the remainder of the image.

The external and internal cost functions are iterated over several times until F_{ext} and F_{in} are zero at all boundary pixels. This indicates the object has been optimally segmented given the cost functions defined by Equations 4.17 and 4.20. Alternatively, the algorithm may stop when a maximum number of iterations has been completed.

4.1.2.3 Boundary Probability Augmented Fast Level Sets

In this work, a combination of region-based techniques based on multi-dimensional colour image histogram data, as well as edge-based techniques based on the filtered boundary probability, W (Equation 4.15), are used to define a novel external cost function for the fast level sets algorithm. This original cost function uses multiple terms to leverage both the colour and depth data from the RGB-D image to improve segmentation.

$$F_{ext}(u, v) = -(P_r(u, v) + \gamma E_l(u)) + (P_l(u, v) + \gamma E_r(u)) \quad (4.21)$$

P_r and P_l are the probabilities that a pixel belongs to the right and left side of the segmentation boundary based on the colour image, $\tilde{I}$. The right and left sides of the boundary are defined by Φ in the same manner as the traditional fast level sets method (Equation 4.16). E_r and E_l represent the impacts of the edges calculated from the filtered boundary probability, W , to move the segmentation contour to the right or to the left, respectively. The tuning variable, γ , adjusts the impact that the colour data and the depth data contribute to the cost function. Each term of the cost function is explained in more detail below.

For segmentation, the image, $\tilde{I}$, is expressed using the HSV colour space. This representation is chosen to handle uneven lighting conditions in the image, as is commonly done in image processing applications. Using a multi-channel histogram with HSV images allows the hue, saturation and value (brightness) to be used to differentiate the object from the background. This provides additional information over other techniques that rely on the grayscale image [9] or only a single colour image channel [86, 96]. Another improvement is the use of the entire probability distribution of the colour image rather than only comparing pixel values to their mean, allowing objects that contain various colours to be segmented properly. After converting the image, $\tilde{I}$, to the new colour space, the histogram terms for each side of the current object boundary are defined as:

$$H_l = \text{hist}(\tilde{I}(u, v)) \quad \forall (u, v) \text{ such that } \Phi(u, v) < 0 \quad (4.22)$$

$$H_r = \text{hist}(\tilde{I}(u, v)) \quad \forall (u, v) \text{ such that } \Phi(u, v) > 0 \quad (4.23)$$

The histograms are then used to generate the probability terms, P_l and P_r , by back-projecting the image against each histogram.

The edge terms are calculated using the filtered boundary probability, $\tilde{W}$ (Equation 4.15), which is also warped to the polar coordinate representation using the estimated object centre to generate $\tilde{W}$. The cost function edge terms are defined by summing along each row of $\tilde{W}$ on the left and right sides of the object boundary.

$$E_l(u) = \sum_v \tilde{W}(u, v) \quad \forall(u, v) \text{ such that } \Phi(u, v) < 0 \quad (4.24)$$

$$E_r(u) = \sum_v \tilde{W}(u, v) \quad \forall(u, v) \text{ such that } \Phi(u, v) > 0 \quad (4.25)$$

Large values of E_l for a given row u will bias the cost function (Equation 4.21) to move the object boundary towards the left, where there are likely object boundaries defined in the polar boundary probability map, $\tilde{W}$. Conversely, large values of E_r will bias the cost function to move the object boundary to the right. This is depicted graphically in Figure 4.7.

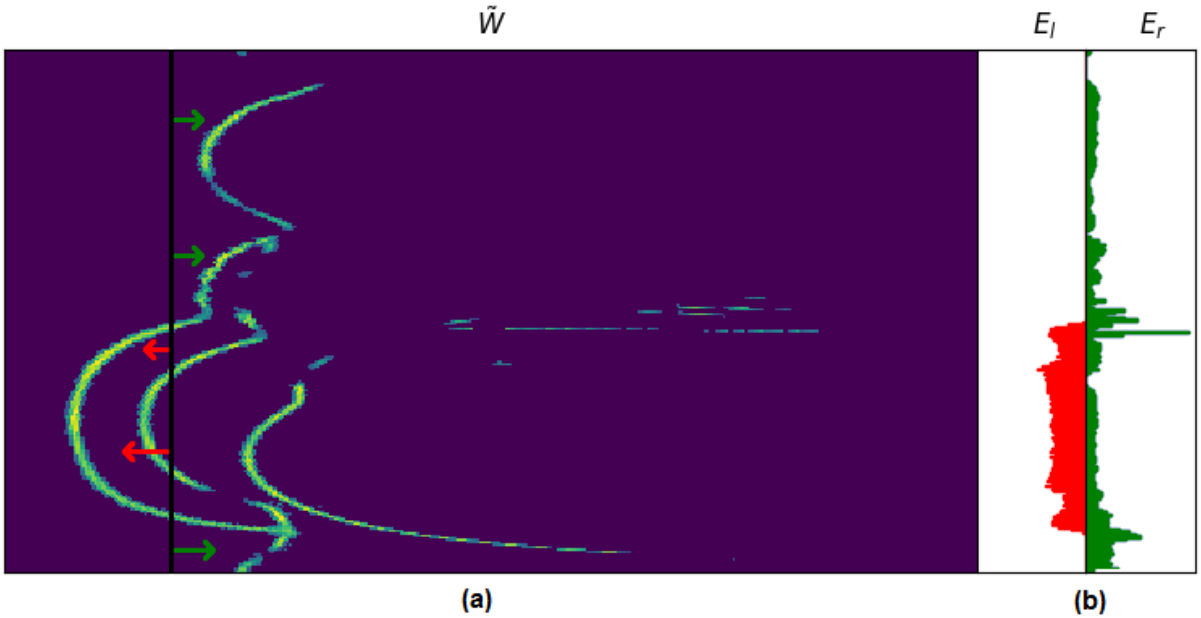


Figure 4.7: (a) Polar filtered boundary probability, $\tilde{W}$, with an initial contour identified (black vertical line) as well some illustrative examples of the net impact of the edge terms, E_l and E_r , on the contour. (b) Edge terms E_l and E_r for each row of the polar filtered boundary probability image.

The novel cost function (Equation 4.21) is used in place of the traditional cost function (Equation 4.17) to segment the object from the boundary. The rest of the segmentation procedure is the same as the traditional level sets algorithm, as described in Section 4.1.2.2.

Algorithm 4 describes the novel boundary probability augmented fast level sets segmentation algorithm. The variables N_1 , N_2 , and N_T represent the number of iterations of

the external cost function, the number of iterations of the internal cost function and the total number of iterations of both stages, respectively.

Algorithm 4: Boundary probability augmented fast level sets segmentation algorithm

Input:

$\tilde{I}$ - polar colour image to be segmented

$\tilde{W}$ - polar filtered boundary probability of the image

N_1, N_2, N_T - number of iterations for external cost function, internal cost function and total iterations for both stages

Output:

Φ - segmentation pixel labels

```

1 Initialize  $\Phi, F_{ext}, F_{in} = 0$  and  $G_\sigma$ 
2 for  $n_0 < N_T$  do
3   for  $n_1 < N_1$  do
4     Calculate  $P_r, P_l, E_r, E_l$  as in Equations 4.22-4.25
5      $F_{ext} = -(P_r + \gamma E_l) + (P_l + \gamma E_r)$ 
6      $\Phi = \text{SwitchIn}(\Phi, F_{ext})$ 
7      $\Phi = \text{SwitchOut}(\Phi, F_{ext})$ 
8   end
9   for  $n < N_2$  do
10     $F_{in} = \Phi * G_\sigma$ 
11     $\Phi = \text{SwitchIn}(\Phi, F_{in})$ 
12     $\Phi = \text{SwitchOut}(\Phi, F_{in})$ 
13  end
14 end

```

Once the algorithm is complete, the segmentation contour described by the function Φ is used to generate a mask to isolate the object from the background. This mask can be applied to the colour and depth images as well as the associated point cloud.

Figure 4.8 shows the segmentation results of an object in the robot workspace with varying values of γ . Equation 4.21 requires choosing a value for γ to vary the weight of the colour and depth effects on the cost function. If the value of γ is zero, the fast level sets algorithm only uses information from the colour image to segment the object. As the value of γ is increased, the impact of the boundary edges on segmentation will increase.

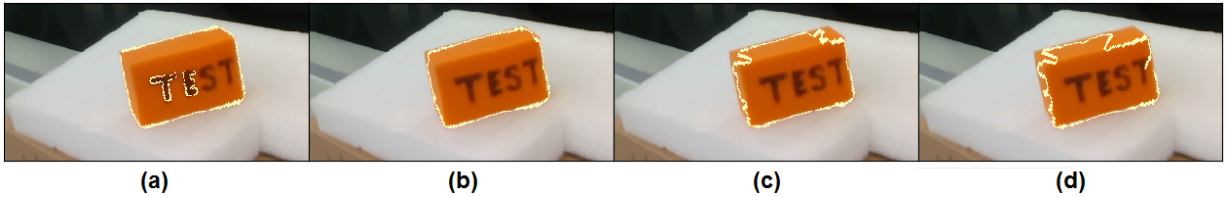


Figure 4.8: Object segmentation using fast level sets with different values of γ . (a) $\gamma = 0$, (b) $\gamma = 0.25$, (c) $\gamma = 0.5$, and (d) $\gamma = 1$.

For the case that relies only on colour ($\gamma = 0$), it can be seen that the text on the object is segmented into the background. It is determined to be more likely to belong to the background than the object itself based on the multi-dimensional colour histograms calculated as the contour evolves. Interestingly only two of the letters written on the object are segmented as part of the background, whereas the other two are still segmented with the object. This may be due to a slight change in colour due to illumination as well as the smoothing component, F_{in} , of the fast level sets algorithm. As the value of γ is increased, the contour moves towards the edges and generates a better segmentation of the object. Particularly good results are achieved with $\gamma = 0.25$. Once the value of γ increases beyond an optimal value, there is an overweighting towards the edges identified in W , see Figure 4.4. This is seen in an extreme case with $\gamma = 1$ where the contour is overfit to the filtered probability boundary and does not do as good of a job segmenting the object.

A variety of other objects were tested using a fixed value of $\gamma = 0.25$, and the results can be seen in Figure 4.9. This value of γ was determined empirically and generated good results for the experiments in this research. In scenarios where the object's colour and texture vary significantly or cases with low contrast between the object and its background the value of γ may need to be adjusted. The results show proper segmentation of the objects, but there are few cases where the algorithm has errors in segmenting the objects completely. In Figure 4.9b it can be seen that the white and black portion of the ball is segmented to the background rather than to the object itself. Even though the algorithm is using colour and edge information, there is a region inside the object that is more similar to the background, and it is not overcome by the edge effects. Conversely, Figure 4.9e has a small section of the object that is not segmented because of very strong edge effects that cause a small region with high colour similarity to be segmented to the background rather than the object. While there are some small errors, the overall segmentation is quite strong, and the errors that do exist will not have a major impact on 3D object completion.

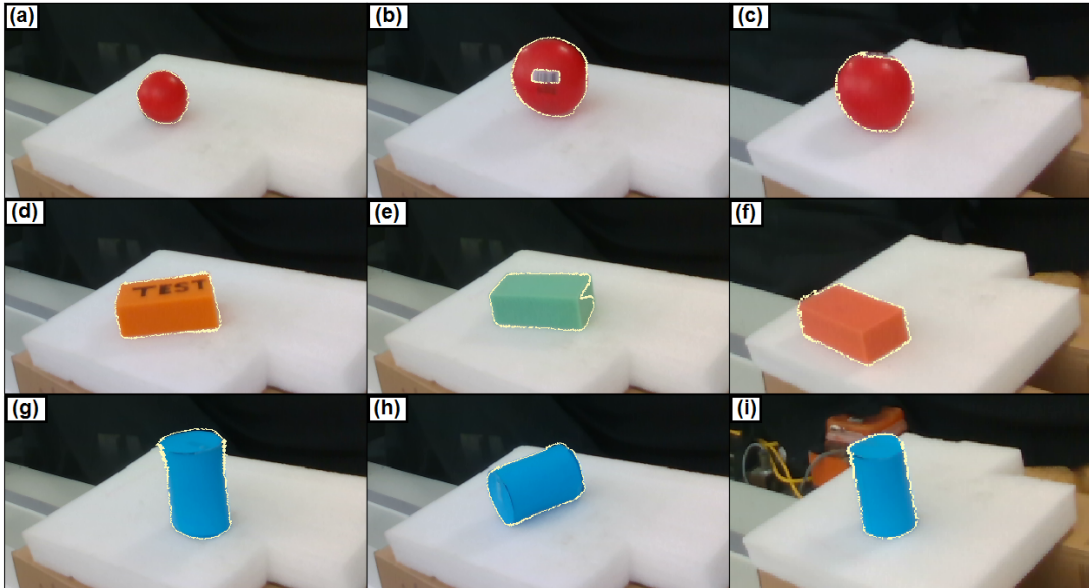


Figure 4.9: Segmentation results for boundary probability augmented fast level sets.

4.1.3 Object Completion

The segmentation mask is used to isolate the object from the background. As the object is only imaged from a single point-of-view, the segmentation mask is considered to only be 2.5D. Generating complete 3D estimates from 2.5D data is an essential step for the framework presented in this thesis. Without a complete 3D estimate of the object, it is impossible to plan grasps or control strategies to reshape the object.

Many techniques for estimating a 3D shape from single-view depth data were presented in Section 2.2. This research uses an algorithm to fit the point cloud generated from the segmentation mask to a variety of primitive shapes and select the shape that has the best fit. The shapes that are considered in this research are spheres, cylinders and boxes, presented in Section 3.3. Additional primitive shapes could easily be added to the algorithm if they are present in the test cases. If more complex objects are used, another algorithm from those discussed in Section 2.2 may be more appropriate, or a new technique altogether may need to be derived.

The object completion technique developed in this thesis is inspired by Schnabel *et al.* [26], who use the RANSAC method to fit planes, spheres, cylinders, cones and tori to unknown point cloud data. The main difference between the work by Schnabel *et al.* and this work is that when fitting spheres and cylinders, the algorithm uses a least-squares fit instead of the RANSAC procedure. The reason for this is because the point cloud from the segmentation mask represents only the object itself, whereas the point cloud in [26] contains a variety of unknown data.

The object completion algorithm presented in this thesis does not represent significant original contributions but is nonetheless a necessary step in the robotic grasping system.

The segmentation contour is used as a mask to generate a point cloud representation of the object from the depth image. The point cloud defines the coordinates in the camera coordinate system and is represented by a set of points $(x_{cam}, y_{cam}, z_{cam})$. Before fitting primitive shapes, outliers are removed by identifying any point that has an x_{cam} , y_{cam} , or z_{cam} value that is more than three standard deviations from the mean of the points enclosed by the contour. This removes points that the contour may contain but are not part of the object to be manipulated.

The point cloud must then be transformed into the robot coordinate system to plan and execute the grasping strategy. The transformation is done using the coordinate transformation calculated in Section 4.1.1.

$$\begin{bmatrix} x_{rob} \\ y_{rob} \\ z_{rob} \\ 1 \end{bmatrix} = T_{cam/rob} \begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \\ 1 \end{bmatrix} \quad (4.26)$$

The coordinates x_{rob} , y_{rob} , z_{rob} represent the point cloud in the CRS F3 robot coordinate system.

For fitting spheres and cylinders, a least-squares fit is performed to find the parameters that best fit the equation for the 3D primitive shape. Starting with a sphere, the equation to fit is:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2 \quad (4.27)$$

This equation can be expanded out and expressed in matrix notation in order to formulate it as a least-squares problem. It is then possible to solve for the radius of the sphere, r , and centre of the sphere, (x_0, y_0, z_0) , that best fits the data.

A similar approach can be taken for fitting cylinders, but the orientation of the cylinder must also be considered. A vertical cylinder is parameterized with the centre of the cylinder on the xy -plane (x_0, y_0) , the angle rotated about the x -axis (β), the angle rotated about the y -axis (α), and the radius (r). This model can be expressed as:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \\ z \end{bmatrix} \quad (4.28)$$

The x' and y' components of the cylinder can be squared and summed to give the radius of the cylinder, r . This leads to an equation that can be formulated as a least-squares problem solving for x_0 , y_0 , β , α and r . Lastly, because this equation is for an infinite cylinder, an estimated height is also calculated by identifying the maximum and minimum values along the z' -axis.

When the rotation about the z -axis is significant, a vertical cylinder does not represent the data well, and a horizontal cylinder must be used. A horizontal cylinder is parameterized in the same way as the vertical cylinder, but it considers both the rotation about the z -axis (γ) and the rotation about the x -axis (β). The rest of the parameterization procedure remains the same as the horizontal cylinder. The data is fit to both the vertical and horizontal cylinder equations to find the best representation for the point cloud data.

A slightly different approach must be taken to fit a box to the data as there are no equations that parameterize the box for least-squares fitting. Given the point cloud data, a plane is fit using the RANSAC technique. The equation for a plane used in the RANSAC algorithm is:

$$Ax + By + Cz = D \quad (4.29)$$

This plane represents one of the faces of the box, determined by the normal of the plane. A second plane is fit to the outliers of the first RANSAC fit. If the second plane is perpendicular to the first plane, then it is determined to be an adjacent face of the box. Given the two faces of the box, the centre of the box is estimated using the mean of the points on each face. The orientation of the box is determined by analyzing the normals of the estimated planes. Lastly, similar to the cylinder, the length, width and height of the box are determined by looking for the maximum and minimum values of the oriented planes.

Once the estimates for each shape are complete, the Mean Square Error (MSE) of each fit is calculated to determine which primitive shape best represents the data. The MSE is calculated by summing the squared distance from each point in the point cloud to the surface of the best fit shape for each class of shapes. The shape with the lowest MSE is used as the 3D estimate for the undeformed object. This object estimate is used as the initial object in the grasp planning and manipulation algorithms. Throughout the remainder of the thesis, the estimated object from 3D shape completion is referred to as the initial undeformed object.

Continuing the example started in Figure 4.3a, the point cloud was fit to a sphere, cylinder and box following the techniques described above. The results of each fit are presented in Table 4.2, and the coloured point cloud of the object alongside the estimated initial undeformed 3D shape (shown as blue dots) for all cases is shown in Figure 4.10. As evidenced by the MSE and the visual representations, it is clear that the box is the best fit for the point cloud data.

Table 4.2: Mean square error for least-squares fit of the segmented point cloud to primitive shapes.

Shape	Mean Square Error [mm^2]
Box	2.72
Sphere	14.18
Cylinder	41.85

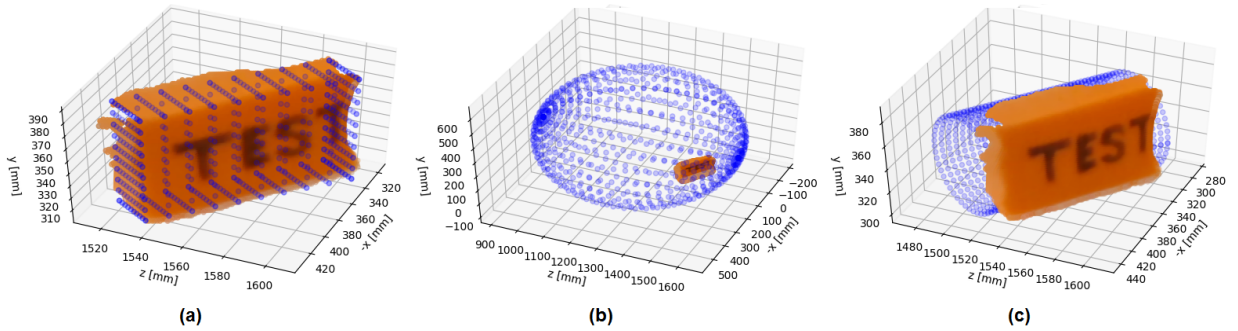


Figure 4.10: 3D object completion results (blue dots) for fitting a (a) box, (b) sphere and (c) cylinder.

The estimates are done using either a least-squares fit or by following the RANSAC procedure, so there are no constraints on the optimal parameters of the primitive shapes. The object completion approach is therefore insensitive to the physical size of the object. This approach also supports a more comprehensive estimation of an object's physical dimensions if only partial data is supplied to the algorithm. In Figure 4.10b, it is visually evident that the sphere fitting for the rectangular sponge would generate a shape too large for manipulation. In this work, the issue of objects being too large for manipulation is handled by the grasp planning algorithm presented in Section 4.2.1. Alternatively, if only

a certain size of objects is desired, a constrained optimization could be performed, or an object recognition-based completion method could be utilized.

Examining Figure 4.10a further, the box was fit with parameters that visually match the measured object quite closely. The estimated length, width and height of the object are 133 mm, 37 mm, and 80 mm compared to the true measurements of 124 mm, 44 mm and 75 mm. The estimated coordinates of the centre of the object, $(x, y, z) = (-379 \text{ mm}, 345 \text{ mm}, 1563 \text{ mm})$ was not compared to the true position, but is a good estimate based on the visual information in Figure 4.10a and is further validated by the grasping positions calculated in subsequent sections.

Sources of error in the fit are mostly due to inaccuracy in the depth data. The 3D camera has an error rate proportional to the sensed object’s distance from the camera. In this case, the object is positioned 675 mm from the camera, knowing the depth sensor accuracy is less than 2% [105] on depth measurements at this distance, the error may be as much as 13.5 mm. The average error of the estimated dimensions is 7 mm, which is within the depth sensor’s expected accuracy.

Object completions for the additional results shown in Figure 4.9 are presented in Figure 4.11. In general, the 3D estimates are accurate representations of the shape and dimensions of the true object. There are some errors in the dimensions, in particular for the box-shaped objects. These errors are a result of the accuracy of the acquired depth data rather than the fitting algorithm itself. There is also a fitting error in Figure 4.11i where a cylinder-shaped object is incorrectly fit like a box. The cylinder fitting algorithm fits infinite cylinder using a least-squares algorithm. Here, it is adjusted to fit cylinders of a finite height, and the algorithm works well in most cases (Figures 4.11g and 4.11h) but is still biased by the faces of the cylinder. This bias may be what causes the incorrect fit for the third cylinder example. This shortfall of the cylinder fitting algorithm is an area that could be improved for more accurate fitting results. Table 4.3 describes the results shown in Figure 4.11.

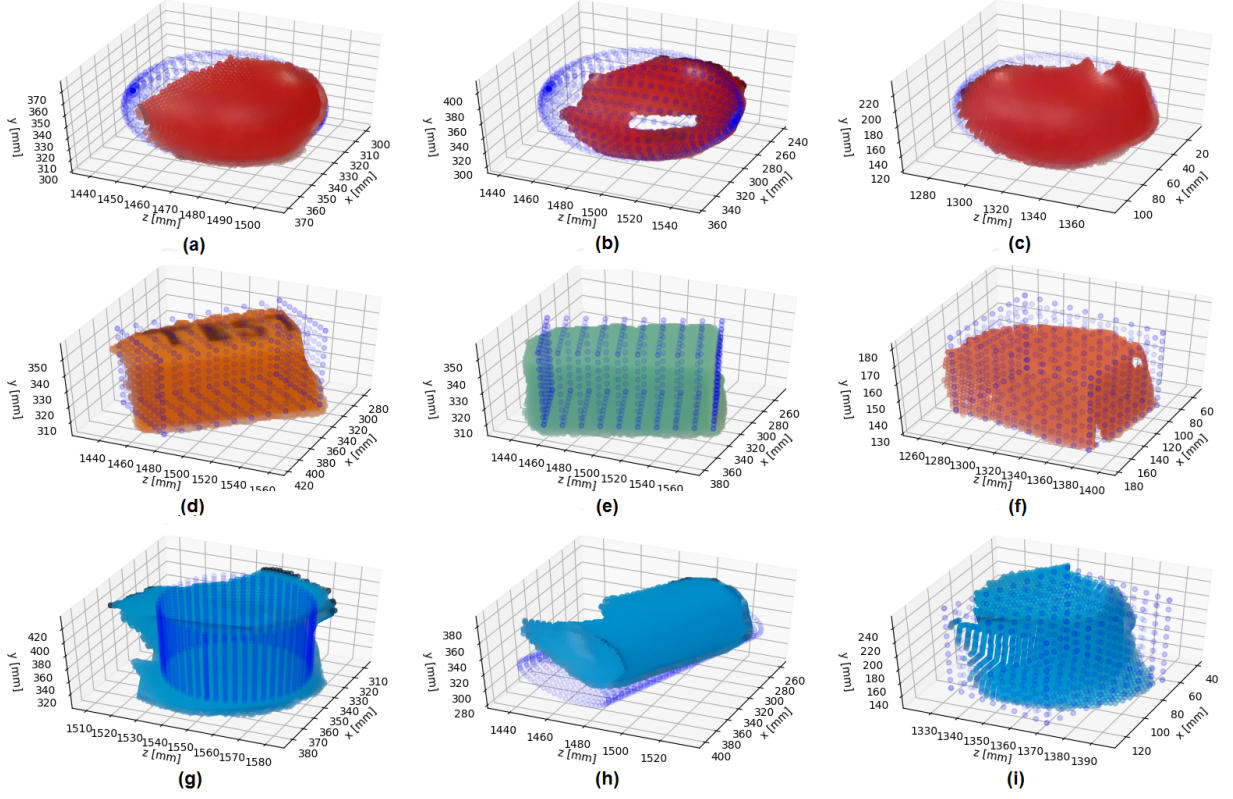


Figure 4.11: 3D object completion results using primitive shape fitting.

Table 4.3: Numerical results for object completions shown in Figure 4.11.

Label	Shape	Best Fit Shape	Average Dimension Error
(a)	Sphere	Sphere	0.6 mm
(b)	Sphere	Sphere	1.2 mm
(c)	Sphere	Sphere	3.2 mm
(d)	Box	Box	9.0 mm
(e)	Box	Box	10.7 mm
(f)	Box	Box	8.0 mm
(g)	Cylinder	Cylinder	8.0 mm
(h)	Cylinder	Cylinder	7.0 mm
(i)	Cylinder	Box	—

4.2 Planning

4.2.1 Grasp Selection

Initial grasping points for the robotic hand need to be selected in order to grasp and deform a 3D object. These points determine where the hand makes initial contact with

the deformable object and define where the object’s deformations occur. For the three-fingered Barrett Hand, the contact points define the three locations that the fingertips will contact the object to be deformed. The initial contact points are determined based on the initial undeformed 3D object obtained from shape completion, as detailed in Section 4.1.3, and a predefined desired 3D shape which the object should converge towards as it reshapes under the external forces of the fingertips. The contact points are selected to optimally deform the initial shape towards the desired shape, under the constraints of maintaining a stable grasp, obeying the hand’s kinematic constraints, and avoiding any obstacles in the workspace.

The problem of selecting contact points on a 3D object to control the deformations to a desired 3D shape has been almost entirely unexplored. The majority of related work uses predefined models to represent objects and works in simulation to design and test algorithms [75, 78]. Alternatively, many works only consider deformation in one or two dimensions rather than three-dimensional deformations [73].

Zaidi *et al.* [90] develop a grasping and manipulation pipeline that selects contact points for stable grasps on deformable objects through simulation and applies them to a real robot manipulator and three-fingered gripper. The work generates stable grasps using a method that ensures contact equilibrium at the interface between the fingertips and the 3D deformable object using a mass-spring model for the objects and force feedback from the robotic hand. The work presents a method to generate stable grasps for pick and place tasks using deformable objects but does not consider the task of shape control.

The work that this research is most closely associated with is presented by Nadon and Payeur [86]. Their work uses a three-fingered hand that is fixed to a platform, and grasp points are selected to transform the 2D contour of a 3D object to a desired shape. The work presented in this thesis significantly expands on previous efforts. Grasps and deformations are planned on 3D models of objects rather than on 2D projections and contours. Additionally, a robot manipulator is used to place the hand in 3D space to grasp and deform unseen objects in undefined environments.

4.2.1.1 Grasp Synthesis

To execute a stable grasp, the concepts of form and force closure must be considered. The form closure and force closure properties of robotic grasping can be loosely defined as the ability of the robotic hand to inhibit all motions of the object being manipulated against any externally applied forces. This includes inhibiting all object translations and rotations. The main difference between the two concepts is that force closure includes forces generated by friction at the contact points.

These concepts can be defined formally with a set of equations that balances contact forces and torques against all external forces applied to the object [111]. While this formalism is necessary for certain applications, it is not implemented here. Formal force closure conditions are not considered in this work because the contact and friction forces applied by the hand cannot be accurately measured. Additionally, the traditional force closure approach cannot be easily implemented for deformable object grasping due to the

complex interactions between the object as it deforms and the forces applied by the fingertips of a robotic hand, which was first studied in [88]. Instead, a more relaxed force closure condition is considered for grasp planning. Generally speaking, the forces applied by the fingertips must sum to zero to create a stable grasp. This research only considers one type of grasp using the three fingertips as opposing forces, sometimes referred to as a three-fingered tripod grasp. Many other grasp types exist that use the fingers to hook, support or wrap around an object [112], but they are outside of the scope of this research.

For the considered grasp type, the force closure condition will be met by ensuring that all grasps have a spread angle greater than 90° . The spread angle of the hand, θ_7 , determines the location of the fingers around the palm, seen in Figure 3.7. Requiring the spread to be greater than 90° guarantees that there are forces generated by the fingertips in opposing directions. This requirement does not meet the strict conditions of force closure, but it ensures that contact forces in opposing directions will resist each other, creating generally stable grasps [104].

An additional grasping condition is that all grasps are made with the centre of the object, $\mathbf{c}$, placed along the z -axis of the end-effector, illustrated in Figure 4.12a. Placing the grasp centre at the centre of the object minimizes the effect of inertial and gravitational forces on the grasp [113], limiting the translation and rotation of the object. Additionally, this condition guarantees that the object is grasped and deformed by the defined paths of the fingertips of the hand as they close towards the z -axis, an important factor for choosing the grasp points for shape control in this work.

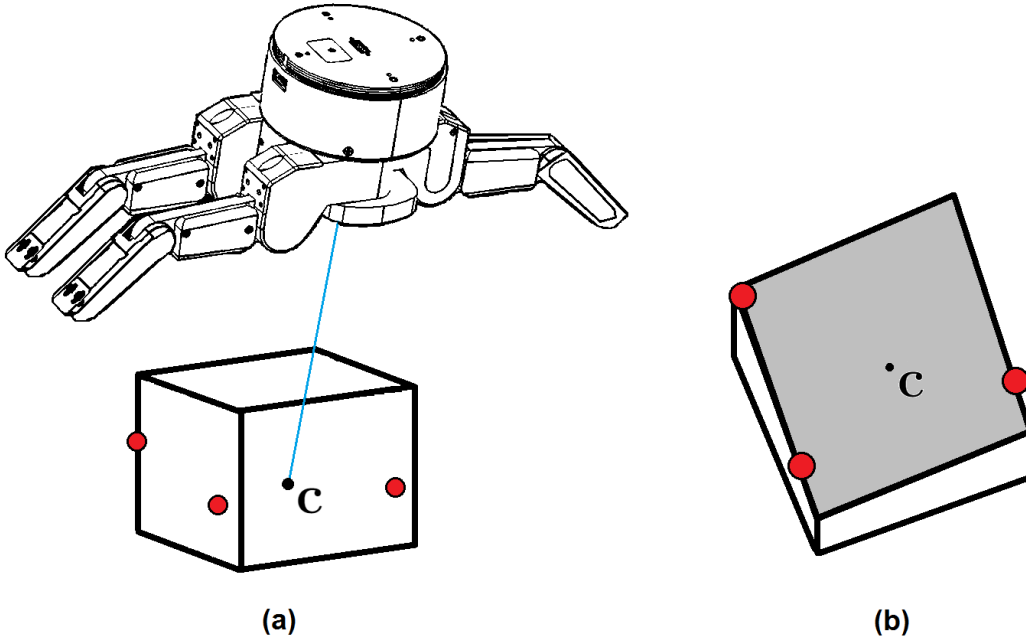


Figure 4.12: Hand configuration for grasping an object with initial contact points identified (red dots). (a) Example of the Barrett Hand in a grasping pose for given desired contact points. (b) Cross section of object to be grasped with desired contact points identified.

Grasp points selection must also take into account the kinematic constraints of the hand, first presented in Section 3.2.3. Although the hand has four DoF, the number of independent fingertip positions is limited because of two major constraints. The condition that the spread is the same for both non-fixed fingers (F1 and F2) limits how the fingers can be distributed around the palm of the hand. The fingertip positions are further limited by the fact that the displacement along z and the displacements along x and y cannot be independently controlled. This limits the control of the displacement along the z -axis of the hand, as it is dependant on the distance from the centre of the palm, first discussed in Section 3.2.3. Given the z -axis of the hand intersects the centre of the object, there is only one possible point of contact for each fingertip given a hand position and spread angle. An illustration of the hand position with potential grasping points is shown in Figure 4.12a. Additionally, a cross-section of the object defined by a plane perpendicular to the z -axis that contains the centre of the object illustrates how the kinematic constraints limit the positions of each fingertip, seen in Figure 4.12b. In practice, the fingertips do not necessarily contact the object at the points defined by the plane because of kinematic constraints that limit the independent control of the z coordinate and the distance from the centre of the grasp along a vector perpendicular to the plane. This perpendicular plane is where the grasp points are planned.

After planning grasp points on the cross-sectional plane, a simplification has been made to use the mean value of the z positions of each fingertip position to place the hand in the grasping position. This simplification will cause inaccuracies in the exact position of the fingertips. Section 4.2.1.3 goes into more detail on how the grasp points are selected and how the limitation imposed by the kinematic constraints of the hand impacts the chosen grasping positions. The amount of error in the fingertip placement is directly related to the distance between each fingertip’s placement and the centre of the object. When the distances are all the same, this represents the grasp points being distributed on a circle centred at the object’s centre, and there will be no error in the placement. As the fingertip placement moves away from the circle, the error in placement increases. In order to ensure a stable grasp, even when there is an error in fingertip placement, the hand’s force sensors are used to ensure contact with the object. This procedure is described in Section 4.3.1.

Other works that use the Barrett Hand have handled the kinematic constraints in other ways to ensure stable grasps on deformable objects. Zaidi *et al.* [90] work within the confines of the kinematic constraints by finding equilateral grasping triangles that satisfy the inverse kinematics of the hand. Each vertex of the triangle represents one fingertip location, with the centre of the triangle placed along the z -axis of the hand. Since the work by Zaidi *et al.* is only concerned with stable grasps rather than shape control, they are not limited to placing the fingertips at points of optimal deformation. The work in this thesis prioritizes shape control, so it does not limit the potential grasp point location at the cost of less accurate fingertip contact points.

Nadon and Payeur [87] work with a hand that is fixed to a tabletop, so the distance from the object must be manually controlled by placing supports underneath the object to ensure the fingertips make contact with the deformable objects. In this work, the hand can be positioned freely using the robot manipulator, so there is no need to manually adjust the height of the objects to be grasped.

4.2.1.2 Object Representation

To plan the initial grasp points, the desired 3D shape which the initial object is to be deformed towards must be supplied to the algorithm. In this thesis, the desired 3D shapes are CAD models that are generated using ANSYS software [114]. ANSYS is an engineering simulation software for product design and testing. Two categories of desired 3D shapes are used in this work. The first category is represented by 3D models generated by applying simulated Finite Element Method (FEM) deformations to an object whose geometry matches the real object being deformed. The objects' material properties (Young's modulus and Poisson's ratio) are assigned to match those of an elastic material. However, they are not intended to match the real objects' material properties. The goal is to represent the desired shape so that the original shape can be deformed towards it in a model-free manner, without knowing or estimating the object's deformation parameters. The other category of objects includes undeformed 3D shapes that do not match the initial object shape. These shapes require large deformations from the initial object. The two categories of objects provide test cases where deforming towards the desired shape is possible and test cases where the hand is severely underactuated to completely achieve the desired deformation. The objects designed in ANSYS are saved as point clouds so they can be used to plan the grasp points outside of the ANSYS software.

The provided shape model is centred at an origin, $(0,0,0)$, meaning the first step in grasp planning is to translate the initial undeformed object's centre to align with the desired objects. This is done by subtracting the mean Cartesian coordinates, $\mathbf{c}$, from the point cloud of the initial undeformed object. Figure 4.13a shows an example of an initial undeformed object obtained from shape completion applied to data collected with an RGB-D sensor, as detailed in Section 4.1.3. Alongside the initial 3D object, Figure 4.13b shows the desired 3D shape toward which the object should deform when external forces are applied by the robotic hand's fingertips. This example's desired shape is a box that is deformed at two locations on the top face.

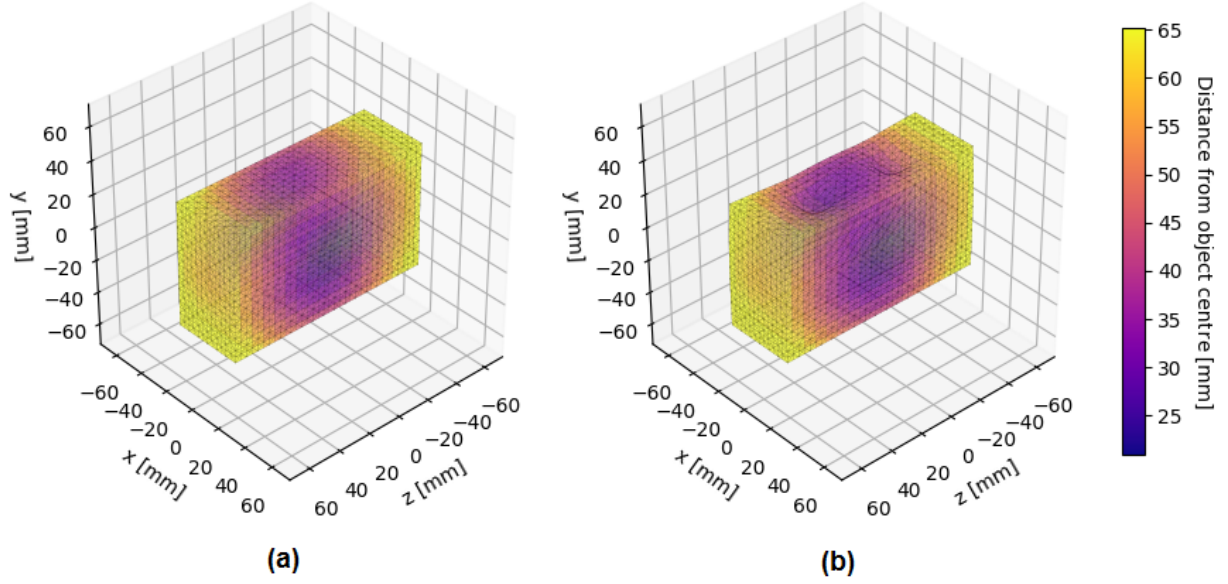


Figure 4.13: Example of (a) an estimated initial undeformed 3D object geometry alongside (b) the desired 3D shape with the distances from the centre of the object to the surface identified.

The surfaces of the initial object and the desired 3D shape are represented using spherical coordinates to assist in selecting initial grasp points. This coordinate system transformation enables grasp point selection for shape control by examining the differences in radius at the surfaces of the initial and desired object shapes.

The new coordinates can be defined by mapping the points from the Cartesian coordinate system in which they are collected to a spherical coordinate system as follows:

$$r = \sqrt{x^2 + y^2 + z^2} \quad (4.30)$$

$$\phi = \text{atan2}(y, x) \quad (4.31)$$

$$\theta = \text{atan2}(\sqrt{x^2 + y^2}, z) \quad (4.32)$$

This follows the standard spherical coordinate convention where r represents the distance from the origin or centre of the object, ϕ is the azimuthal angle, and θ is the elevation angle.

The Cartesian coordinates can be retrieved by performing the inverse calculations as follows:

$$x = r \sin(\theta) \cos(\phi) \quad (4.33)$$

$$y = r \sin(\theta) \sin(\phi) \quad (4.34)$$

$$z = r \cos(\theta) \quad (4.35)$$

For this research, the spherical coordinates are then further transformed into an image, or regularized grid, where the x coordinate is defined by the azimuthal angle on the interval $[-\pi, \pi]$, the y coordinate is defined by the elevation angle on the interval $[0, \pi]$, and the intensity is defined by the distance r from the object centre. This representation is referred to as a surface image, S , as it represents all the points on the object's surface in spherical coordinates. The value of representing the objects in this way is that it reduces the dimensionality of the object and any possible deformations. The shape control of the object can now be classified as a finite DoF problem. Planning the shape control contact points using this object representation is one of the contributions of this thesis.

A resampling technique is performed to project the spherical coordinates onto a grid defined on the intervals that ϕ and θ represent. Resampling is done using bilinear interpolation. The interpolation is constructed by triangulating the convex hull of the input data with the grid. On each triangle, barycentric linear interpolation is performed to give the result for a point on the grid. Given a point on the grid, $\mathbf{Q}$, the radius can be interpolated using the values of the three spherical coordinates that triangulate the point ($\mathbf{V}_1$, $\mathbf{V}_2$ and $\mathbf{V}_3$) and a set of assigned weights (W_{v_1} , W_{v_2} and W_{v_3}).

$$Q_r = \frac{W_{v_1} V_{1_r} + W_{v_2} V_{2_r} + W_{v_3} V_{3_r}}{W_{v_1} + W_{v_2} + W_{v_3}} \quad (4.36)$$

For barycentric interpolation the weights are calculated as follows [115]:

$$W_{v_1} = \frac{(V_{2_\theta} - V_{3_\theta})(Q_\phi - V_{3_\phi}) + (V_{3_\phi} - V_{2_\phi})(Q_\theta - V_{3_\theta})}{(V_{2_\theta} - V_{3_\theta})(V_{1_\phi} - V_{3_\phi}) + (V_{3_\phi} - V_{2_\phi})(V_{1_\theta} - V_{3_\theta})} \quad (4.37)$$

$$W_{v_2} = \frac{(V_{3_\theta} - V_{1_\theta})(Q_\phi - V_{3_\phi}) + (V_{1_\phi} - V_{3_\pi})(Q_\theta - V_{3_\theta})}{(V_{2_\theta} - V_{3_\theta})(V_{1_\phi} - V_{3_\phi}) + (V_{3_\phi} - V_{2_\pi})(V_{1_\theta} - V_{3_\theta})} \quad (4.38)$$

$$W_{v_3} = 1 - W_{v_1} - W_{v_2} \quad (4.39)$$

Figure 4.14 illustrates the transformation of an object from 3D Cartesian space to spherical coordinates and then to a surface image.

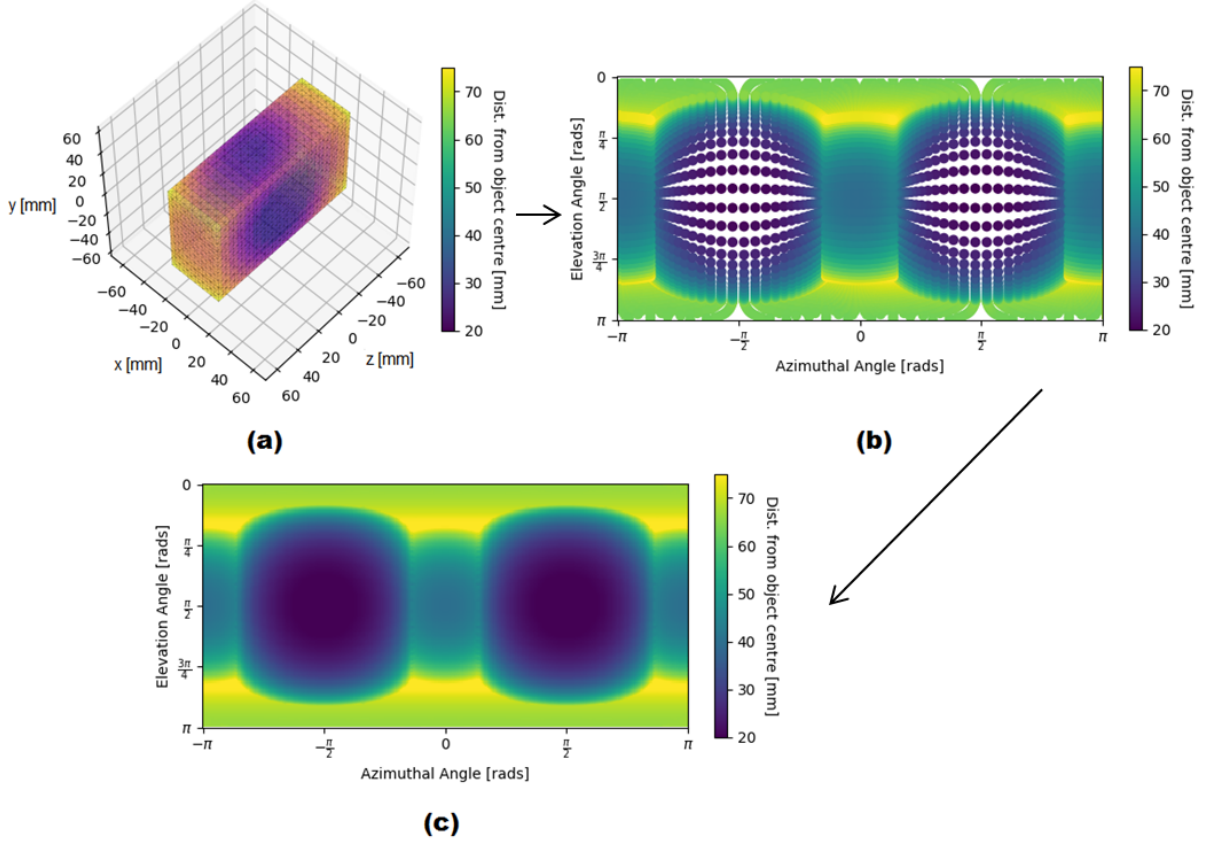


Figure 4.14: Transformations from 3D Cartesian space to a surface image representation. (a) Object in 3D Cartesian space. (b) Object represented in spherical coordinates, azimuthal angle and elevation angle on the axes and radius as the marker colour. (c) Surface image interpolated from the spherical coordinate representation.

After calculating the surface image for the initial undeformed object, S_i , and the desired 3D shape, S_f , a difference surface image, S_{diff} is obtained by subtracting the desired surface image from the initial surface image.

$$S_{diff} = S_i - S_f \quad (4.40)$$

Representing the differences between the objects' surfaces in this way allows traditional image processing techniques to be used to plan the initial grasp points and deformations. The difference surface represents the difference in radius between the original object and the desired shape at a finite number of points represented using spherical coordinates. The difference can be used to determine the amount of deformation required to control the object to its desired shape at each point on the surface. To completely deform the initial undeformed object to its desired shape, each point on the difference surface image would have to be manipulated towards or away from the centre of the object by the amount specified by the pixel intensity value. Performing deformation with the robotic hand does not enable moving the surface away from the object's centre. Therefore, only the positive values of S_{diff} need to be considered when selecting the candidate contact points.

$$S_{diff} = S_{diff} \cdot \mathbf{1}_{S_{diff} > 0} \quad (4.41)$$

Figure 4.15 shows the surface images of the initial undeformed object and desired shape presented in Figure 4.13, and the difference surface image between them.

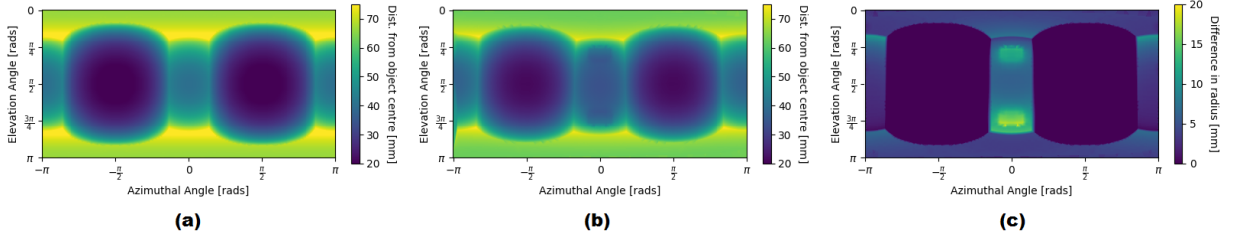


Figure 4.15: (a) Surface image for the initial undeformed 3D object. (b) Surface image for the desired 3D shape. (c) Difference surface image between the initial undeformed object and desired shape.

Because the hand is free to grasp the object from any orientation, an alignment between the initial object and the desired shape is performed. The alignment is performed by calculating the total error of the difference surface image for all possible orientations of the desired shape and selecting the orientation that minimizes the initial error. The rotated desired shape is used to generate the difference surface image and plan the initial contact points.

The major limitation of using this spherical coordinate transformation is that this solution is not viable for objects that contain multiple points along the same vector from the object's origin because there is only a single mapping in the spherical coordinate image. This can be thought of mathematically as surfaces with a unique radius for all azimuthal angle (ϕ) and elevation angle (θ) pairs, described by the function $f(\phi, \theta) = r$. The 2D analog would be contours that uniquely solve the function $r = f(\theta)$, a 2D example of valid and invalid contours are shown in Figure 4.16. Therefore in this work, all 3D objects must adhere to this limitation.

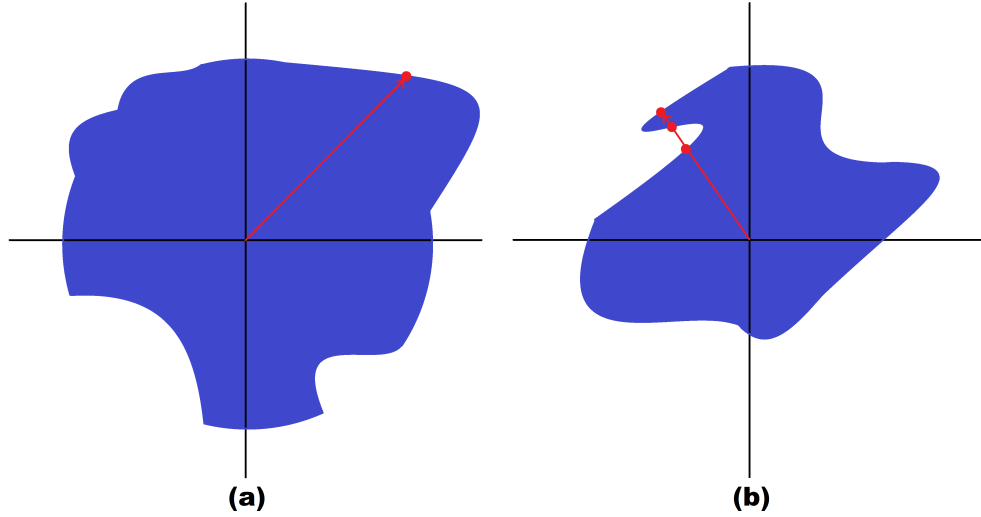


Figure 4.16: Examples of 2D contours that have valid one-to-one mappings (a) and invalid one-to-one mappings (b).

Representing the deformable objects in this way generates a finite number of control points on the object that need to be deformed to achieve the desired 3D shape. This allows a complete solution to be developed. However, the problem is still severely underactuated as the hand only has three fingers and is limited by its kinematic constraints discussed in Section 4.2.1.1. Therefore, it is only possible to deform the object towards its centre at three distinct but not independently chosen points on the object’s surface.

4.2.1.3 Grasp Points Selection

The grasp constraints discussed in Section 4.2.1.1 and the object representation described in Section 4.2.1.2 lead to the development of an original grasp points selection algorithm for controlling the shape of a deformable object towards a desired 3D shape.

To choose the initial grasping points, it inherently makes sense to choose locations where the difference surface image has maximal values, to deform surface locations that represent the largest difference between the undeformed and desired 3D shapes. Due to the fact that the gripper can only control points towards the centre of the object from the initial position, the initial grasp points are chosen at the points of the difference surface image, S_{diff} , that represent local maxima. This concept is an extension of the 2D grasp selection concept presented by Nadon and Payeur [87]. By choosing points that represent local maxima, the object’s surface will deform to bring the points that are furthest from the desired locations to their desired final positions. The well-established concept of diminishing rigidity [116] dictates that the points on the surface of the object closest to the control point will deform most, while points further from the control point will experience less deformation. The theory applies broadly to all deformable objects and can be used without precise deformable object models and used for objects of varying unknown material properties.

A maximum filter is applied to the difference surface image to find all local maxima in the difference surface image. The maximum filter replaces each pixel with the maximum value of the pixels in its local neighbourhood. The local neighbourhood is a $k_s \times k_s$ square centred at the pixel under evaluation, where k_s represents the kernel size of the filter. Local maxima are identified at locations where the difference surface image is equal to the filtered image. A merging technique is used to avoid finding multiple local maxima when local neighbourhoods contain multiple points that are equal to the maximum. If more than one local maxima in a neighbourhood are found, the mean location of all max values is selected. Finding the local maxima on a difference surface image is illustrated in Figure 4.17.

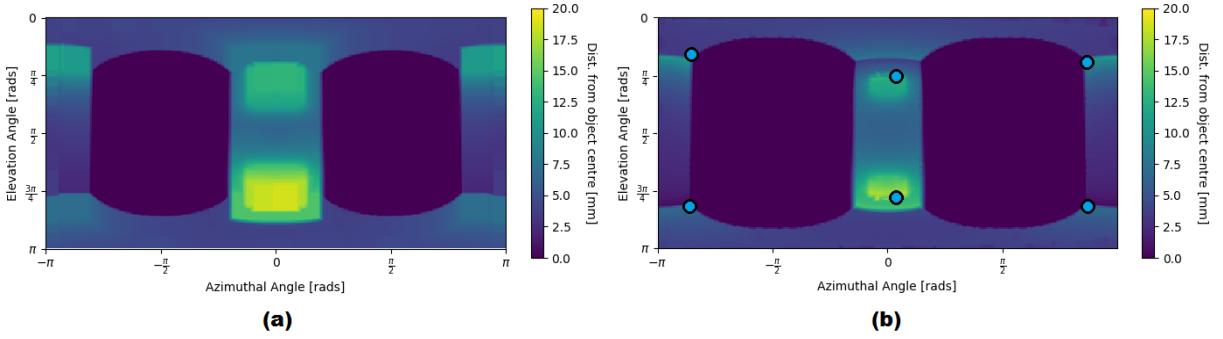


Figure 4.17: Finding local maxima on a difference surface image. (a) Difference surface image filtered with a maximum filter (kernel size of 5) (b) Difference surface image with local maxima identified.

The number of local maxima found in the difference surface image determines the potential grasping points on the object that can be used to control the object towards the desired shape. The set of potential grasp points for a given difference surface image is referred to as GP, with an arbitrary grasp point $\hat{\mathbf{p}}^i \in \text{GP}$. The arbitrary grasp point $\hat{\mathbf{p}}^i$ represents a point on the difference surface image, S_{diff} , and is represented by an azimuthal angle, $\hat{p}_\phi^i$, an elevation angle, $\hat{p}_\theta^i$, and a radius, $\hat{p}_r^i = S_{diff}(\hat{p}_\phi^i, \hat{p}_\theta^i)$. Any planned grasp must adhere to the force closure constraint and the kinematic constraints of the hand. The kinematic constraint of the hand that requires equal and opposite rotation of the non-fixed fingers results in two free choices for distributing the fingers around the palm of the hand. Therefore, the algorithm searches for two locations to optimally deform the object towards the desired shape and starts by determining how to distribute the fingers around the palm of the hand to contact the object at these points.

If two or more candidate grasp points are identified, all combinations of two points are sorted based on the total distance between the undeformed shape and the desired shape at both surface points. Grasp planning begins using the candidate grasp points with the largest total distance. The two candidate points are labelled $\hat{\mathbf{p}}^A$ and $\hat{\mathbf{p}}^B$, the point with the larger deformation being set to $\hat{\mathbf{p}}^A$. These two points are used to plan the grasping pose of the robotic hand. The validity of the grasp is checked after completing the grasp plan. If the grasp does not correspond to any valid configurations, the next set of points

is selected in descending order until a valid configuration is found. Details on how the validity is checked are presented after the grasp plan.

Before continuing the grasp points selection process, relevant coordinate systems and transformations are introduced in Figure 4.18. The robot coordinate frame, *rob*, is the frame fixed at the origin of the CRS F3 robot (introduced in Section 3.2.2). The hand coordinate frame, *hand*, is fixed at the wrist of the Barrett Hand (introduced in Section 3.2.3). The *obj* coordinate frame is fixed at the centre of the undeformed object, and the orientation is determined based on the chosen contact points described in the remainder of this section. The *grasp* coordinate frame represents the ideal pose for the Barrett Hand to grasp and deform the object at the chosen contact points. The robotic system will deform the object by moving the *hand* reference frame to align with the *grasp* reference frame. The remainder of this section describes how these frames are obtained. All coordinates introduced in the remainder of this section are given a subscript to indicate which coordinate frame they are represented in.

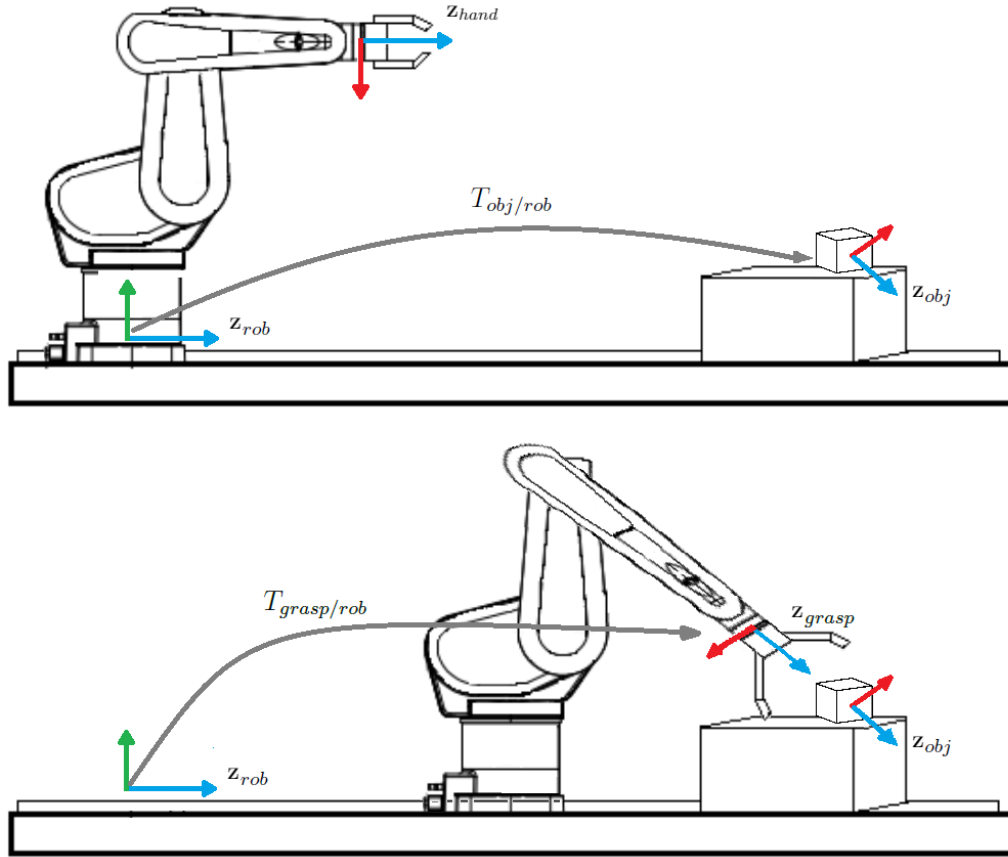


Figure 4.18: Relevant references frames and transformations for grasp points selection.

The two candidate grasp points, $\hat{\mathbf{p}}^A$ and $\hat{\mathbf{p}}^B$, obtained from the difference surface image are transformed to the Cartesian CRS F3 robot coordinate system using the inverse transformations from Section 4.2.1.2. The candidate grasp points in the robot coordinate

system, $\mathbf{p}_{rob}^A$ and $\mathbf{p}_{rob}^B$, along with the estimated centre of the object, $\mathbf{c}_{rob}$, are used to define the object coordinate system. This coordinate system is used to place the z -axis of the hand through the centre of the object and defines the cross-sectional plane of the object to plan the grasping points, as discussed in Section 4.2.1.1.

The origin of the coordinate system is set to be the centre of the undeformed object, $\mathbf{c}_{rob}$. The x -axis of the object coordinate system, $\mathbf{x}_{obj}$, is the vector from the centre of the object to the candidate grasp point with the largest difference between the desired surface and the estimated surface, $\mathbf{p}_{rob}^A$. The z -axis of the object coordinate system, $\mathbf{z}_{obj}$, is defined by the cross product of $(\mathbf{p}_{rob}^A - \mathbf{c}_{rob})$ and $(\mathbf{p}_{rob}^B - \mathbf{c}_{rob})$. Because the cross product is anticommutative, $\mathbf{z}_{obj}$ has two possible solutions. Both solutions are checked to determine whether they correspond to valid grasp poses.

Lastly, the $\mathbf{y}_{obj}$ -axis is defined by making the coordinate system right-handed. An illustration of this coordinate system definition is shown in Figure 4.19. Defining the coordinate system in this way generates the cross-section of the object on the $\mathbf{x}_{obj}\mathbf{y}_{obj}$ -plane on which the grasp points are planned, as discussed in Section 4.2.1.1.

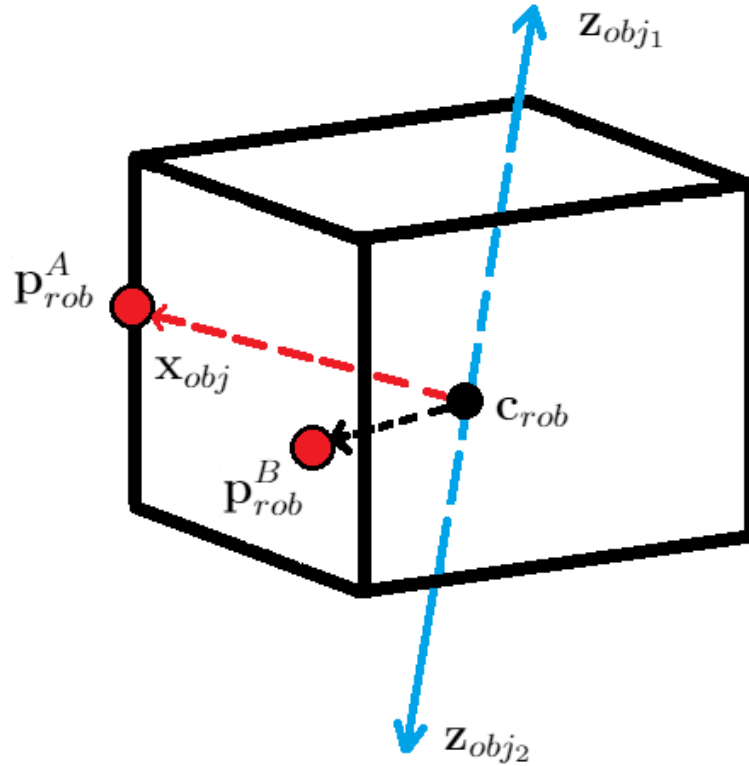


Figure 4.19: The object coordinate system based on candidate grasping points identified from the difference surface image.

$$\mathbf{x}_{obj} = (\mathbf{p}_{rob}^A - \mathbf{c}_{rob}) \quad (4.42)$$

$$\mathbf{z}_{obj1} = \mathbf{x}_{obj} \times (\mathbf{p}_{rob}^B - \mathbf{c}_{rob}) \quad (4.43)$$

$$\mathbf{z}_{obj2} = (\mathbf{p}_{rob}^B - \mathbf{c}_{rob}) \times \mathbf{x}_{obj} \quad (4.44)$$

$$\mathbf{y}_{obj1} = \mathbf{z}_{obj1} \times \mathbf{x}_{obj} \quad (4.45)$$

$$\mathbf{y}_{obj2} = \mathbf{z}_{obj2} \times \mathbf{x}_{obj} \quad (4.46)$$

This process defines two possible object coordinate systems, corresponding to two unique grasping poses for the end-effector. Depending on the specific task and any obstacles in the environment, it is possible that only one orientation is feasible. For example, when picking an object off of a tabletop, only the $\mathbf{z}_{obj}$ -axis that defines a grasp from above the object is valid, as it is impossible to grasp an object off of a surface from underneath that surface. If both coordinate systems represent valid grasping positions for the arm and hand, the one that is used is determined by the manipulator path planning algorithm in Section 4.2.2. The remainder of the grasp selection algorithm description proceeds to describe the grasp selection based on one object coordinate system. Using the coordinate axes defined above, a homogeneous transformation matrix that defines the object coordinate system with respect to the robot coordinate system is:

$$T_{obj/rob} = \begin{bmatrix} x_{objx} & y_{objx} & z_{objx} & c_{robx} \\ x_{objy} & y_{objy} & z_{objy} & c_{rob_y} \\ x_{objz} & y_{objz} & z_{objz} & c_{robz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.47)$$

The homogeneous transformation matrix can be used to convert the first two fingertip positions to the newly defined coordinate system.

$$\begin{bmatrix} \mathbf{p}_{obj}^A \\ 1 \end{bmatrix} = T_{obj/rob}^{-1} \cdot \begin{bmatrix} \mathbf{p}_{rob}^A \\ 1 \end{bmatrix} \quad (4.48)$$

$$\begin{bmatrix} \mathbf{p}_{obj}^B \\ 1 \end{bmatrix} = T_{obj/rob}^{-1} \cdot \begin{bmatrix} \mathbf{p}_{rob}^B \\ 1 \end{bmatrix} \quad (4.49)$$

The grasping algorithm plans contact points on the $\mathbf{x}_{obj}\mathbf{y}_{obj}$ -plane of the object's coordinate system to accommodate for the kinematics constraints of the hand. This idea was first described in Section 4.2.1.1 and illustrated in Figure 4.12. In practice, the contact points are not necessarily distributed on this plane because the distance from the centre of the object to its surface is not guaranteed to be the same at each point. Due to the Barrett Hand's kinematics constraints, there are only two free choices for distributing the fingers around the palm of the hand. The placement of the first two fingertips determines the possible locations of the third fingertip and the finger assignment for each grasping point.

After all three grasping points are planned on the $\mathbf{x}_{obj}\mathbf{y}_{obj}$ -plane of the object's coordinate system, the error in the $\mathbf{z}_{obj}$ -axis is minimized by adjusting the end-effector position along this axis.

The finger assignments indicate which finger of the hand is responsible for contacting each grasp point. In order to determine the valid configurations of the hand, the first thing that needs to be calculated is the angle between the two identified fingertip positions. Because of how the coordinate system is defined, this is an angle on the $\mathbf{x}_{obj}\mathbf{y}_{obj}$ -plane of the object's coordinate system. With $\mathbf{p}_{obj}^A$ defined to be along the $\mathbf{x}_{obj}$ -axis, the angle between the candidate grasp points can be calculated as:

$$\alpha = \text{atan2}(p_{objy}^B, p_{objx}^B) \quad (4.50)$$

The angle, α , between the two grasp points determines the possible locations for the third grasp point, $\mathbf{p}_{obj}^C$, that ensures a stable grasp. The criteria for grasp stability in this work, first described in Section 4.2.1.1, is that the spread angle of the fingers of the Barrett Hand, θ_7 , must be greater than 90° , to ensure there are fingertip forces in opposing directions. The angle, α , determines the possible ways to distribute the fingers around the palm to ensure a stable grasp.

If α is greater than 0° and less than or equal to 90° there is only one possible way to distribute the fingers around the object while satisfying the stability criteria and the kinematic constraints of the hand. This corresponds to assigning the third fingertip position, $\mathbf{p}_{obj}^C$, to the fixed finger of the hand, F3, and placing it opposite the midpoint of the arc defined by $\mathbf{p}_{obj}^A$ and $\mathbf{p}_{obj}^B$, illustrated in Figure 4.20. This distribution of the fingers around the palm of the hand is realized by setting the spread angle of the hand, θ_7 , equal to $\pi - \frac{\alpha}{2}$. Mathematically $\mathbf{p}_{obj}^C$ is calculated by rotating $\mathbf{p}_{obj}^A$ about the $\mathbf{z}_{obj}$ -axis by an angle, ψ , equal to $\pi + \frac{\alpha}{2}$.

$$\mathbf{p}_{obj}^C = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 & 0 \\ \sin(\psi) & \cos(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{p}_{obj}^A \quad (4.51)$$

Figure 4.20 illustrates the placement of the fingers on the $\mathbf{x}_{obj}\mathbf{y}_{obj}$ -plane of the object's coordinate system. In this configuration, the hand's geometry dictates that F1 is placed at the position corresponding to $\mathbf{p}_{obj}^A$, and F2 is placed at the position corresponding to $\mathbf{p}_{obj}^B$, that is at angle α from $\mathbf{p}_{obj}^A$. New variables, $\mathbf{f}_{obj}^1$, $\mathbf{f}_{obj}^2$ and $\mathbf{f}_{obj}^3$, are defined in order to associate the grasping positions with the required fingers. In this case, the assignments are:

$$\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^A \quad (4.52)$$

$$\mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^B \quad (4.53)$$

$$\mathbf{f}_{obj}^3 = \mathbf{p}_{obj}^C \quad (4.54)$$

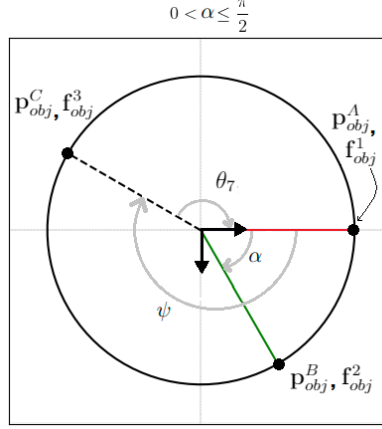


Figure 4.20: Possible third grasping point given two grasping points and the hand's kinematic constraints for $0 < \alpha \leq \frac{\pi}{2}$.

When α is greater than 90° and less than 180° , there are three possible configurations that generate stable grasps. Each configuration is associated with assigning a different fingertip position to the fixed finger, F3. In the first case, F3 is assigned to the third fingertip position, $\mathbf{p}_{obj}^C$, which is positioned at the location calculated by rotating $\mathbf{p}_{obj}^A$ by ψ equal to $\pi + \frac{\alpha}{2}$. The spread angle of the hand, θ_7 , is set to $\pi - \frac{\alpha}{2}$ to generate a stable grasp and realize the remaining fingertip positions. In this configuration, the hand's geometry dictates that $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^A$ and $\mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^B$. This configuration is illustrated in Figure 4.21a.

The second case places the fixed finger, F3, at $\mathbf{p}_{obj}^B$, and the spread angle of the hand, θ_7 , is α to satisfy the position of $\mathbf{p}_{obj}^A$ and define the location of $\mathbf{p}_{obj}^C$. This results in ψ equal to 2α and the remaining finger assignments dictated by the hand's geometry, $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^C$ and $\mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^A$. This configuration is illustrated in Figure 4.21b.

The third case places the fixed finger, F3, at $\mathbf{p}_{obj}^A$, and the spread angle of the hand, θ_7 , is α to satisfy the position of $\mathbf{p}_{obj}^B$ and define the location of $\mathbf{p}_{obj}^C$. This results in ψ equal to $-\alpha$ and the remaining finger assignments dictated by the hand's geometry, $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^B$ and $\mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^C$. This configuration is illustrated in Figure 4.21c.

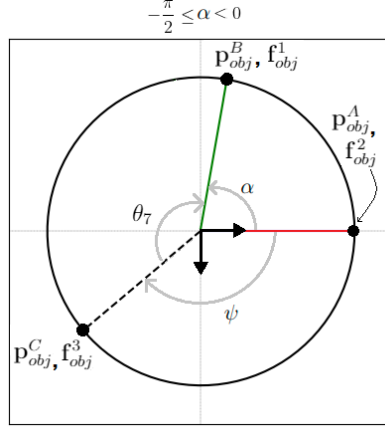


Figure 4.23: Possible third grasping point given two grasping points and the hand's kinematic constraints for $-\frac{\pi}{2} \leq \alpha < 0$.

For all of the cases described above, the variables that determine the hand configuration are shown in Table 4.4, and the process is described in detail in Algorithm 5.

It should be noted that some limit cases are excluded from the analysis. In particular, α equal 0° is not analyzed because the grasp point candidates must be unique, and an angle of 0° is not possible for unique grasp points. Additionally, α equal to 180° is not considered because there is no way to satisfy this work's force closure criteria if the two candidate grasp points are 180° apart. If this case occurs, the candidate grasp points are determined to be invalid, and the next set of points is considered.

Table 4.4: Variables for valid grasping configurations of the robotic hand.

α	Case	θ_7	ψ	$\mathbf{f}_{obj}^1$	$\mathbf{f}_{obj}^2$	$\mathbf{f}_{obj}^3$
$0 < \alpha \leq \frac{\pi}{2}$	—	$\pi - \frac{\alpha}{2}$	$\pi + \frac{\alpha}{2}$	$\mathbf{p}_{obj}^A$	$\mathbf{p}_{obj}^B$	$\mathbf{p}_{obj}^C$
$\frac{\pi}{2} < \alpha < \pi$	1	$\pi - \frac{\alpha}{2}$	$\pi + \frac{\alpha}{2}$	$\mathbf{p}_{obj}^A$	$\mathbf{p}_{obj}^B$	$\mathbf{p}_{obj}^C$
	2	α	2α	$\mathbf{p}_{obj}^C$	$\mathbf{p}_{obj}^A$	$\mathbf{p}_{obj}^B$
	3	α	$-\alpha$	$\mathbf{p}_{obj}^B$	$\mathbf{p}_{obj}^C$	$\mathbf{p}_{obj}^A$
$-\pi < \alpha < -\frac{\pi}{2}$	1	$\pi + \frac{\alpha}{2}$	$\pi + \frac{\alpha}{2}$	$\mathbf{p}_{obj}^B$	$\mathbf{p}_{obj}^A$	$\mathbf{p}_{obj}^C$
	2	$-\alpha$	$2(\alpha - \pi)$	$\mathbf{p}_{obj}^A$	$\mathbf{p}_{obj}^C$	$\mathbf{p}_{obj}^B$
	3	$-\alpha$	$-\alpha$	$\mathbf{p}_{obj}^C$	$\mathbf{p}_{obj}^B$	$\mathbf{p}_{obj}^A$
$-\frac{\pi}{2} \leq \alpha < 0$	—	$\pi + \frac{\alpha}{2}$	$\pi + \frac{\alpha}{2}$	$\mathbf{p}_{obj}^B$	$\mathbf{p}_{obj}^A$	$\mathbf{p}_{obj}^C$

Algorithm 5: Find all possible locations for third grasping point

Input:

$\mathbf{p}_{obj}^A, \mathbf{p}_{obj}^B$ - candidate grasp points

Output:

θ_7 - the spread angle of the Barrett Hand

$[\mathbf{p}_{obj}^C]$ - list of possible coordinates of the third grasping position

The associations of each finger for a given set of grasping points

```
1  $\alpha = \text{atan2}(p_{objy}^B, p_{objx}^B)$ 
2 if  $0 < \alpha \leq \frac{\pi}{2}$  then
3    $\theta_7 = \pi - \frac{\alpha}{2}$ 
4    $\mathbf{p}_{obj}^C = \text{RotateZ}(\mathbf{p}_{obj}^A, \pi + \frac{\alpha}{2})$ 
5    $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^A, \mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^B, \mathbf{f}_{obj}^3 = \mathbf{p}_{obj}^C$ 
6 else if  $\frac{\pi}{2} < \alpha < \pi$  then
7   case 1 do
8      $\theta_7 = \pi - \frac{\alpha}{2}$ 
9      $\mathbf{p}_{obj}^C = \text{RotateZ}(\mathbf{p}_{obj}^A, \pi + \frac{\alpha}{2})$ 
10     $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^A, \mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^B, \mathbf{f}_{obj}^3 = \mathbf{p}_{obj}^C$ 
11  case 2 do
12     $\theta_7 = \alpha$ 
13     $\mathbf{p}_{obj}^C = \text{RotateZ}(\mathbf{p}_{obj}^A, 2\alpha)$ 
14     $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^C, \mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^A, \mathbf{f}_{obj}^3 = \mathbf{p}_{obj}^B$ 
15  case 3 do
16     $\theta_7 = \alpha$ 
17     $\mathbf{p}_{obj}^C = \text{RotateZ}(\mathbf{p}_{obj}^A, -\alpha)$ 
18     $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^B, \mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^C, \mathbf{f}_{obj}^3 = \mathbf{p}_{obj}^A$ 
19 else if  $-\pi < \alpha < -\frac{\pi}{2}$  then
20  case 1 do
21     $\theta_7 = \pi + \frac{\alpha}{2}$ 
22     $\mathbf{p}_{obj}^C = \text{RotateZ}(\mathbf{p}_{obj}^A, \pi + \frac{\alpha}{2})$ 
23     $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^B, \mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^A, \mathbf{f}_{obj}^3 = \mathbf{p}_{obj}^C$ 
24  case 2 do
25     $\theta_7 = -\alpha$ 
26     $\mathbf{p}_{obj}^C = \text{RotateZ}(\mathbf{p}_{obj}^A, 2(\alpha - \pi))$ 
27     $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^A, \mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^C, \mathbf{f}_{obj}^3 = \mathbf{p}_{obj}^B$ 
28  case 3 do
29     $\theta_7 = -\alpha$ 
30     $\mathbf{p}_{obj}^C = \text{RotateZ}(\mathbf{p}_{obj}^A, -\alpha)$ 
31     $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^C, \mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^B, \mathbf{f}_{obj}^3 = \mathbf{p}_{obj}^A$ 
32 else if  $-\frac{\pi}{2} \leq \alpha < 0$  then
33    $\theta_7 = \pi + \frac{\alpha}{2}$ 
34    $\mathbf{p}_{obj}^C = \text{RotateZ}(\mathbf{p}_{obj}^A, \pi + \frac{\alpha}{2})$ 
35    $\mathbf{f}_{obj}^1 = \mathbf{p}_{obj}^B, \mathbf{f}_{obj}^2 = \mathbf{p}_{obj}^A, \mathbf{f}_{obj}^3 = \mathbf{p}_{obj}^C$ 
```

In the cases where there are more than one possible positions for $\mathbf{p}_{obj}^C$, a decision must be made on which fingertip position to proceed with. Various criteria can be used to determine the third position. Priority can be given to grasp stability, deformation control, or fingertip positional accuracy. In order to assess these criteria, the undeformed object surface image, S_i , and difference surface image, S_{diff} , are used. The surface image coordinates $\hat{\mathbf{p}}^A$ and $\hat{\mathbf{p}}^B$ are already known, while $\hat{\mathbf{p}}^C$ is calculated from $\mathbf{p}_{obj}^C$. The coordinates of the third grasp point in the robot coordinates system are calculated as follows:

$$\begin{bmatrix} \mathbf{p}_{rob}^C \\ 1 \end{bmatrix} = T_{obj/rob} \cdot \begin{bmatrix} \mathbf{p}_{obj}^C \\ 1 \end{bmatrix} \quad (4.55)$$

The coordinates can then be transformed to spherical coordinates, and the exact radius of the object's surface at each grasping point can be obtained from the surface image of the undeformed object, S_i . Additionally, the difference between the radius of the underformed object and the desired object can be obtained for each grasp point from the difference surface image, S_{diff} .

Table 4.5 displays multiple ways to choose the third finger position and what each method prioritizes.

Table 4.5: Different criteria that can be used to select third finger position.

Criteria	Condition	Priority
Minimize deformation	$\arg \min_i S_{diff}(\hat{\mathbf{p}}_i^C)$	Grasp stability
Maximize finger spread	$\arg \max_i \theta_{\tau_i}$	Grasp stability
Maximize deformation	$\arg \max_i S_{diff}(\hat{\mathbf{p}}_i^C)$	Deformation control
Closest to mean radius	$\arg \min_i [S_i(\hat{\mathbf{p}}_i^C) - \frac{1}{2}(S_i(\hat{\mathbf{p}}^B) + S_i(\hat{\mathbf{p}}^A))]$	Fingertip position

Prioritizing grasp stability can be done by using the position for $\mathbf{p}_{obj}^C$ that has the smallest corresponding value on the difference surface image. This ensures that there is a fixed, stable point for the forces of the other fingertips to work against. An alternative method prioritizing grasp stability is to select the position for the third finger that corresponds with the largest spread of the fingers. This will improve the force closure of the grasp by prioritizing that the forces are in opposite directions.

Maximizing the deformation control is done by checking the difference surface, S_{diff} , at each of the possible positions for $\mathbf{p}_{obj}^C$ to find the position with the largest value. This is done so that the object deforms more closely towards the desired shape.

If prioritizing the accuracy of the fingertip positions is desired, then the third finger position can be chosen so that its initial radius is closest to the average of the other two fingers. Due to the kinematic constraints of the hand, the fingertip positions are less accurate when the grasp point radiuses are significantly different from the mean radius of all points.

Once the position for $\mathbf{p}_{obj}^C$ and the finger associations for each point are determined, the grasp orientation can be fully deduced by rotating $T_{obj/rob}$ by the amount $\mathbf{f}_{obj}^3$ is rotated

from $\mathbf{p}_{obj}^A$, the initial $\mathbf{x}_{obj}$ -axis. This is because the x -axis of the end-effector naturally lines up with the fixed finger (F3) of the hand, as seen in Figure 3.7.

$$\beta = \text{atan2}(f_{obj_y}^3, f_{obj_x}^3) \quad (4.56)$$

Therefore, to get the final grasping orientation of the end-effector it must be rotated by β about its z -axis.

This orientation can then be used with a position that places the end-effector the required distance away from the object along the $\mathbf{z}_{obj}$ -axis. The joint positions for each finger that place the fingers on the surface of the object must be calculated. The radius of each fingertip can be found using the initial surface image coordinates of the grasping points and the finger associations. The radius for each finger to place the fingertip on the surface of the object can be calculated using the undeformed object surface image and the finger associations determined in Algorithm 5. The surface image coordinates for each fingertip are $\hat{\mathbf{f}}^1$, $\hat{\mathbf{f}}^2$ and $\hat{\mathbf{f}}^3$. The azimuthal angle (ϕ) and elevation angle (θ) coordinates for each fingertip are calculated by converting the coordinates in the object coordinate system to polar coordinates. The undeformed object radius at each fingertip position is calculated using the fingertips' surface image coordinates and the undeformed object surface image, S_i .

$$\hat{f}_r^1 = S_i(\hat{f}_\phi^1, f_\theta^1) \quad (4.57)$$

$$\hat{f}_r^2 = S_i(\hat{f}_\phi^2, f_\theta^2) \quad (4.58)$$

$$\hat{f}_r^3 = S_i(\hat{f}_\phi^3, f_\theta^3) \quad (4.59)$$

Referring back to the inverse kinematics of the Barrett Hand in Section 3.2.3 and using Equation 3.34, the radius values calculated above can be used to get the joint encoder value that places each fingertip on the surface of the object.

$$e_1 = \arg \min_{e_1} |(t_r^1(\theta_7, e_1) - f_r^1)| \quad (4.60)$$

$$e_2 = \arg \min_{e_2} |(t_r^2(\theta_7, e_2) - f_r^2)| \quad (4.61)$$

$$e_3 = \arg \min_{e_3} |(t_r^3(\theta_7, e_3) - f_r^3)| \quad (4.62)$$

The values for t_z^1 , t_z^2 , and t_z^3 can be calculated using the encoder values, the spread of the hand (θ_7), and the forward kinematic model described in Equation 3.32.

A simplification has been made to assume the fingers will all have the same z value with respect to the origin of the hand coordinate system. When grasping the object, the mean $\mathbf{z}_{hand}$ distance for each finger at the desired radius is used to position the palm of the hand in the grasping position.

$$z_{mean} = \frac{(t_z^1 + t_z^2 + t_z^3)}{3} \quad (4.63)$$

This simplification causes inaccuracies in the exact position of the fingertips but is necessary to account for the kinematic constraints of the hand.

The rotation about the $\mathbf{z}_{obj}$ -axis, the distance along the $\mathbf{z}_{obj}$ -axis, the spread of the fingers and the encoder positions for the three fingers fully define the pose and configuration of the end-effector to grasp and deform the object. The grasp pose can be expressed by the following homogeneous transformation matrix:

$$T_{grasp/rob} = T_{obj/rob} \cdot \begin{bmatrix} \cos(\beta) & -\sin(\beta) & 0 & 0 \\ \sin(\beta) & \cos(\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -z_{mean} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.64)$$

Figure 4.24 illustrates the hand in the grasping pose, with all required variables identified. To place the end-effector in the grasping pose the manipulator must align the hand coordinate system with the coordinate system defined by $T_{grasp/rob}$.

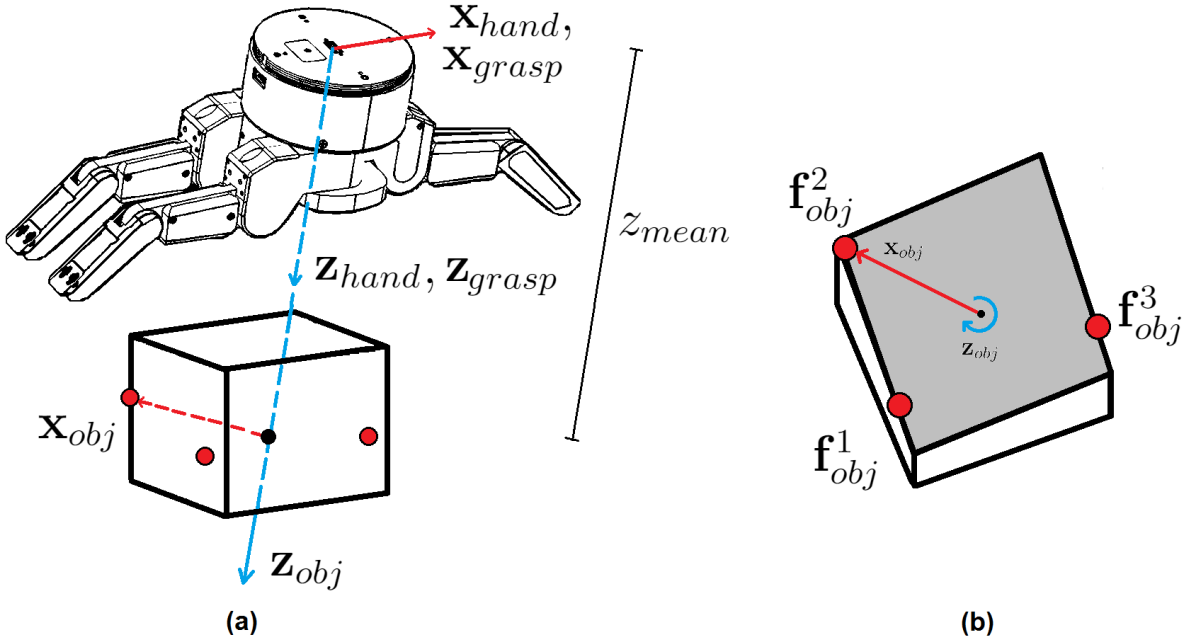


Figure 4.24: Barrett Hand placed in the grasping pose to grasp and deform a 3D object, with coordinate systems and variables identified. (a) View of the hand in grasping pose before engaging fingers. (b) Cross-sectional view of the object to be grasped with contact points identified.

Before executing the grasp, the end-effector pose must be checked to ensure that it corresponds to a valid grasp for the hand. There are three conditions that are checked

to ensure that the pose is valid. The first condition is the pose does not cause collisions between the robot hand and any obstacles in the environment. The second condition checks to ensure the fingertips of the hand do not collide with any obstacles as they grasp the object. Finally, the grasp is validated by checking to see if the radius at each grasp point is less than the maximum radius of the fingertips. If any of these conditions are not met, the grasp pose is redefined using the next pair of grasp points in the sorted list of candidate points described above. This process is repeated until a valid grasp is found or the list is exhausted, in which case no valid grasps for shape control are identified.

The full process of determining a valid grasp pose for shape control from an undeformed object surface image and a difference surface image is provided in Algorithm 6.

Algorithm 6: Potential grasp points selection method

Input:

S_i - undeformed object surface image

S_{diff} - difference surface image

Output:

$T_{grasp/rob}$ - optimal valid grasping pose

e_1, e_2, e_3, θ_7 - robot hand parameters

```

1 GP = FindLocalMaxima ( $S_{diff}$ )
2 TotalDeformation = PairGraspingPoints (GP)
3 SortedPairs = Sort (TotalDeformation)
4 for  $i < \text{num}(\text{SortedPairs})$  do
5    $\hat{\mathbf{p}}^A, \hat{\mathbf{p}}^B = \text{SortedPairs}(i)$ 
6    $T_{obj_i} = \text{GenerateCoordinateSystems} (\hat{\mathbf{p}}^A, \hat{\mathbf{p}}^B)$ 
7    $\mathbf{p}_{obj}^C, \theta_7 = \text{FindThirdPoint} (\hat{\mathbf{p}}^A, \hat{\mathbf{p}}^B)$ 
8   Get finger/grasp point association ( $\mathbf{f}_{obj}^1, \mathbf{f}_{obj}^2, \mathbf{f}_{obj}^3$ ) and ( $\hat{\mathbf{f}}^1, \hat{\mathbf{f}}^2, \hat{\mathbf{f}}^3$ )
9    $\beta = \text{atan2}(f_{obj_y}^3, f_{obj_x}^3)$ 
10  foreach finger  $k$  do
11     $\hat{f}_r^k = S_i(\hat{f}_\phi^k, \hat{f}_\theta^k)$ 
12     $e_k = \text{InverseHandKinematics} (\hat{f}_r^k, \theta_7)$ 
13     $t_z^k = \text{ForwardHandKinematics} (e_k, \theta_7)$ 
14  end
15   $z_{mean} = (t_z^1 + t_z^2 + t_z^3)/3$ 
16   $T_{grasp/rob} = T_{obj/rob} \cdot \text{RotateZ} (\beta) \cdot \text{TranslateZ} (-z_{mean})$ 
17  if  $T_{grasp/rob}$  is valid then
18    break
19  end
20 end
```

Results shown in Figure 4.25 present multiple desired 3D shapes and the difference surface images associated with the initial 3D object. The difference surface images have all the potential grasp points identified as well as the optimal valid grasp points that are used to determine the end-effector pose. In the cases where there is more than one candidate position for the third finger position, one of the criteria defined in Table 4.5 is used to

determine the position. In these examples, the criteria used to determine the third finger position was to maximize the deformation control by selecting the grasp point with the maximum value on the difference surface image.

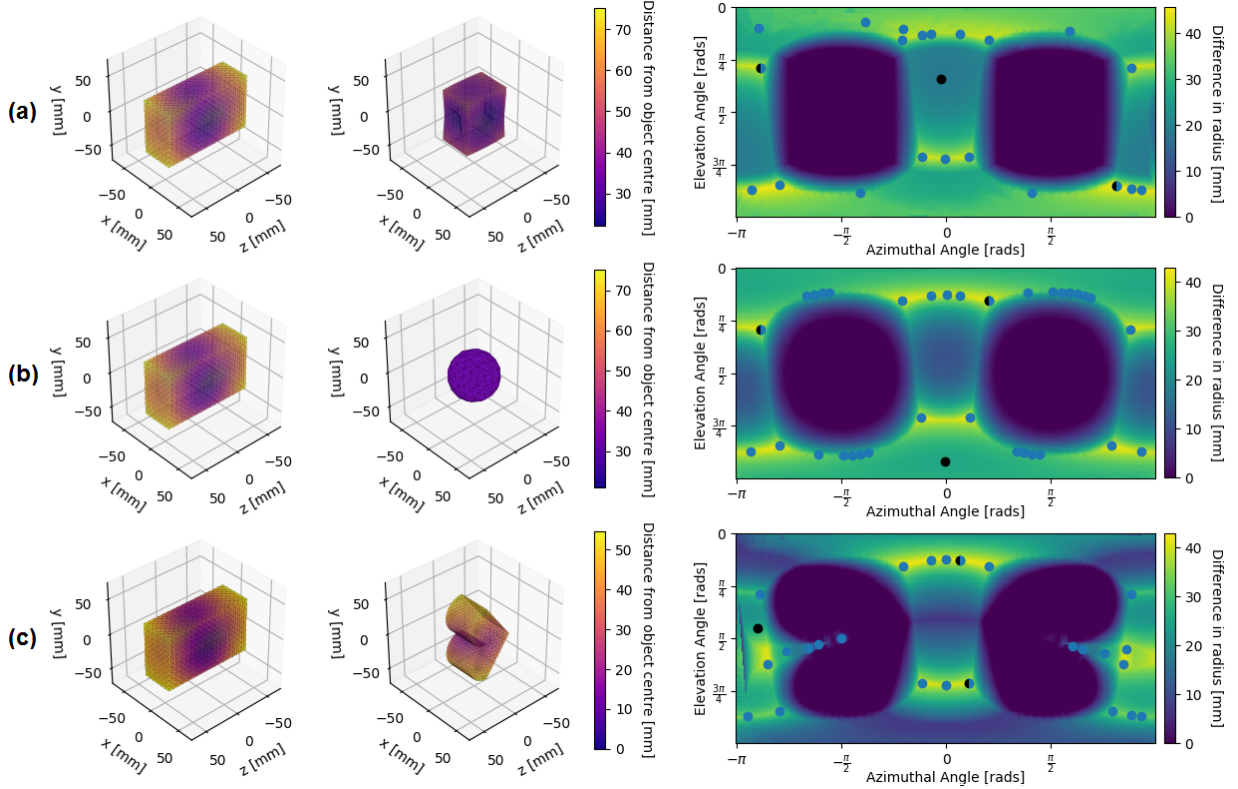


Figure 4.25: Additional examples of initial undeformed 3D objects, a desired 3D shape to deform towards and the resulting difference surface image between them. Difference surface images have all potential grasp points identified (blue) and the optimal valid grasp points identified in (black). The two chosen candidate points, p^A and p^B , are coloured blue and black.

4.2.2 Robot Pose and Trajectory Planning

Once the initial grasp points are selected, a trajectory for the arm must be planned to place the end-effector at the desired position and orientation in order to optimally deform the object towards the desired 3D shape. The trajectory will move the robot from an initial home position to the object along a collision-free path. Path planning in this way automatically generates trajectories and motion control for robots with known kinematics to reach predefined targets in autonomous systems.

As discussed in Section 3.2.2, the CRS F3 robotic arm is a 7-DoF robotic manipulator and therefore has no closed-form solution for the inverse kinematics as the manipulator is over-actuated. Therefore, to determine the final joint positions for the manipulator, the trajectory planning is done in the configuration space of the robot. The algorithm

used to accomplish this overcomes the redundancy of the manipulator by using the robot’s Jacobian to transform joint motion in the configuration space to end-effector motion in the robot workspace. The solution presented in this work is a sampling-based motion planning algorithm based on the Jacobian Transpose-directed Rapidly Exploring Random Trees (JT-RRTs) algorithm [63] presented in Section 2.3. This method leverages the RRT algorithm’s [60] power for exploring very high-dimensional configuration spaces while also using the kinematic solution of the CRS F3 robot manipulator developed in Section 3.2.2. The Jacobian guided RRT solution focuses the end-effector’s movement towards the desired goal pose calculated in Section 4.2.1. A hybrid approach is developed to move the end-effector close to a goal pose using the known methods presented above and then moving to the exact pose by fixing the value of the track (the first degree of freedom of the CRS F3 robot) and solving the inverse kinematics of the 6-DoF using the solution developed in Section 3.2.2. This hybrid solution allows the end-effector to be positioned accurately and quickly while leveraging known path planning algorithms.

As discussed in Section 2.3, sampling-based motion planning techniques rely heavily on collision-detection functions to determine whether moving to a pose will cause a collision between the robot and the environment or a robot self-collision. Section 4.2.2.1 details how collision detection is done for the CRS F3 robot in this work. Section 4.2.2.2 describes in detail the motion planning algorithm implemented in this research. Lastly, Section 4.2.2.3 describes the motion planning pipeline for grasping deformable objects with the robot arm and hand system.

4.2.2.1 Collision Detection

Robot motion planning relies heavily on collision detection to determine whether poses can be safely moved to. This research relies on detecting collisions by attaching virtual collision spheres to the robot and end-effector, and detecting whether these would come in contact with each other, with known obstacles or with sensed obstacles at a certain joint configuration. The virtual spheres are attached to the robot at various locations so that the physical robot is fully enclosed. Possible collisions are detected by calculating the forward kinematics for each position on the robot where a sphere is attached using the robot model developed in Section 3.2.2. Figure 4.26 shows a part model of the robot with and without the attached collision spheres.

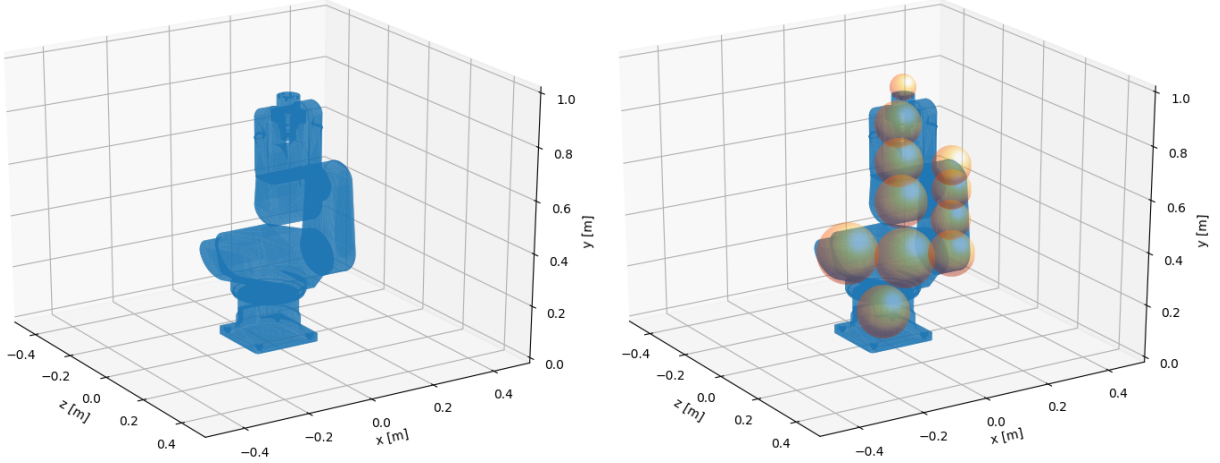


Figure 4.26: A part model of the CRS F3 robot manipulator illustrated with and without virtual collision spheres.

Calculating robot self collisions using virtual spheres requires minimal computing as it is trivial to calculate the distance between the surfaces of spheres with known locations and radiuses. Additionally, known obstacles such as the robot platform or walls can be added as planes which also allow easy collision computation. Lastly, any detected obstacles can be added as point clouds, and each sphere can be checked for a collision with the sensed obstacles. While checking collisions between a sphere and a point cloud is more computationally expensive, it allows the trajectory planning to be more flexible when previously unknown obstacles are in the robot workspace.

4.2.2.2 JT-RTT Motion Planner

The Jacobian Transpose-directed Rapidly Exploring Random Trees (JT-RRTs) algorithm [63] leverages the power of the RRT algorithm [60] for exploring very high-dimensional configuration spaces while also using the known kinematics of a robot manipulator to focus the end-effector movement towards a desired goal pose. This known method is implemented with the kinematic model of the CRS F3 robot developed in Section 3.2.2. An optimization is also introduced to the algorithm by using the RRT* algorithm [59] to potentially find shorter collision-free paths than the traditional JT-RRT method. Once the end-effector is moved close to the goal, a hybrid approach is developed that includes fixing the first joint of the robot and solving the 6-DoF inverse kinematic problem to find joint positions for the robot to move to the exact grasping pose calculated by Equation 4.64.

The configuration space, $\mathcal{C}$, of the CRS F3 robot is defined by all possible configurations that the robot may assume, $\mathcal{Q}$, with a single joint configuration described by $\mathbf{q}$. Each joint is described by either a distance, D_0 , for the linear joint associated with the track at the base of the robot, or an angle, θ_i , for the six revolute joints. An arbitrary joint configuration, $\mathbf{q}_n$, for the CRS F3 manipulator is described by:

$$\mathbf{q}_n = (D_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \quad (4.65)$$

The workspace of the robot, $\mathcal{X}$, is the space of all possible end-effector positions and orientations, SE(3). An arbitrary pose, x_n , is defined by the Cartesian coordinates and Euler angles in the robot coordinate system.

$$\mathbf{x}_n = (x, y, z, r_z, r_y, r_x) \quad (4.66)$$

The path planning problem can be mathematically defined as searching for a mapping from a starting joint configuration, $\mathbf{q}_i \in \mathcal{Q}$, to a final end-effector position, $\mathbf{x}_g \in \mathcal{X}$.

The Jacobian of the manipulator is leveraged to quickly plan the motion towards the goal, using the fact that the Jacobian maps the rate of change of the joints to the rate of change of the pose variables. This is expressed as:

$$\dot{\mathbf{x}} = J\dot{\mathbf{q}} \quad (4.67)$$

In general, a control solution of the form $\dot{\mathbf{q}} = kJ^{-1}(\mathbf{x}_g - \mathbf{x})$ would guide the end-effector to the goal. While this technique is effective, it is quite slow to compute due to the Jacobian of the 7-DoF manipulator not having a closed-form solution. Therefore, the inverse would have to be computed numerically at each iteration. This process is rather time-consuming and relies on complex techniques to approximate the numerical Jacobian at each iteration.

Therefore an alternative approach is to use the transpose of the Jacobian rather than the inverse, as was first presented in [117]. This results in the instantaneous motion of the end-effector given by:

$$\dot{\mathbf{q}} = kJ^T(\mathbf{x}_g - \mathbf{x}) \quad (4.68)$$

In general, this type of controller is guaranteed to reach a goal pose, but it does not account for obstacles in the workspace or motion constraints on the joints of the robot. Implementing the controller in Equation 4.68 would lead to infeasible joint velocities when the robot is far from the goal pose, and it does not avoid moving the robot through any poses associated with robot singularities. To implement the Jacobian transpose based-controller, additional path planning techniques need to be used to compensate for these limitations.

The idea of goal-directed RRT algorithms, introduced in Section 2.3, is that they preserve the random nature of the sampling-based path planning algorithms while improving the time to find a collision-free path by occasionally moving directly towards the goal position. JT-RRT [63] implements this concept by moving in the direction of the goal pose a certain probability of the time, p_g , by extending the random tree in the direction of the goal pose, $\mathbf{x}_g$, using the transpose of the Jacobian, J^T , from the current configuration, $\mathbf{q}_{cur}$. This biases the search to move towards the goal configuration much faster than an unbiased RRT-based algorithm. The selection for the value of p_g will determine how often

the tree extends randomly versus in the direction of the goal. This parameter can be tuned to adjust for how quickly the algorithm finds a path to the goal depending on the number of obstacles in the workspace. In this work, the JT-RTT algorithm is further optimized by combining concepts from the RRT* algorithm [59] to remap node edges if shorter collision-free paths are discovered while adding nodes to the tree. Depending on the obstacles in the workspace, this may lead to finding significantly shorter paths than simply using the algorithm presented in the literature [63].

The algorithm finds a path to the goal by continually adding nodes until a node is within a distance, d , of the goal pose. The unitless distance metric is calculated based on the Euclidean distance between positions (in millimetres) and the difference in orientation of the Euler angles (in radians).

$$d = \delta x + \delta y + \delta z + \nu(\delta r_x + \delta r_y + \delta r_z) \quad (4.69)$$

The variable ν weighs the difference in orientation to be proportional to the difference in position. This value can be adjusted to prioritize the accuracy of the position or the orientation.

Once a node is within d of the goal pose, the algorithm stops and returns the shortest path to the node. The shortest path is found by moving from the node closest to the goal pose, x_g , backwards through nodes that are further from the goal than the current node.

The structure for this implementation of the Jacobian transpose-directed optimized RRT is provided in Algorithm 7.

Algorithm 7: Structure for the JT-RTT algorithm

Input:

p_g - probability of extending tree randomly

$\mathbf{q}_i$ - starting configuration of the robot

$\mathbf{x}_g$ - goal pose of the end-effector

r - radius for nodes to be connected

k - step size/scaling parameter

d - distance threshold to goal pose

Output:

Q_{traj} - trajectory of joint positions to reach goal

```

1  $Q_{cur} = \mathbf{q}_i$ 
2 while distance to goal  $> d$  do
3    $p = U(0, 1)$  - generate a uniform random variable
4   if  $p < p_g$  then
5      $Q_{cur} = \text{ExtendRandom}(Q_{cur}, r)$ 
6   else
7      $Q_{cur} = \text{ExtendJacobianTranspose}(Q_{cur}, \mathbf{x}_g, r, k)$ 
8   end
9 end
10  $Q_{traj} = \text{FindPath}(Q_{cur})$ 

```

In the case where the tree is extended randomly, the procedure follows in the same way described in [59]. The first step is to select a random joint configuration that lies within the joint limits of the robot. Next, the nearest neighbouring node is identified, and a new node is generated along the direction of the random configuration scaled by the maximum radius, r , if it does not produce any collisions. This node is then connected to the node in its neighbourhood, r , that has the smallest path distance. After adding the new node, the tree is rewired for any neighbouring nodes whose distances can be reduced by wiring through the new node. This process is presented in Algorithm 8.

Algorithm 8: Extend random

Input:

Q_{cur} - tree of all current viable joint nodes

r - radius for nodes to be connected

Output:

Q_{cur} - tree of all current viable joint nodes

```

1  $\mathbf{q}_{rand} = \text{Random}()$ 
2  $\mathbf{q}_{near} = \text{Nearest}(Q_{cur}, \mathbf{q}_{rand})$ 
3  $\mathbf{q}_{new} = \mathbf{q}_{near} + r \frac{\mathbf{q}_{rand} - \mathbf{q}_{near}}{\|\mathbf{q}_{rand} - \mathbf{q}_{near}\|}$ 
4 if CollisionFree( $\mathbf{q}_{new}$ ) then
5   |  $Q_{cur} = Q_{cur} \cup \mathbf{q}_{new}$ 
6   |  $Q_{cur} = \text{Rewire}(Q_{cur})$ 
7 end
```

For the case where the tree extends in the direction of the Jacobian transpose, the first step is to find the joint configuration, $\mathbf{q}_{cur}$, that is closest to the goal pose. This is simple to do as the closest configuration is stored and updated as new nodes are added to the tree. Next, that node is used to calculate the current end-effector pose, $\mathbf{x}_{cur}$, and the Jacobian using the forward kinematics of the robot as outlined in Section 3.2.2. The transpose of the Jacobian is used to map the changes in the parameters $\mathbf{x}$, $\delta\mathbf{x}$, directly to the joint velocities, $\delta\mathbf{q}$. Because this calculation produces the instantaneous direction of movement towards the goal at the particular pose, it is important to only take a small step along the direction. Therefore, the change in joint configuration, $\delta\mathbf{q}$, is scaled by a constant k before adding to the current configuration. Before adding the new joint configuration to the tree it is necessary to check for collisions at this configuration and only add it to the tree if the pose is collision-free. The step-by-step process for extending in the Jacobian transpose direction is shown in Algorithm 9.

Algorithm 9: Extend Jacobian transpose

Input: Q_{cur} - tree of all current viable joint nodes $\mathbf{x}_g$ - goal pose of the end-effector k - step size/scaling parameter**Output:** Q_{cur} - tree of all current viable joint nodes

```
1  $\mathbf{q}_{cur} = \min_d(Q_{cur})$ 
2  $\mathbf{x}_{cur} = \text{ForwardKinematics}(\mathbf{q}_{cur})$ 
3  $J = \text{Jacobian}(\mathbf{q}_{cur})$ 
4  $\delta \mathbf{x} = \mathbf{x}_g - \mathbf{x}_{cur}$ 
5  $\delta \mathbf{q} = k J^T \cdot \delta \mathbf{x}$ 
6  $\mathbf{q}_{new} = \mathbf{q}_{cur} + \delta \mathbf{q}$ 
7 if CollisionFree( $\mathbf{q}_{new}$ ) then
8   |  $Q_{cur} = Q_{cur} \cup \mathbf{q}_{new}$ 
9 end
```

Following this general procedure leads to trajectories that map from the start configuration to the goal pose but does not account for cases where joint limits are reached, or collisions are detected. Therefore, it is important to add additional criteria to the algorithm to handle these cases. This is done in [63] by adding a flag to each node in the tree that indicates whether the nodes that have been extended in the direction of the Jacobian transpose lead to a collision or the limits are being exceeded for any of the joints. Once a node has this flag set, it will not be extended from again in the direction of the Jacobian transpose. Additionally, if the case arises where all nodes have the flag set, the algorithm extends randomly until this is no longer the case.

The JT-RRT algorithm requires several input parameters to be set in order to plan the trajectory. The experiments conducted used a probability of extending the tree randomly, p_g , equal to 0.05. This value was used as the robot workspace in this work is generally free of obstacles. Therefore the goal pose can be found faster with a smaller value, prioritizing moving towards the goal. In cases where there are significant obstacles, increasing the value of p_g would result in faster path planning as more randomness would be required to find a trajectory to the goal around the obstacles. The other parameters used can be seen in Table 4.6; these values provide a balance between the time to find a goal and limiting the robot's motion to small movements in order to not plan a path through an obstacle.

Table 4.6: JT-RRT parameters used in experiments.

Variable	Symbol	Value
Probability of extending tree randomly	p_g	0.05
Radius for nodes of tree to be connected [mm]	r	30
Step size	k	6×10^{-4}
Distance threshold to goal (stopping criteria)	d	20
Variable for weighing position and orientation difference	ν	10^{-3}

A motion planning validation for the JT-RTT algorithm was performed without any obstacles in the robot workspace. The robot is able to move unobstructed from its initial pose, seen in Figure 4.27ai, to a pose where the robot is approaching the red ball, seen in Figure 4.27ax. Snapshots of the robot moving through the workspace are shown in the remainder of the images in Figure 4.27a. In order to safely test the algorithm in a situation where obstacles are present along the previously unobstructed path, a virtual object was used. The virtual object was added as a plane that obstructs the obstacle-free trajectory. Real obstacles in the workspace, including the robot track, table, and platform for the red ball, were also considered in the motion plan for both tests. Snapshots of the robot moving through the workspace around the virtual object (identified as a yellow line) are shown in Figure 4.27b.

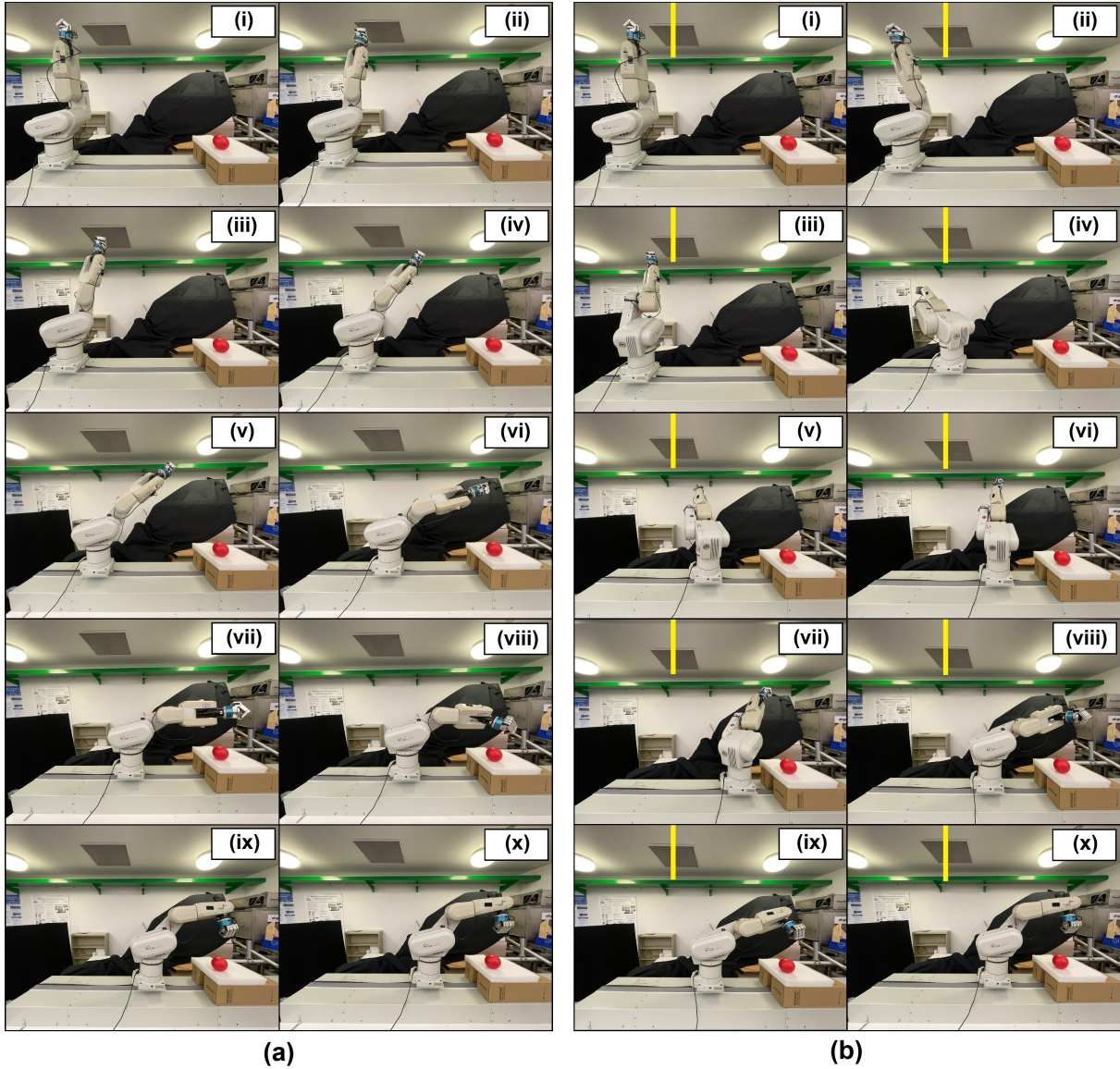


Figure 4.27: Snapshots of the robot moving from an initial pose to a grasping approach position. (a) The workspace is free of obstacles and the robot can move along a collision-free path towards to the approach pose. (b) The workspace contains an obstacle along the direct path to the approach position that the path planning algorithm must avoid as it moves to the approach position.

The joint trajectories that move the robot from the initial pose to the approach pose for both runs can be seen in Figure 4.28. In Figure 4.28a, the trajectories are mostly smooth, as they can move unobstructed towards the goal. The trajectories only move off the smooth trajectory when the algorithm moves in a random direction, and then the smooth trajectory resumes. In Figure 4.28b, the initial trajectories for all joints are quite choppy due to the algorithm extending randomly more often because moving directly towards the goal results in collisions between the end-effector and the virtual obstacle.

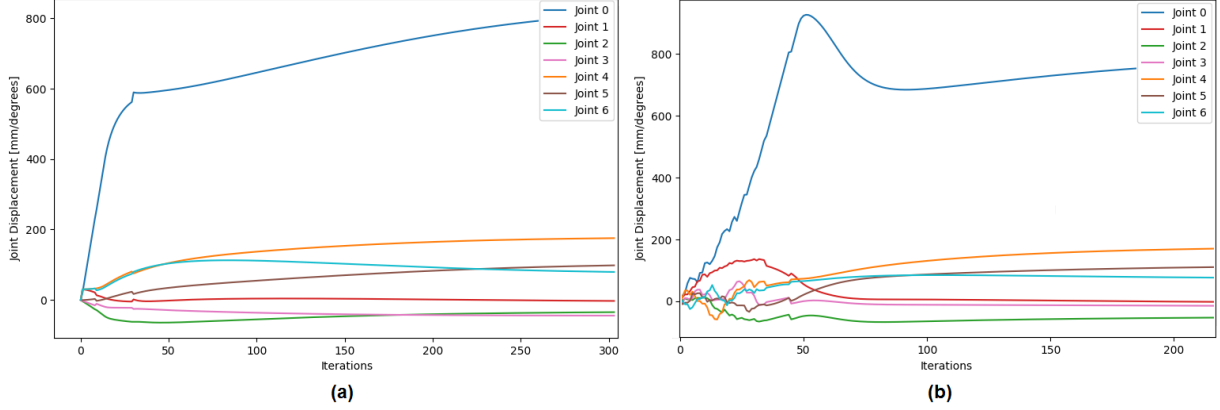


Figure 4.28: Joint trajectories for the robot manipulator moving from an initial pose to a grasping approach pose. (a) Without obstacles in the workspace and (b) with a virtual obstacle in between the initial and goal poses.

The trajectory calculated from the JT-RTT algorithm moves the robot up to within a distance d of the goal pose. Then, the robot trajectory is extended to include the exact goal pose by fixing the value for the first joint, the track, and calculating a set of solutions for the inverse kinematics of the 6-DoF robot presented in Section 3.2.2. If d is small enough, one or more solutions will always be generated and the solution that is closest to the final configuration in the trajectory is added to the path. This hybrid approach generates an exact solution for the robot to achieve the goal pose.

As well as ensuring the robot is placed in the exact desired pose, the inverse kinematic step improves the time to find a collision-free trajectory. To move to the exact pose without the inverse kinematics step, the variable d would need to be set to an arbitrarily small value. When d is extremely small, the traditional JT-RTT algorithm requires a significant amount of time to find a path to the goal pose because the extensions become increasingly small as the end-effector approaches the goal pose. Therefore it can be shown that this hybrid approach greatly increases the speed of finding a trajectory and ensures that the final configuration in the trajectory maps exactly to the desired goal pose.

A series of tests were run to assess the time to plan a collision-free trajectory and the number of iterations in the trajectory with and without the inverse kinematic step using various JT-RTT algorithm parameters. The tests planned motion for the manipulator from an initial pose, seen in Figure 4.27bi to an approach pose for grasping the red ball, seen in Figure 4.27bx. Table 4.7 shows the results for the different tests.

The results show that using a larger value for the stopping criteria distance, d , and adding the inverse kinematic step reduces the time to plan the motion and reduces the number of iterations in the trajectory. Reducing the number of iterations reduces the time for the CRS F3 robot to complete the motion plan. It can also be observed that increasing p_g leads to more random exploration of the workspace, resulting in more iterations and a longer path planning time.

Table 4.7: Time to plan path to goal using different methods and parameters.

Method	Time [s]	Iterations
JT-RTT ($p_g = 0.05, d = 20$) + IK	6.69	142
JT-RTT ($p_g = 0.05, d = 1$)	10.85	787
JT-RTT ($p_g = 0.01, d = 20$) + IK	4.05	245
JT-RTT ($p_g = 0.01, d = 1$)	6.80	566
JT-RTT ($p_g = 0.2, d = 20$) + IK	8.44	423
JT-RTT ($p_g = 0.2, d = 1$)	12.18	752

4.2.2.3 Motion Planning Pipeline

The modified JT-RTT algorithm is used to plan collision-free trajectories to the grasping poses calculated in Section 4.2.1. The motion planning pipeline starts with the robot and hand in a home configuration. The home configuration of the CRS F3 is the pose when all joint values are set to zero. The Barrett Hand’s home configuration is defined by setting the spread angle of the hand to zero and maximally closing all the fingers. This configuration for the hand minimizes the size of the required collision sphere, allowing additional collision-free poses during motion planning.

Before moving to a grasping pose, the hand is first moved to an approach pose, $T_{approach/rob}$, which is offset by a distance along the negative z -axis of the grasp coordinate system, defined in Section 4.2.1.3. The offset, z_{max} , is equal to the maximum distance that the fingers can reach along the z -axis of the hand plus a buffer to compensate for any errors in sensing the object. This ensures that the fingers can be opened at the approach pose without contacting the object to be grasped.

$$T_{approach/rob} = T_{grasp/rob} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -z_{max} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.70)$$

A motion pipeline is developed to safely and efficiently move the hand first to the approach pose and then to the grasping pose. This pipeline is illustrated in the flowchart in Figure 4.29. The pipeline starts by planning the trajectory to the approach pose using the JT-RTT algorithm to move the hand within a distance, d , of the exact approach pose. The track is then fixed, and the inverse kinematics are used to plan the exact joint positions that move the hand into the approach pose. The fingers of the hand are opened, and the fingers are spread to the angle θ_7 from the grasp point selection algorithm (Section 4.2.1.3).

Next, the JT-RTT algorithm plans the motion again to move from the approach pose to within a distance, d , of the grasp pose, $T_{grasp/rob}$. Once the trajectory is planned, the track is fixed, and the inverse kinematics are used to plan the exact joint positions that move the robot into the grasp pose. Once in the grasp pose, the fingers of the hand are moved to make initial contact with the object to be grasped. The fingers are closed using the encoder values (e_1, e_2, e_3) calculated in Equations 4.60, 4.61, and 4.62 to make contact with the surface of the object.

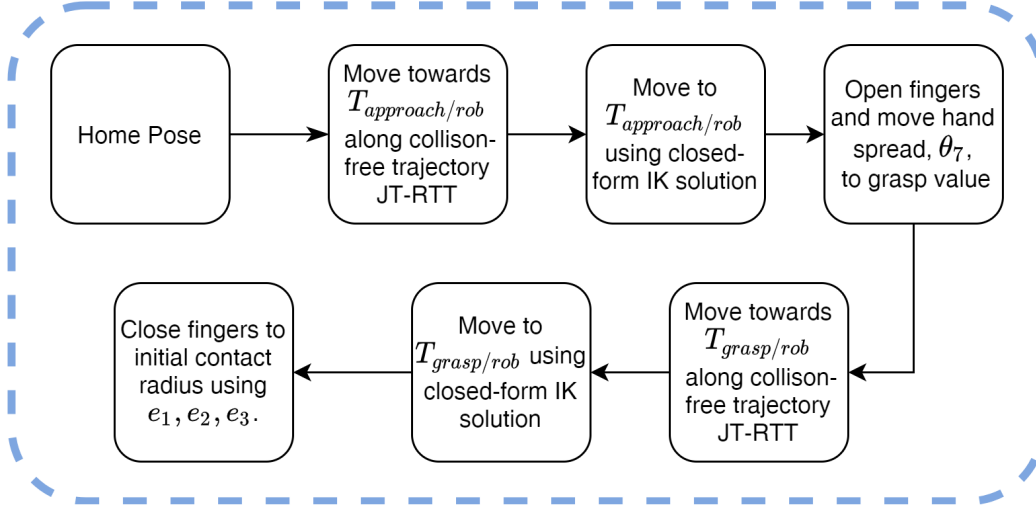


Figure 4.29: Motion pipeline flowchart for moving the manipulator from its home position to the grasping pose.

An example of the motion pipeline is shown in Figure 4.30 for one of the deformable grasping and shape control experiments. The figure shows the robot in its home pose in Figure 4.30a, in the approach pose with the hand closed in Figure 4.30b, in the approach pose with the hand opened in Figure 4.30c, and in the grasp pose with the fingers making initial contact with the object in 4.30d.

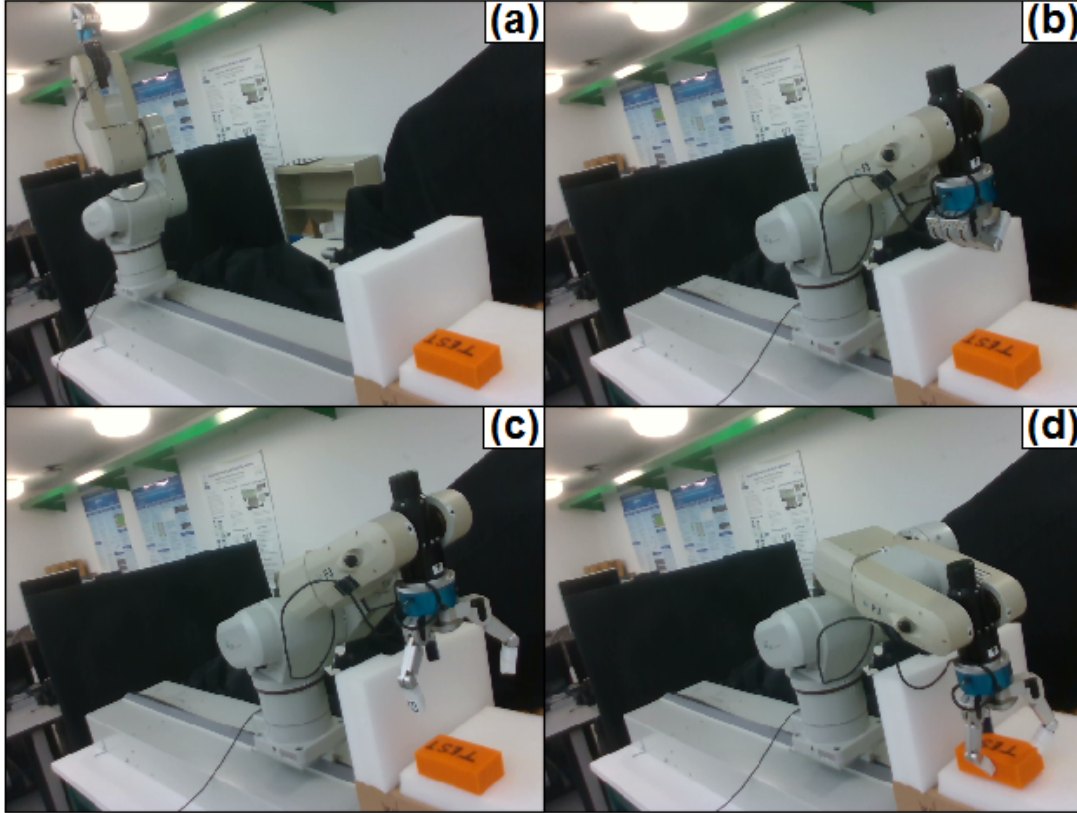


Figure 4.30: Example of the robot motion pipeline for deformable grasping and shape control. The images show the robot in its (a) home pose, (b) approach pose with fingers closed, (c) approach pose with fingers open, and (d) grasp pose with fingers at initial contact radii.

4.3 Execution

4.3.1 Shape Control

After the initial contact points are planned and the end-effector is moved to the grasping pose, the grasp is executed, and the object is deformed. The grasp's quality is assessed to ensure a stable grasp to carry out the shape control procedure. Grasp quality is assessed by comparing the force at the fingertips to an initial force measurement taken before grasping commenced. If the force applied to the fingertip has not increased from the initial measurement for a specific finger, that finger is closed until this condition is met. Once the force measurement for all fingertips is greater than the initial value, there are contact forces in opposing directions given the hand's known configuration. For most objects, the strain criteria is met as soon as contact is made. However, for some classes of deformable objects, like sponges, there is very little force created from small deformations, and occasionally significant deformation would need to occur before the condition is met. This process

completes the force closure requirement defined in this research and ensures a stable grasp on the object to be deformed.

After ensuring that the fingertips are in contact with the object, the deformation is executed by moving the fingertips towards the centre of the object by the amount defined by the difference surface image, S_{diff} , at each grasp point. This procedure optimally deforms the object towards the desired 3D shape. The new grasp radius for each finger is calculated using the desired shape's surface image, S_f (defined in Section 4.2.1.2):

$$f_{1r} = S_f(f_{1\phi}, f_{1\theta}) \quad (4.71)$$

$$f_{2r} = S_f(f_{2\phi}, f_{2\theta}) \quad (4.72)$$

$$f_{3r} = S_f(f_{3\phi}, f_{3\theta}) \quad (4.73)$$

The joint encoder value to move each finger to the radius of the desired shape's surface using the hand's inverse kinematics are calculated using Equation 3.34.

Continuing the example from previous sections, the deformed object can be seen in Figure 4.31.



Figure 4.31: Object grasping and deformation at initial contact locations.

4.3.2 Quality Assessment

Once the manipulation task of controlling the object towards the desired shape is complete, a quality assessment stage evaluates the deformed object's similarity with the desired shape.

Many of the deformation tracking and assessment techniques presented in Section 2.4.2 only use a single point of view of the object [92, 93, 104, 96], limiting the ability to assess the deformation on all dimensions or generate a complete 3D estimate of the deformed object.

Other works track the deformation of objects using feature points from RGB or RGB-D images [97, 100]. Using feature points and multiple viewpoints allows complete object estimates of the deformable object to be produced but relies on complex computer vision methods and high-resolution image data.

This work proposes a multi-view assessment strategy relying on registration from the robot encoders rather than visual information to estimate the similarity of the deformed 3D object with the desired shape. Section 4.3.2.1 discusses how optimal assessment poses are planned and how the partial point clouds obtained at each pose are registered together. Section 4.3.2.2 describes how a complete 3D estimate of the deformed object is generated and details the method for evaluating the deformed object's similarity with the desired shape.

4.3.2.1 Multi-view Assessment Poses and Registration

In the grasping pose, shown in Figure 4.31, it is difficult to assess whether the grasp and deformation are successful or not. Therefore the robot manipulator moves the deformed object into a series of assessment poses. Using multiple assessment poses allows the 3D sensor to capture the object's deformed shape from multiple perspectives, providing a more comprehensive assessment of the deformation. A series of assessment poses are defined in the camera coordinate system that places the centre of the grasped object at the centre of the images captured by the 3D sensor. The objects are positioned by the robot at a distance, d_a , from the camera. The camera coordinate assessment poses are defined as:

$$T_{c_i} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & d_a \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{for } i = 1 \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & d_a \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\frac{2\pi(i-1)}{5}) & 0 & \sin(\frac{2\pi(i-1)}{5}) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\frac{2\pi(i-1)}{5}) & 0 & \cos(\frac{2\pi(i-1)}{5}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{for } i = \{2, 3, 4, 5, 6\} \end{cases} \quad (4.74)$$

These poses define a view of the object directly in line with the negative z -axis of the end-effector and multiple views perpendicular to the z -axis of the end-effector. An example of a grasped object in each of the assessment poses can be seen in Figure 4.32. These views limit the occlusions caused by the end-effector itself, providing multiple mostly unobstructed views of the object.

The distance from the sensor, d_a , is chosen to optimize the accuracy of sensed depth data. It has been shown that IR stereo depth sensors are most accurate when the object being sensed is as close to the sensor as possible, but still further away than the minimum sensing distance [106, 118]. The distance was chosen using an iterative process, increasing d_a from the minimum value until an inverse kinematic solution for each assessment position was found. To determine whether the inverse kinematic solution is possible, the assessment poses are transformed to the robot coordinate systems

$$T_{a_i} = T_{cam/rob}T_{c_i} \quad (4.75)$$

Since there is no closed-form solution for the inverse kinematics, solutions are checked at each step for increasing values of the track, D_0 . Once an inverse kinematic solution is found, the joint values, $\mathbf{q}$, are checked for collisions with the environment. This process continues until a pose with a valid inverse kinematic solution is also collision-free. Optimal assessment poses are determined for each object prior to it being grasped, as the poses will vary depending on the obstacles in the environment. The process for finding the optimal assessment poses is outlined in Algorithm 10.

Algorithm 10: Find optimal assessment poses

Input:

T_{c_i} - assessment pose i in the camera coordinate system

$T_{cam/rob}$ - transformation matrix from camera coordinate system to robot coordinate system

Output:

T_{a_i} - optimal valid assessment pose i in the robot coordinate system

```

1 for  $d_a \leftarrow \min(d_a)$  to  $\max(d_a)$  do
2   for  $D_0 \leftarrow \min(D_0)$  to  $\max(D_0)$  do
3      $T_{a_i} = T_{cam/rob}T_{c_i}$ 
4      $\mathbf{q} = \text{InverseKinematics}(D_0, T_{a_i})$ 
5     if  $\mathbf{q}$  exists then
6       if CollisionFree( $\mathbf{q}$ ) then
7         return  $T_{a_i}$ 
8   end
9 end
```

The robot manipulator plans collision-free paths to each of the assessment poses using the path planning solution described in Section 4.2.2. At each assessment pose, colour and depth images are captured, and the grasped object is segmented from the background using the fast level set technique from Section 4.1.2.3. In this phase, the centre of the object is known, and the algorithm is run without including any boundary edge conditions ($\gamma = 0$). The object presented at each assessment pose is shown in Figure 4.32, with the object segmentation contour displayed in the colour image.

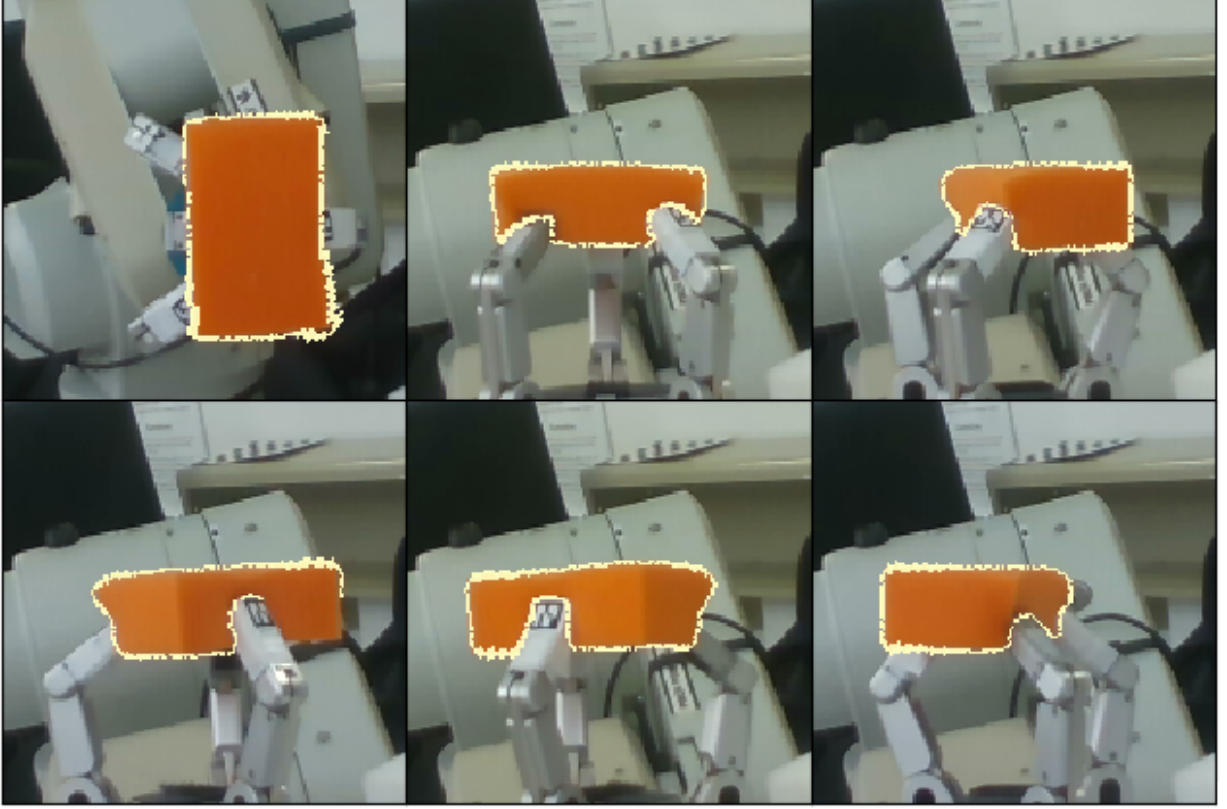


Figure 4.32: Deformed object at each assessment pose, defined in Equations 4.74, with the segmentation contour identified.

The segmentation mask is used to generate a point cloud from the observed portion of the object in each image. The partial point clouds are aligned through a registration process to get a complete estimate of the deformed object's shape. The registration is done using the known pose of the robot at each assessment position. Relying on the accuracy of the robot's encoders allows registration to be completed without using the lower-accuracy data collected from the 3D sensor.

Registration for the data collected at the i^{th} assessment pose is done through a series of coordinate system transformations using the transformations calculated in previous steps and the known assessment pose. The first step is to transform the partial point cloud generated by the segmentation mask from the camera coordinate system to the robot coordinate system. The points in the camera coordinate system, $(x_{cam_i}, y_{cam_i}, z_{cam_i})$, are multiplied by $T_{cam/rob}$, calculated in Section 4.1.1, to generate points in the robot coordinate system. From the robot coordinate system, the points can be transformed to the current end-effector coordinate system using the inverse of the assessment pose, $T_{a_i}^{-1}$. These two transformations align the data in the end-effector coordinate system. A final transformation is done to align the deformed object's point cloud with the undeformed object in order to facilitate assessing the quality of the deformation. This step aligns the deformed object with the undeformed object and the desired shape following the object setup described in Section 4.2.1.2. This transformation is done using the grasping pose, $T_{grasp/rob}$, calculated

from Equation 4.64. Following this procedure the deformed object point cloud $(x_{d_i}, y_{d_i}, z_{d_i})$ is calculated as follows:

$$\begin{bmatrix} x_{d_i} \\ y_{d_i} \\ z_{d_i} \\ 1 \end{bmatrix} = T_{grasp/rob} \cdot T_{a_i}^{-1} \cdot T_{cam/rob} \cdot \begin{bmatrix} x_{cam_i} \\ y_{cam_i} \\ z_{cam_i} \\ 1 \end{bmatrix} \quad (4.76)$$

The above equation transforms the sensed data points at the i^{th} assessment pose to the robot coordinate system, with the points centred at the estimated undeformed object's centre. Once all assessment poses are analyzed and the coordinates of the data are transformed, the partial point clouds are merged to a single point cloud. This point cloud represents the visible portion of the deformed object from the multi-view assessment.

4.3.2.2 Deformation Evaluation

To evaluate the quality of the deformation a complete 3D object must be estimated from the point cloud obtained in Section 4.3.2.1. The known fingertip positions are added to the deformed object point cloud to improve the quality of the data. As it is assumed the fingertips are in contact with the object at the grasping points, additional points can be added to the deformed object point cloud that were occluded by the fingertips in the initial data. The encoder values and forward kinematics of the arm and hand map the position of each fingertip during the grasp. Points at these positions are added to the deformed object's point cloud to reduce the amount of missing data occluded by the hand. Some occlusions can be filled in through the forward kinematics, but there are also occlusions caused by the hand where it is not in contact with the object, and the occlusions cannot be filled in.

There are many sources of error in the estimate of the deformed object. Two challenges faced by an object deformation task are compounded to produce significant errors in the results. Firstly, the error from the sensor is significant, making it quite difficult to accurately assess how the deformed object compares to the desired 3D shape. This is compounded by the occlusions caused by the end-effector, which in this case significantly impacts the quality of the data.

In order to get a complete estimate of the deformed object, the parts of the object that are still occluded are interpolated using bilinear interpolation on a grid in spherical coordinates. This is the same method that was applied in Section 4.2.1.2 using Equations 4.36 and 4.37. After interpolation the data is smoothed using Gaussian filter to reduce the noise from the sensor and smooth any errors in registration from the robot encoder positions.

Figure 4.33 shows the point cloud of the deformed object from Figure 4.32 with the fingertip positions identified.

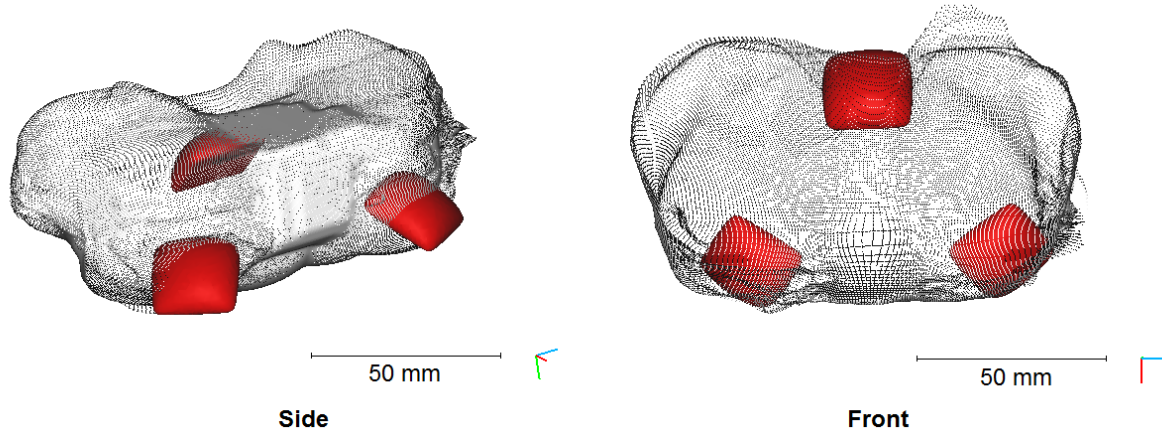


Figure 4.33: Two views of the complete point cloud from multi-view assessment registration, with fingertip positions shown in red.

After interpolating the deformed object, the visualization software Cloud Compare [119] is used to visually and quantitatively assess the deformation. The first step in assessing the deformation is to generate a surface for the deformed object point cloud using a Poisson surface reconstruction [120]. The algorithm reconstructs a triangular mesh from a set of oriented 3D points by solving a Poisson system. The reconstructed surface enables the deformed object to be visually assessed and compared to the desired object.

The undeformed object and the desired 3D shape are also imported into the Cloud Compare software to assess the results before and after deformation. The point cloud of the measured undeformed object after shape completion, calculated in Section 4.1.3, is imported and converted to a mesh using a Poisson surface reconstruction. The desired shape that was developed in the ANSYS software is imported as a mesh model.

The deformation is assessed quantitatively by calculating the average unsigned distance from each element of the deformed object's reconstructed surface to the nearest point on the desired object's surface. This distance is defined as the C2M distance [119]. The C2M distance is also calculated between the undeformed object and the desired object's surface. Comparing the two C2M values indicates how much closer, or further away, the deformed object is to the desired object than the initial undeformed object to the desired object. This metric is used to determine the quality of the shape control and deformation.

Continuing the example from Figure 4.33, Figure 4.34 shows the desired object, the deformed object and the initial undeformed object from two viewpoints. The deformed and undeformed objects are shown with the C2M distance displayed on the surface of the object.

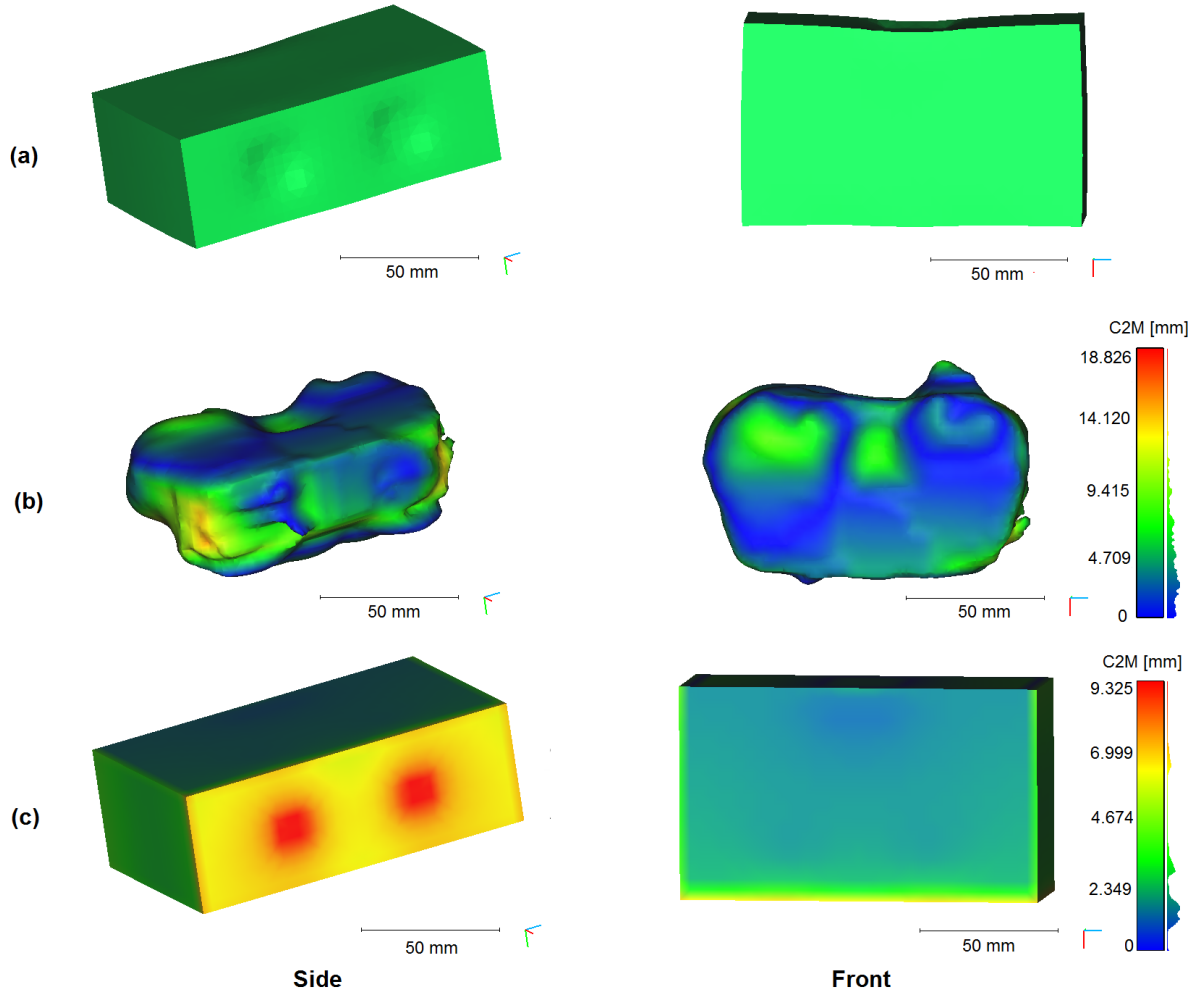


Figure 4.34: Visual assessment of deformation and shape control for the sample object presented in Figure 4.31. Side and front views of the (a) desired shape (from Figure 4.13b), (b) measured deformed object with C2M distance from desired shape shown on surface, and (c) measured undeformed object with C2M distance from desired shape shown on surface.

Observing Figures 4.32, 4.33, and 4.34, it can be seen that the contact points on the object were accurately chosen and the manipulator succeeded in placing the fingertips at the contact locations.

It is clear that the contact points and amount of deformation are good candidates to deform the original object to the desired shape. During the assessment of the deformed object, the sensor error makes it difficult to accurately assess the quality of the deformation when the changes to the object shape are small. In Figure 4.34, the deformed object has a mean C2M of 4.53 mm compared to the desired shape. By comparison, the undeformed object has a mean C2M of 2.41 mm when compared to the desired shape. Even though the deformation contact points are chosen correctly, and the hand is placed at the correct position to deform the object at these points, the error in the initial undeformed object

compared to the desired object is smaller than the deformed object compared to the desired object. Observing Figures 4.33 and 4.34, it is clear that much of the error is a result of the inaccuracies of the sensed data. Because the desired object was generated from applying deformations to an object model that matched the size of the real object being used, the C2M error is small. This example shows the difficulty in assessing small deformations with the sensor used for this research.

In addition to using desired objects that are deformed from models of the actual objects, this research tested grasping and shape control from original object shapes towards another shape entirely. These tests require significantly larger deformations in order to deform the initial object towards the desired shape. Figure 4.35 shows the estimated undeformed object and the desired shape to deform towards for one of these cases.

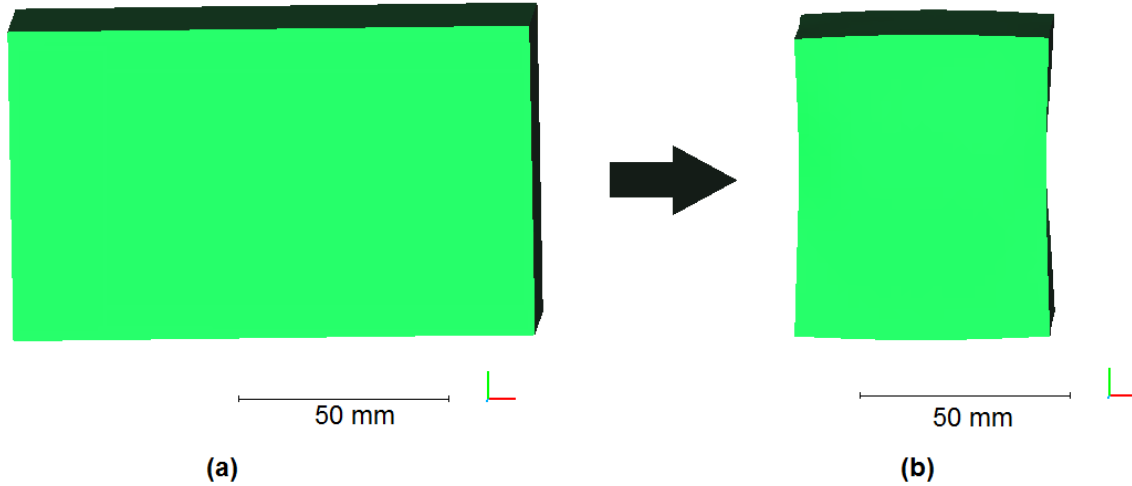


Figure 4.35: (a) Estimated initial object shape and (b) desired shape requiring significant deformation.

As shown in Figure 4.36, the object is grasped successfully using contact points calculated in Section 4.2.1. It is moved to the assessment positions where the deformed object is segmented from the background. The single-view point clouds are registered together to generate a single point cloud of the deformed object. The grasped object is shown at each assessment pose with the segmentation contour identified in Figure 4.36a. The registered point cloud of the deformed object with fingertip positions is shown from two viewpoints in Figure 4.36b.

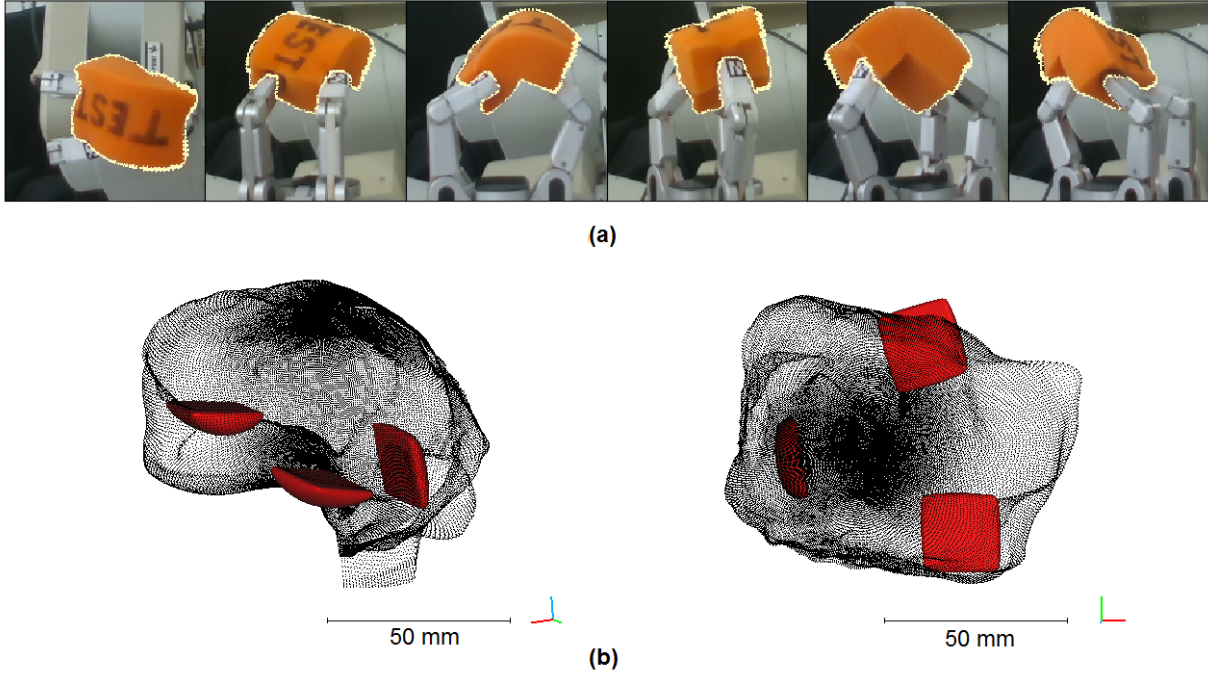


Figure 4.36: (a) Object presented at all assessment poses with segmentation contour identified and (b) complete point cloud of deformed object with fingertips positions from two perspectives.

Using the Cloud Compare software, the surface of the deformed object is reconstructed. The error between the deformed object and the desired shape is calculated at each point on the surface of the deformed object. The error between the undeformed object and the desired shape is also calculated. Results are illustrated in Figure 4.37.

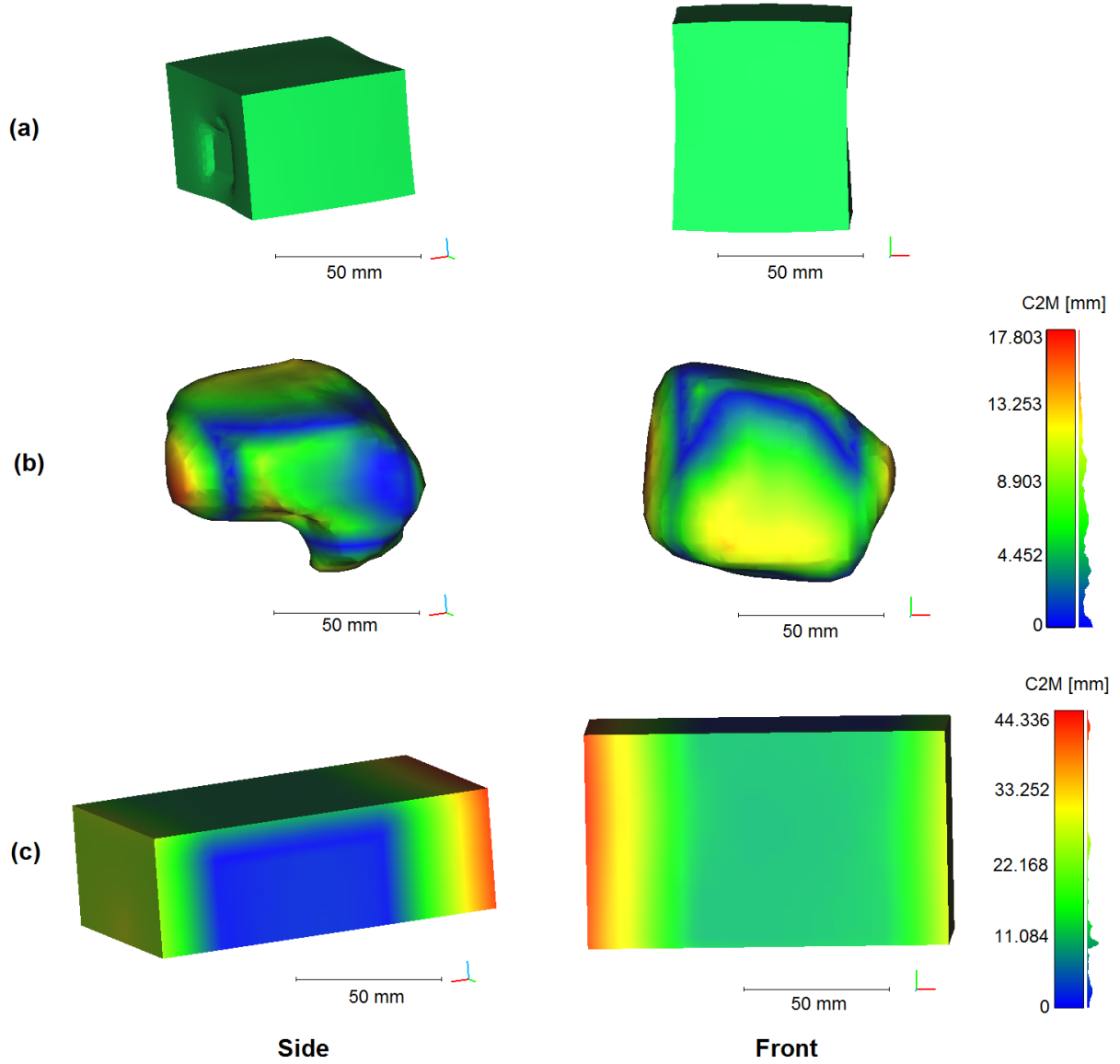


Figure 4.37: Visual assessment of deformation and shape control for desired shape requiring significant deformation. Side and front views of the (a) desired shape, (b) measured deformed object with C2M distance shown on surface, and (c) estimated undeformed object with C2M distance shown on surface.

The mean C2M distance between the deformed object and the desired object is 6.2 mm. The mean C2M distance between the undeformed object and the desired object is 16.4 mm. The C2M distance is a measure of dissimilarity between two shapes being assessed. Therefore, the C2M values for the deformed object and the undeformed object can be used to determine the change in similarity with the desired shape after deformation. Equation 4.77 defines a metric to measure improving similarity of the undeformed object with the desired shape:

$$\Delta = \frac{C2M_{\text{undeformed}} - C2M_{\text{deformed}}}{C2M_{\text{undeformed}}} \times 100 \quad (4.77)$$

A positive value of Δ indicates the deformed shape is more similar to the desired shape than the undeformed shape. If Δ is negative, it indicates the deformed shape is less similar to the desired shape than the undeformed shape.

The C2M values presented above for the shapes in Figure 4.37 results in a 62.2% increase in the similarity of the deformed object shape with the desired shape compared to the undeformed object shape with the desired shape.

Although the hand is able to improve the similarity of the initial undeformed object substantially it is still far from perfect. It can be seen that the sponge does not deform exactly as expected. There is a clear bend of the sponge, evident in Figures 4.36 and 4.37, rather than the desired compression. This may be due to material properties of the object or inaccuracies in the fingertip position or deformation direction. This example highlights the complexity of the problem of shape control for 3D deformable objects.

4.4 Summary

This chapter develops the proposed robotic solution for grasping and shape control of 3D deformable objects. Several original contributions were developed throughout the chapter by extending previous relevant work presented in Chapter 2. The system is broken up into a series of modules, each of which is developed alongside supporting experimental data.

The solution begins with an initialization phase that uses visual information from colour and depth images captured from a commercial 3D sensor to identify and segment an object within a user-defined grasping area. The segmentation is performed using an original implementation of the fast level sets algorithm that segments based on colour image data and a boundary probability estimate calculated from a depth image. The segmentation algorithm is able to overcome objects of various sizes that contain a variety of colours and textures. The segmented data is then used in an object completion algorithm to generate 3D estimates of primitive shapes from the single view depth data. This process generates 3D estimates of undeformed objects that are required for grasp and shape control planning.

In the planning phase, initial contact points are identified to grasp the undeformed object and control it to a desired 3D shape. This research presents the first true 3D contact point selection algorithm using an original object representation. An original surface image model allows for grasp point selection to be performed in a finite degree of freedom representation while also taking into account the kinematic constraints of the robotic hand. The system also plans collision-free paths for the end-effector to be placed in the optimal valid grasping pose by the 7-DoF robot manipulator by implementing a unique combination of motion planning techniques with the kinematic model of the robot developed in this work.

The execution and assessment of the grasp and deformation use force values at the fingertips to ensure the fingers' contact forces are working against each other to secure the

object. Once the fingers are on the object, the deformation is performed to control the object's shape towards the desired shape. After deformation, the object is moved to the calculated optimal assessment poses. At the assessment poses, the object is segmented again, and the partial point clouds from six different views of the object are registered together to generate a full 3D point cloud of the visible areas of the object. A 3D surface reconstruction is performed using the registered point cloud data and the fingertips' known positions. The reconstructed surface is used to assess the quality of the deformation by comparing the similarity of the deformed object to the desired shape against that of the undeformed object to the same desired shape.

The robotic system for deformable grasping and shape control presented in this research finishes after assessing the quality of the deformed object. Complete end-to-end results for a variety of experiments are presented in Chapter [5](#).

Chapter 5

Experimental Results and Analysis

5.1 End-to-End System Results

The main objective of the research presented in this thesis is to design a system to control the shape of a previously unseen three-dimensional deformable object to a desired shape using a robotic arm equipped with a three-fingered gripper. Chapter 3 described in detail the general framework of the system, the equipment used in this research, and the objects used in the experiments. Chapter 4 described in detail the algorithms and methodologies used to complete the research objectives. Several original contributions were presented in the development of the solution.

Experimental validation of the system framework first presented in Figures 3.1 and 3.2 was completed, demonstrating the viability of the proposed autonomous robotic deformable object grasping and manipulation system.

The system diagram in Figure 3.4 shows all of the components and how they interact to complete the objectives. The experiments were developed in Python using previously developed libraries to communicate with the Intel RealSense camera and the Barrett Hand. A library was developed to communicate with the CRS F3 robot through the serial port to control the robotic arm's motion. Some precompiled libraries (NumPy, SciPy, OpenCV) were used for implementing known techniques. However, all of the major algorithms described in Chapter 4 were developed for the completion of this work. The desired deformed shapes were created in ANSYS [114] and the deformation assessment was performed using the Cloud Compare software [119].

The system is designed to autonomously control the shape of an unknown object, but it does rely on the exogenous inputs, first presented in Figure 3.2. These include the coordinate system transformation, RGB and depth images of the workspace, size and location of obstacles in the workspace, the desired 3D shape of the object, kinematics of the hand and arm, and a user input grasping area in the robot workspace in which to search for objects. A series of experiments were designed using the required exogenous inputs.

For each experiment, results from all components of the system are presented. In the figures that follow the labelled subfigures represent the following:

- Initialization
 - (a) The robot in its home position with the user-defined grasp area identified and the segmented object.
 - (b) The partial 2.5D point cloud generated from the segmentation mask.
 - (c) The estimated 3D object completion of the initial undeformed object.
 - (d) The desired 3D shape to deform the object towards.
- Planning
 - (e) The difference surface image between the initial undeformed object and desired 3D shape with candidate grasp points and optimal valid grasp points identified.
 - (f) The joint trajectories for a collision-free motion plan to bring the robot and end-effector to the optimal valid grasping pose.
- Execution
 - (g)-(j) Images of the robot and hand at the approach and grasping poses.
 - (k) The segmented object at each assessment pose.
 - (l) The desired 3D shape to deform the object towards.
 - (m) The deformed object point cloud with finger positions identified.
 - (n) The deformed object surface reconstruction with the C2M distance to the desired 3D shape identified.

Various object positions were chosen to validate the system’s ability to grasp objects throughout the robot’s workspace. Additionally, a variety of obstacles were introduced to the workspace to validate the path planning and grasp selection algorithms. This required the robot to reach over, around or underneath obstacles to grasp and deform the objects. Objects were placed in traditional tabletop settings and also tested attached to walls and suspended from the ceiling to show the generality of the system. The system was able to plan grasps and collision-free motion to bring the end-effector to the optimal valid grasping pose in each experiment. In total, 14 different experiments were run using the objects identified in Section 3.3. The remainder of this section presents the results of a subset of these experiments. Experiments not analyzed here are summarized in Appendix A. Videos of all of the experiments can be accessed at: https://www.youtube.com/channel/UCak01Y6gjy_SbNVtcca8Ekw.

5.1.1 Experimental Test Case 1

Figure 5.1 shows the robotic system’s initial setup to grasp and deform a rubber ball placed in a grasping area. The figure illustrates the user-selected grasping area to search for an object and the resulting object segmentation from the algorithm described in Section 4.1.2. The ball is placed behind an obstacle on a flat surface, presenting obstacles that the grasping and path planning algorithms must consider. The size and locations of obstacles are input to the system by a user for all experiments in this thesis. Automatic obstacle detection is a possible area for future work (see Section 6.3).

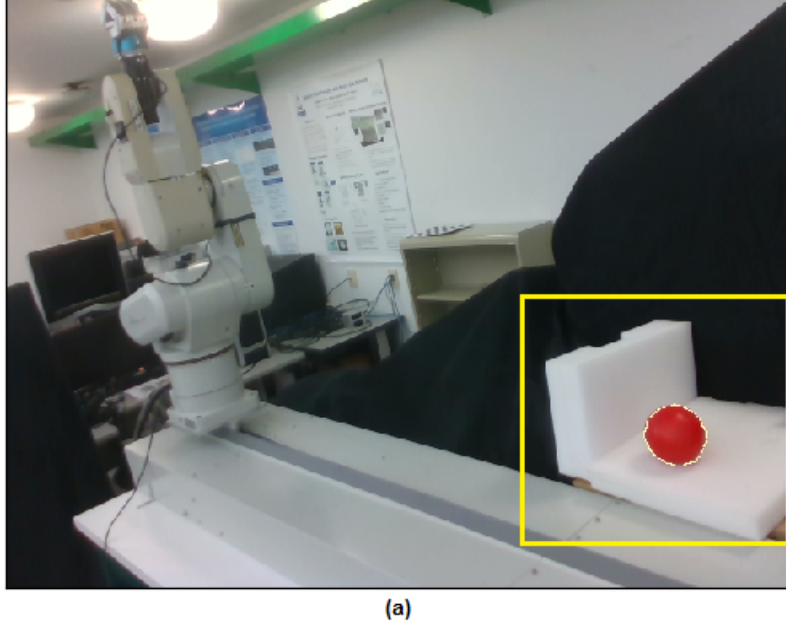


Figure 5.1: Initial setup for grasping and deforming a rubber ball.

After segmenting the object from the background, the system performs object completion on the 2.5D partial point cloud generated using the segmentation mask, following the procedure outlined in Section 4.1.3. The partial point cloud is presented in Figure 5.2b, and the best shape completion for the data is shown in 5.2c. In this case, the size and shape are accurately identified, and the estimated complete 3D geometry of the undeformed object is used for grasp planning. In this example, the desired 3D shape is a sphere with two deformations applied to the surface, shown in Figure 5.2d, which leads to the distance from the centre of the sphere being slightly smaller on the left-hand side of the desired shape model.

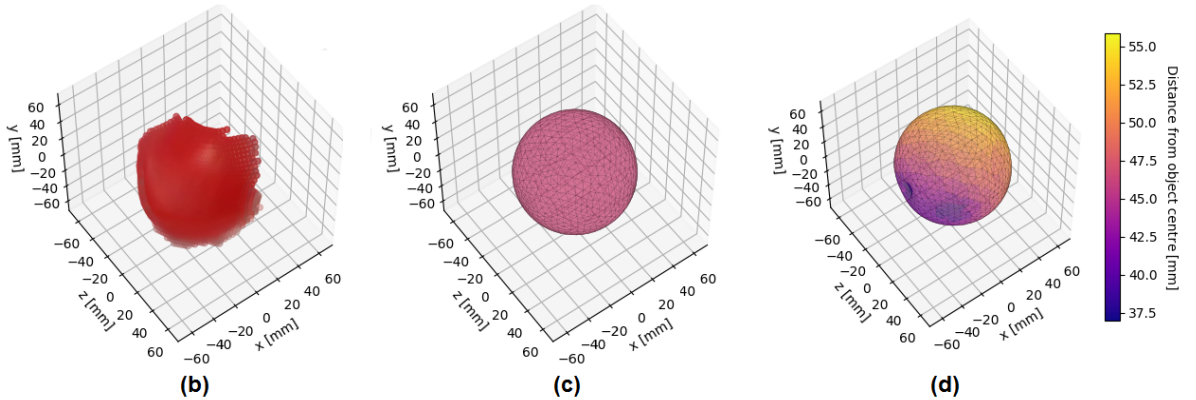


Figure 5.2: Results from initialization phase of robotic grasping system for rubber ball experiment. (b) Partial point cloud of object from segmentation mask. (c) Estimated 3D shape of undeformed object. (d) Desired 3D shape to deform the object towards with distance from object centre illustrated.

The difference surface image between the undeformed and desired objects, shown in Figure 5.3e, illustrates these deformations as areas with significant differences between the shapes. The grasp planning algorithm presented in Section 4.2.1 identifies all candidate grasp points and the optimal grasp points that lead to a valid grasping pose. The path planning algorithm described in Section 4.2.2 plans a collision-free motion plan to guide the end-effector to the approach and grasping poses, avoiding the obstacles placed in the workspace in the same way shown in Figure 4.27b. The joint paths are shown in Figure 5.3f. The joints generally follow smooth trajectories as the end-effector moves towards the object. Breaks from the smooth trajectories are evident when the algorithm explores the workspace randomly to plan a path around any obstacles in the workspace.

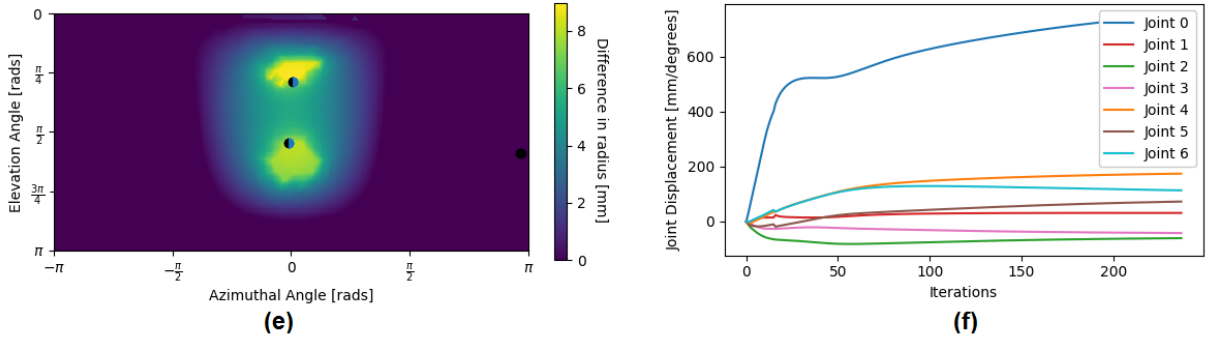


Figure 5.3: Results from the planning phase of robotic grasping system for the rubber ball experiment. (e) The difference surface image between the undeformed object and the desired shape with candidate grasp points (blue), chosen grasp points (black) and chosen candidate points, p^A and p^B (black/blue). (f) Joint trajectories to move the end-effector from the home pose to the optimal valid grasp pose.

After planning is complete, the robot moves to the approach pose, opens the fingers, and moves to the grasping pose. The end-effector grasps the object using the encoder values that move the fingertips to the object's surface. At this point, the force sensors in the fingertips are used to ensure the fingertips are in contact with the object before continuing with the shape control procedure. If the fingertips are not in contact with the object due to inaccuracy in sensing the object or because of the simplification made to accommodate the kinematic constraints, any finger that is not in contact is closed until the force sensors detect contact with the object. Next, the fingers deform the object towards the desired shape using values from the difference surface image. A sequence of images of the robot moving through the motion planning pipeline can be seen in Figure 5.4.



Figure 5.4: Images of the robot as it moves from the approach pose into the grasping pose and deforms the object. (g) Robot in approach pose with hand closed, (h) and open. (i) Robot in the grasping pose with fingers in contact with surface of the object, (j) and after deforming the object towards the desired shape.

The robot moves the object to the assessment poses defined in Section 4.3.2 to evaluate the quality of the deformation. The segmentation at each assessment pose is shown in Figure 5.5k. The partial point clouds from each assessment pose are registered together to generate the complete point cloud of the deformed object, shown in Figure 5.5m. The point cloud is used to generate an estimated surface of the object using a Poisson reconstruction. The reconstructed surface is shown in Figure 5.5n, with the distance between each point of the deformed object and the desired object's surface shown on the surface of the object.

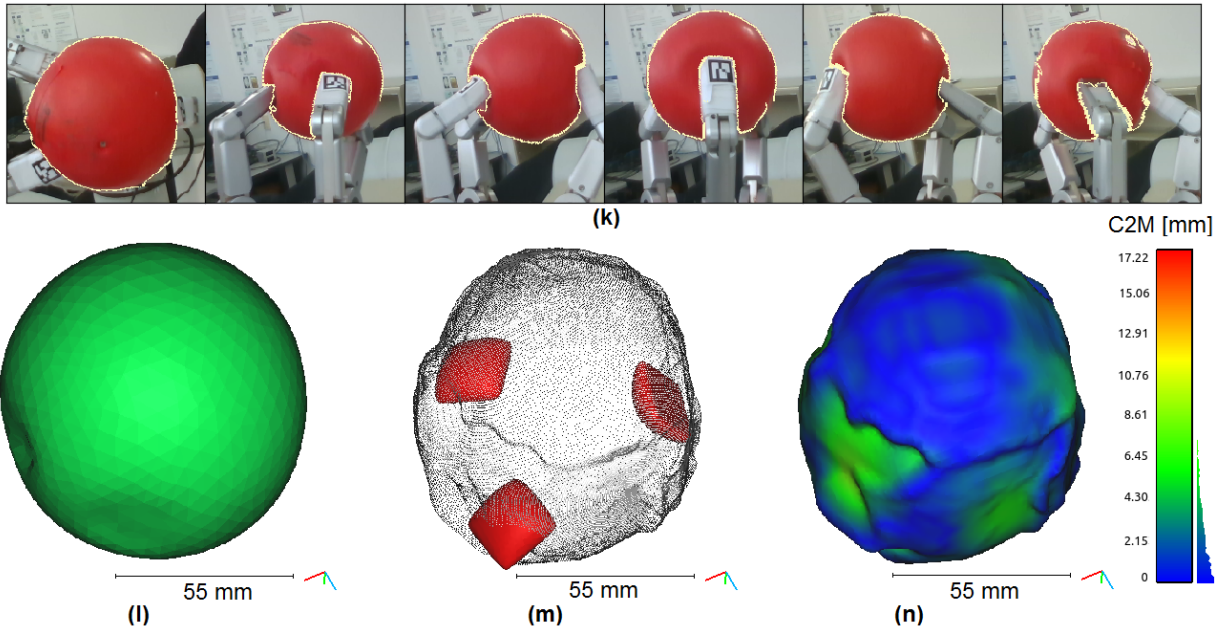


Figure 5.5: Results from the execution phase of robotic grasping system for the rubber ball experiment. (k) Segmented deformed object at multiple assessment poses. Views from the same point-of-view of the (l) desired 3D shape, (m) registered point cloud with fingertip positions identified, and (n) reconstructed surface of the deformed object with distance from the desired shape identified.

It is evident from Figure 5.3e that the grasping point selection algorithm chooses the correct points to deform the object towards the desired goal, choosing points that have the largest values in the difference surface image. The exact positions are slightly off, evident observing Figures 5.5l and 5.5m. This is due to a combination of sensor error in estimating the exact position of the object and the hand’s kinematic constraints. Quantitatively, the deformation does not improve the similarity between the deformed object and the desired shape compared to the undeformed object and the desired shape. The mean C2M distance for the deformed object to the desired shape was 2.86 mm compared to 0.85 mm for the undeformed object to the desired shape. This is due to the fact the deformation is small, meaning the undeformed object is already extremely similar to the desired object. Any improvement in the shape of the deformed object towards the desired shape is offset by the sensor error when assessing the results. The limited accuracy of the sensor can be observed on the deformed object’s surface reconstruction as the appearance of waves on areas that would be expected to be smooth. Error in registration due to the accuracy of the robot’s encoders can also be seen in the registered point cloud, where areas do not line up perfectly, resulting in slight discontinuities on the surface.

5.1.2 Experimental Test Case 2

The experiments included cases where the object is not placed on a surface. Instead, it is attached to a vertical plane or hung from the ceiling. Figure 5.6 illustrates a case where the rubber ball from the previous example is fixed to a vertical obstacle in the workspace. The figure highlights the user-defined grasping area and the initial segmentation of the object.

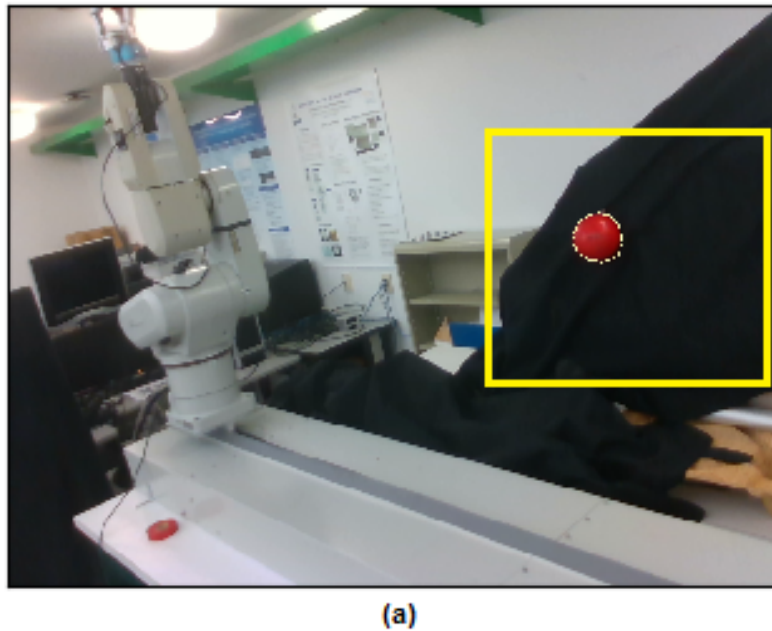


Figure 5.6: Initial setup for grasping and deforming a rubber ball fixed to a vertical surface.

The desired shape control task in this experiment was to deform the spherical ball to a rectangular box shape. The estimated undeformed object and desired 3D shape to deform towards are shown in Figures 5.7d and 5.7c. The contact point selection algorithm plans contact points on the top and bottom of the object to deform the sphere towards a box. A valid grasp is identified, and the path planning algorithm finds a collision-free path to place the end-effector in the optimal pose.

After deformation, the mean C2M distance for the deformed object to the desired shape was 8.86 mm compared to 12.80 mm for the undeformed object to the desired shape. This represents an improvement of 31% in the similarity of the deformed object and the desired 3D shape. This result is illustrated visually in Figure 5.7n, where the ball appears significantly more box-like than the undeformed object.

This is a considerable improvement in the overall shape and highlights the system’s ability to automatically plan a manipulation that successfully deforms an object towards the desired input shape. That being said, the results also highlight some minor limitations of the system. In Figure 5.7k, it can be seen that the tops of the fingertips of the robot hand are in contact with the ball, rather than the fingertips themselves. The algorithm plans for contact with the fingertips but as the ball is being deformed, the object rotates, and the contact points slide to the tops of the fingertips. There is no implemented method to handle this, so the reconstruction is still done by adding the fingertip locations to the estimated deformed object. The result is incorrect points being added to the point cloud of the deformed object impacting the reconstructed shape and the quality of deformation estimate. Visually this can be observed in Figure 5.7m, where the two fingertips on the right of the object should be oriented differently to accurately represent the true grasp.

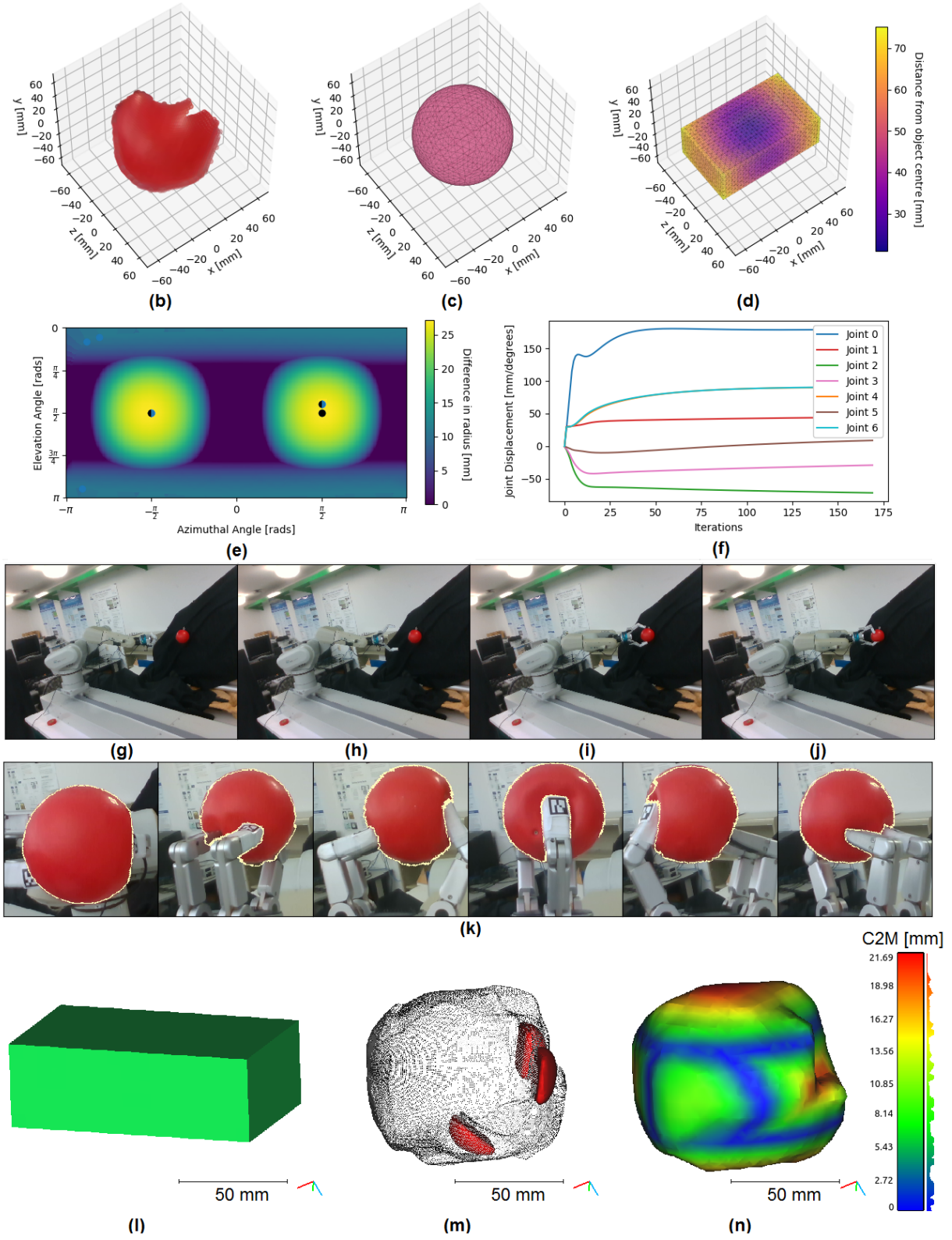


Figure 5.7: End-to-end results for grasping and deforming a rubber ball fixed to a vertical surface.

5.1.3 Experimental Test Case 3

Experiments that involved complex shape control problems where the manipulator is severely underactuated to deform the object to the desired shape were also considered. Figure 5.8 shows in the initial setup for an experiment to deform a rectangular sponge into the shape of a heart (Figure 5.9d).

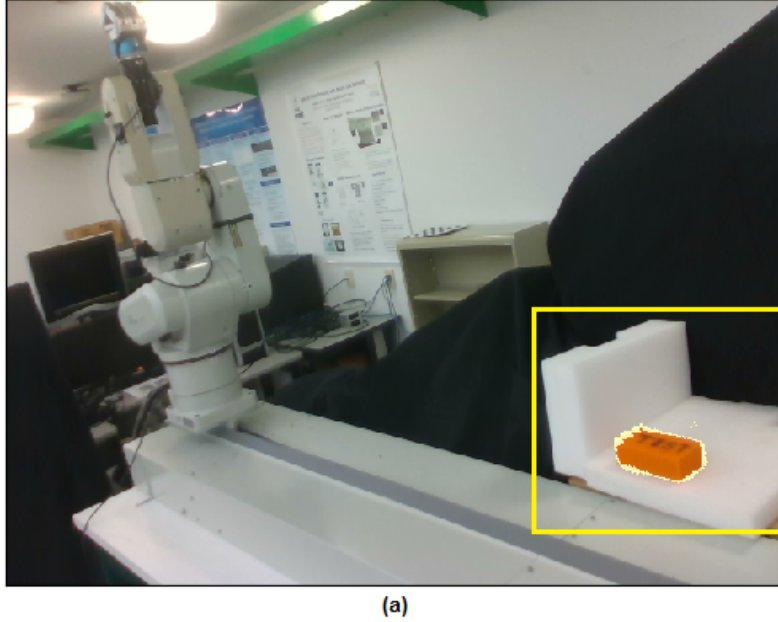


Figure 5.8: Initial setup for grasping and deforming a sponge.

Due to the complexity of the shape control task, many potential grasp points are identified on the undeformed object's surface. All candidate points and the optimal valid grasp points (in blue/black) given the obstacles in the environment are shown in Figure 5.9e. The deformed object is visualized in Figure 5.9n. The deformed 3D shape is not overly similar to the desired shape, with an average C2M distance of 15.23 mm, highlighting the complexity of the problem. Although the similarity is not overly high, there is still a significant improvement from the undeformed shape, which had a mean C2M distance of 19.99 mm, resulting in a 24% improvement.

The complexity of this task is so high that even a human cannot perform significantly better than the robot. When the shape control task was attempted, the anecdotal results were similar to those achieved by the robot. This was not validated experimentally as it is outside of the scope of the research but would represent an interesting area to explore in future work (see Section 6.3).

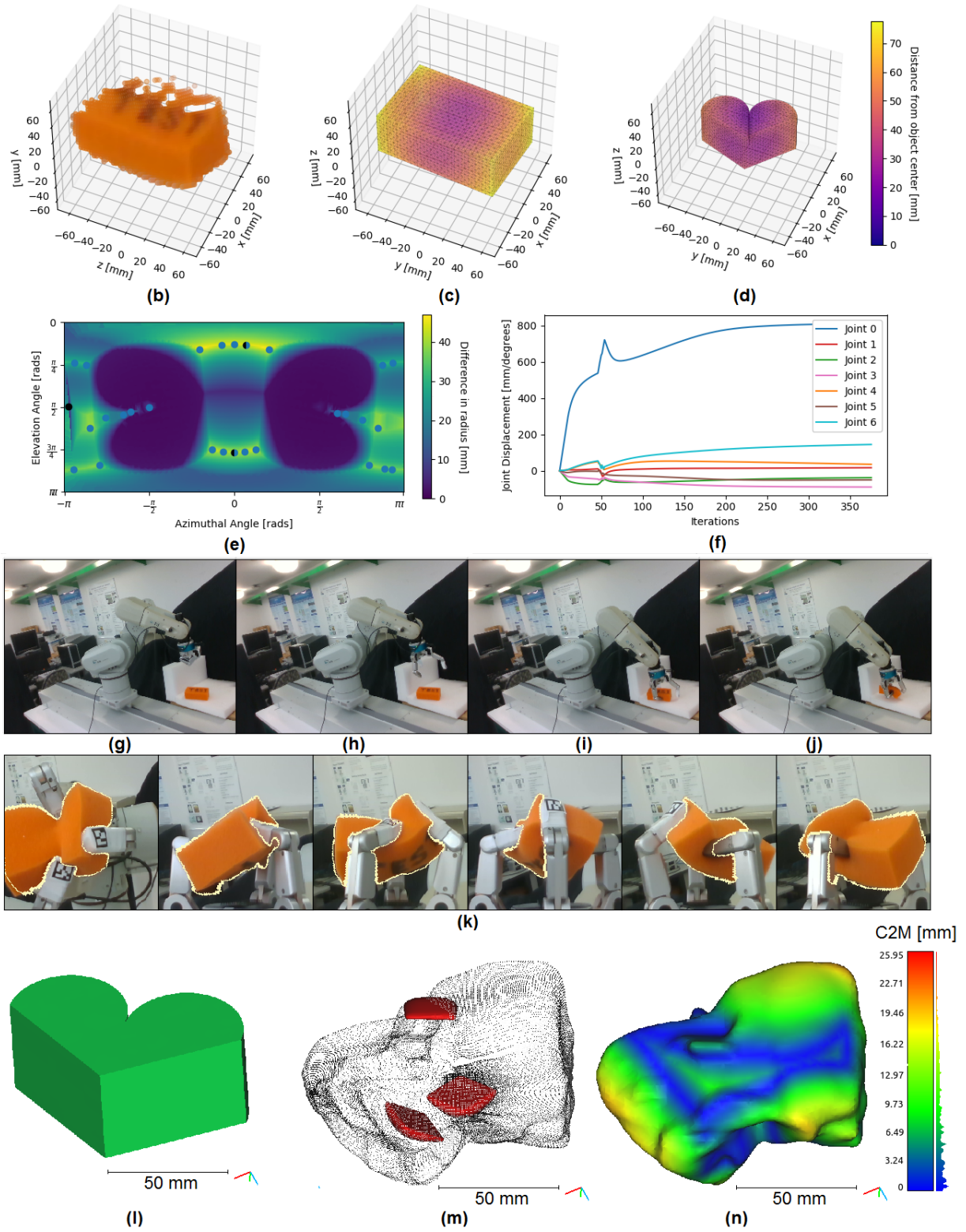


Figure 5.9: End-to-end results for grasping and deforming a sponge placed on a tabletop surface behind an obstacle.

5.1.4 Experimental Test Case 4

Figure 5.11 shows the initial setup for deforming a foam cylinder towards an hourglass shape (Figure 5.11d).

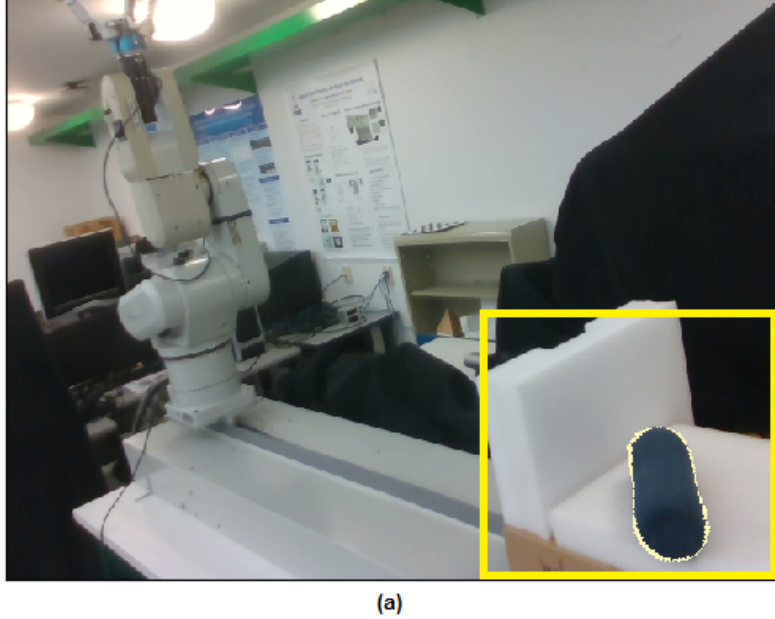


Figure 5.10: Initial setup for grasping and deforming a foam cylinder.

Optimal grasp points were planned to deform the cylinder toward the shape of an hourglass given the obstacles in the environment. After executing the collision-free path, the robot attempted to grasp and deform the object, but as the fingers made contact with the object, it rotated rather than deforming as expected. This can be seen in Figures 5.11i and 5.11j. The reason the object rotates rather than deforms as desired is due to the less strict force closure requirement implemented in this thesis. The foam cylinder is significantly stiffer than the other objects and requires more force to deform it. In this example, the object rotates rather than deforms because the contact forces do not meet strict force closure requirements. Because the object rotates rather than deforms as desired, the similarity of the deformed object is not an improvement on the undeformed object. The fingertip force sensors are used to ensure the fingertips make contact with the object but are not used during the shape control process. The issue of rotation and translation during shape control may be handled by continuous use of the fingertip force sensors throughout the grasp and deformation process and is an area for potential future work.

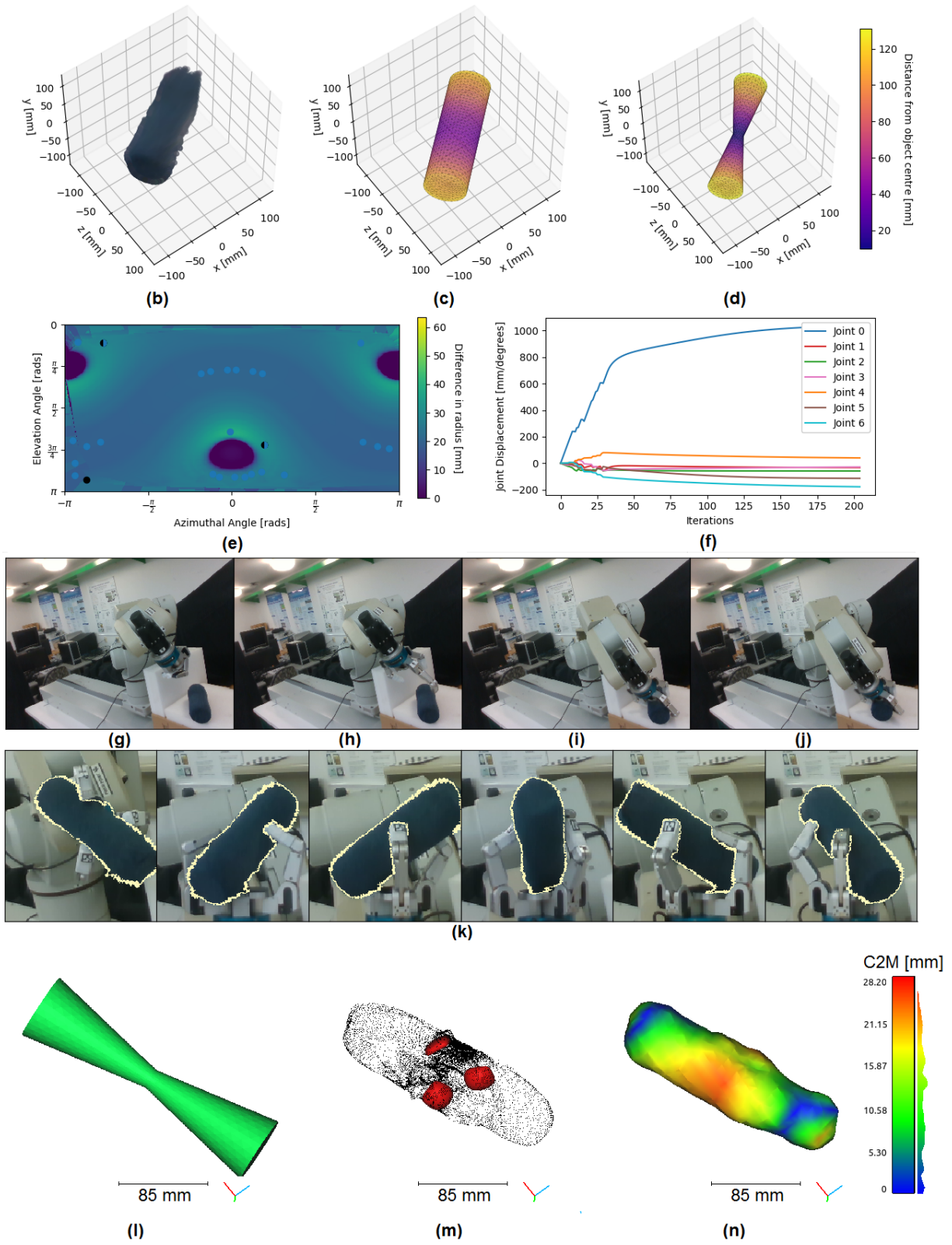


Figure 5.11: End-to-end results for grasping and deforming a foam cylinder placed on a tabletop surface behind an obstacle.

5.2 Discussion

End-to-end results shown in Section 5.1 provide examples of the robotic grasping system autonomously bringing the robotic hand to the optimal valid grasping location for shape control and completing successful grasps while deforming the object towards the desired shape. The system was able to complete successful grasps in 93% of the experiments (13 out of 14).

Images from the experiment that had a failed grasp are shown in Figure 5.12. The gripper is in the grasp pose with the fingers open in Figure 5.12a. In Figure 5.12b, the fingers are closed to make initial contact with the object, but one finger misses the surface of the object due to the error in the fingertip placement compared to the desired contact points. The finger that misses the grasp contact point is on the far side of the object relative to the camera and can be seen on top of the object, rather than making contact with the edge where the desired contact point was planned. Figure 5.12c shows the robot moving from the grasp pose after the unsuccessful grasp.

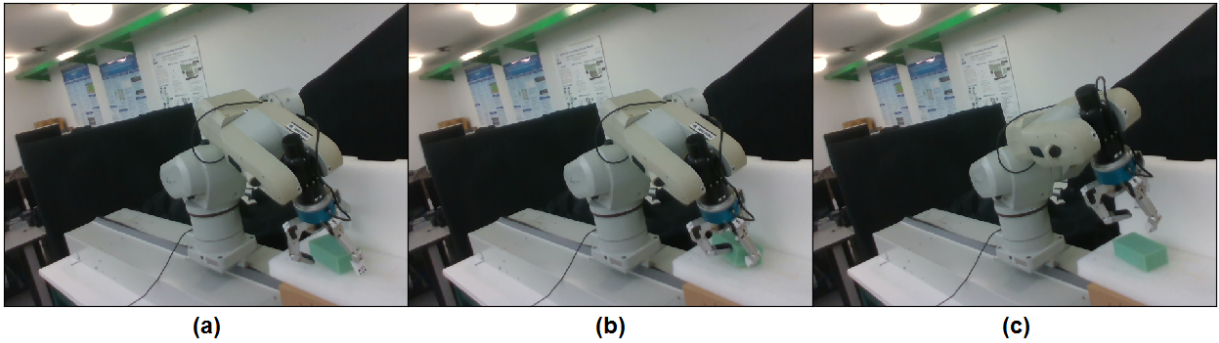


Figure 5.12: Example of unsuccessful grasp by robot. (a) End-effector in grasp pose with fingers open. (b) End-effector in grasp pose after closing fingers to desired radiuses. (c) Robot moved away from object after unsuccessful grasp.

As discussed in Section 4.2.1, there is a simplification made in the fingertip placements to accommodate the hand's kinematic constraints. The fingertip positional error varied between experiments based on the optimal valid contact points for each object. In most cases, the error was small enough not to impact the ability of the hand to successfully grasp the object. Table 5.1 shows the average fingertip positional error across all attempted grasps. The error is defined as the difference between the fingertip position during the grasp, calculated using the forward kinematics of the arm and hand, and the contact points on the object's surface obtained from the grasp point selection algorithm.

Table 5.1: Fingertip positional error ($n = 14$).

Finger	Mean RMS Error [mm]	Standard Deviation [mm]
F1	3.44	3.66
F2	2.37	2.20
F3	3.86	3.76
All Fingers	3.22	3.35

Larger fingertip positional errors are present when there is a large variance in the contact points’ distances from the object’s centre. In future work, a solution to minimizing fingertip positional error could be implemented using additional criteria when selecting grasp points to ensure the points are all within a certain distance from the object’s centre. This would significantly limit the number of valid grasps configurations, reducing the ability to select strong grasp point candidates for shape control, but would ensure more stable grasps on the object.

An additional consideration is the choice of fingertip placement. If the grasp contact points are at the extremities of an object (edges or corners), then the fingertip position error is more likely to cause a missed grasp. Whereas if the contact points are in the middle of the object, fingertip positional error will only result in inaccuracies of the deformation.

After successfully grasping the objects, the quality of the shape control was determined using the multi-view assessment algorithm described in Section 4.3.2. In general, the results for shape control of deformable objects showed improved similarity with the desired shape when the required deformations were large enough to be accurately assessed by the sensor. The tests were broken up into two categories. The first category included experiments where the desired shape was a deformed model of the undeformed shape, requiring small deformations to deform the object towards the desired shape. The second category included experiments where the desired shape was a different shape altogether from the undeformed object, requiring significant deformation to complete the shape control task.

The experiment in Section 5.1.1 belongs to the small deformation category, and experiments in Sections 5.1.2, 5.1.3 and 5.1.4 belong to the large deformation category. For the deformation results presented in Section 4.3.2.2, results in Figure 4.34 belong to the small deformation category and results in Figure 4.37 belong to the large deformation category. The remaining deformation results for both categories are presented in Appendix A.

The average results for the C2M distance introduced in Section 4.3.2 for all objects in the two categories are shown in Table 5.2. The mean average improvement in similarity for each category of objects is also included in the table.

Table 5.2: Mean quantitative results for shape control ($n = 13$)

Desired Shape Category	Deformed to Desired C2M [mm]	Undeformed to Desired C2M [mm]	Change in Similarity (Δ)
Small deformations ($n = 5$)	4.22	2.84	-48%
Large deformations ($n = 8$)	9.32	12.95	28%

The overall improvement in similarity resulting from shape control for the experiments with large deformations was 28%. This result shows that it is possible to deform objects towards desired shapes even when the problem is severely underactuated, and the robot hand has significant kinematic constraints.

The small deformation object category results show that it is not possible to accurately assess small deformations in this research, given the accuracy of the 3D sensor used. To properly assess the quality of deformations when they are small, a different sensor should be used. Some high accuracy, metrology-oriented alternatives to the stereo IR depth sensor used in this research are available commercially [121] or for research [122, 123]. These sensors offer extremely high accuracy depth measurements and are priced accordingly.

The solution was designed and implemented using non-optimized Python code. This implementation led to relatively fast computation times for the initialization and planning stages. The average time to complete each element of the system is shown in Table 5.3.

Table 5.3: Mean computation time for different components of the grasping system ($n = 14$).

System Element	Mean Time [s]	Standard Deviation [s]
Probability Boundary Detection	12.53	13.13
Object Segmentation	2.56	1.09
Object Completion	0.31	0.08
Grasp Contact Points Planning	8.13	0.93
Robot Path Planning	6.02	7.24

The component of the system that takes the longest amount of time is the probability boundary detection used in the segmentation algorithm. Assessing the depth data at each detected edge requires significant computation. If speed is a priority, it may be worthwhile to perform object segmentation without including the depth data. As shown in Figure 4.8, the segmentation results are not as good without including the depth data, so a trade-off between speed and performance would have to be made. The standard deviation of the computation time of the probability boundary detection is significant because the time to complete the algorithm is dependant on the number of edges in the images.

The time to complete grasp point selection and path planning depends on the number of obstacles in the robot workspace and their placement. If the object is placed in a location where many hand poses are not valid, the grasp point selection algorithm takes longer and results in less optimal grasp points for shape control. When the manipulator must move around obstacles to get to the optimal grasp point, the path planning algorithm takes significantly longer.

Although the solution works well in the cases provided, there are limitations. The most notable is the requirement of an operator to identify the grasping area for the object. By cropping the images to only include the current grasping area, the computation required in processing the image is greatly reduced, and the identification and segmentation process is simplified by removing many extraneous objects. Depending on the application, this

limitation may be acceptable, while other deployments may require an additional solution to automatically identify the area to look for an object.

Another limitation of the system is that it is designed to only work on objects that are of a class of primitive shapes. Other more general 3D shape completion methodologies were tested, including 3D volumetric CNN completions. There are numerous successful examples in the literature, but two major issues prevented them from being successful in this research. The first was the lack of training data. When the algorithm was trained to different datasets, it did not work well on new testing data captured from the system’s 3D sensor. Additionally, the quality of the data received due to sensor error makes it difficult to accurately represent the detected object, leading to significant errors in the 3D completions. This issue will be present in any method but seemed to be exacerbated from the CNN compared to the simple primitive shape fitting method that was chosen instead.

Another issue uncovered in this work, already discussed in Section 5.1, is the object rotating or sliding as it is being deformed by the robot hand. This may be a result of inaccurate object estimates, causing the initial contact points of the gripper to not be placed exactly on the object’s surface. Alternatively, it may be due to the simplifications made in the grasp selection algorithm to accommodate the hand’s kinematic constraints. In either case, it makes evident the need for a regrasping, or recovery, strategy to be developed to address the issue, which was beyond the scope of this work.

Possible solutions to some of these limitations are proposed as future work in Section 6.3.

Chapter 6

Conclusion

6.1 Summary

In this thesis, a novel approach to automated 3D deformable object grasping and shape control using a robotic hand was proposed and experimentally validated. A general end-to-end framework for the robotic manipulation of deformable objects based on visual and force sensing was designed and executed in a series of modules.

The proposed solution integrates original contributions in object segmentation using colour and depth images, contact point planning for shape control of 3D deformable objects, collision-free path planning for redundant manipulators and multi-view deformation assessment.

The result of this combination of approaches is a system that successfully identifies a previously unseen object and plans and executes a grasp to control the shape of the 3D deformable object towards the desired input shape. The feasibility and robustness of the proposed solution are validated through a variety of experiments presented in Chapter 5. The results presented in this thesis highlight the complexity of 3D shape control for deformable objects.

The solution to the infinite degree of freedom problem is limited by the number of fingers on the Barrett Hand and its kinematic constraints. Complex reshaping of objects with only three fingers is challenging for humans to achieve, highlighting the difficulty in automating a solution. Additionally, the solution is planned on objects with unknown material properties, making it challenging to predict the true deformations caused by manipulation. Lastly, the limited accuracy of the depth measurements permitted by the RGB-D sensor used in this work inhibits accurate measurements of undeformed objects and the assessment of small deformations.

This work presents significant original contributions to the field and identifies key challenges and limitations for the development of potential future work.

6.2 Contributions

The main contribution presented in this thesis is the development of an original algorithm to select the initial contact points on the surface of a 3D object and automatically position a three-fingered robot hand to deform a sensed object to a desired shape. The algorithm respects the kinematic constraints imposed by the robotic arm and hand but is flexible enough to be applied to any variety of arm and hand combinations. The algorithm is also general enough to grasp and deform objects in various settings. The deformable object grasping system implements novel grasping techniques and an original object representation.

The following is a list of original contributions developed in this thesis:

- A novel augmented fast level sets algorithm with an efficient implementation that segments based on multi-channel colour image data and object boundary probability estimates using the geometry data from a depth image.
- An original object representation using the concept of surface images to represent the measured and desired shape of 3D objects. This reduces the deformable grasping problem to a finite degree of freedom problem.
- A novel algorithm for identifying candidate contact points for deformation control of unknown 3D objects towards desired 3D shapes, which represents a significant advancement over state-of-the-art grasp planning strategies for deformable objects.
- An original path algorithm for redundant manipulators to position a three-fingered robot hand in an optimal valid pose that reaches the selected grasping points while satisfying the kinematic constraints of both the robot arm and hand.
- A multi-view deformation assessment methodology to compare the deformed object to the desired shape and quantify the change in similarity compared to the undeformed object.

The system provides additional benefits that can be applied generally in other solutions:

- Does not rely on any formal object modelling techniques.
- Does not depend on a preexisting 3D object library which is an important benefit as object libraries are not widely available for deformable objects.
- Is general enough to be applied to any combination of robot and end-effector with known kinematics, including mobile robots.
- Generates flexible grasping strategies and end-effector positions that are not limited to specific settings, such as tabletop, and can be applied to a variety of problems.

Another contribution is the presentation of a general end-to-end framework for the robotic manipulation of deformable objects based on visual and force sensing. Up until now, this research has contributed to the publication of two workshop papers. The first is a paper on building a general framework for 3D deformable object grasping and manipulation [104]. The second paper describes building 3D deformable object models in partially observable robotic environments [124]. Additional publications in the field of 3D deformable object manipulation are in preparation at the time of publication of this thesis.

6.3 Future Work

The work presented in this thesis leaves room for significant development in any of the components that make up the deformable object grasping framework. Due to the wide range of problems being addressed, there are many possible directions for future work.

In the initialization stage (Section 4.1), the system can be modified to automatically identify the area to search for objects using a visual attention solution. The system could also be modified to detect and segment multiple objects rather than just a single object. The 3D object completion algorithm could also be modified to work on objects that do not represent the primitive shapes that are currently being detected. Depending on the task at hand, it may be more suitable to use an object detection technique if the objects to be grasped are from a known database or use a neural network-based approach if appropriate training and testing data can be gathered. Current work was implemented using a low-cost single point of view RGB-D sensor, which captures 2.5D observations of the environment. Future work could be developed using higher accuracy full-image range sensors to overcome the limitation of data accuracy in both the initialization and assessment stages. Additionally, multiple sensors could be used with other techniques to complete more accurate and complex 3D shapes of objects before they are grasped. There is significant room for future work in this domain that can be directed based on the specific task.

The planning stage (Section 4.2) relies significantly on the kinematics of the CRS-F3 robot manipulator and the Barrett Hand. While the algorithms are all based on the kinematics of the components being used, they are not limited to specific constraints and could easily be used with robots that have different kinematic constraints. The path planning algorithm is general enough to be used on any redundant manipulator with known kinematics and could even be adapted for mobile robots. The grasp point selection algorithm could also be adapted for another end-effector with different kinematics, potentially increasing the number of possible ways to grasp and deform the object. Future work could also explore different types of grasps other than the impactive fingertip grasps used in this research. Grasp point selection could also be improved to ensure stable grasps by adding a condition on grasp point selection that ensures all grasp points are situated at the same distance from the object's centre. The current work relies on obstacles to be input manually by a user. Automatic obstacle detection could be implemented to increase the autonomy of the system.

Future work in the execution and assessment stage (Section 4.3) could include real-time adjustment of fingertip displacement based on assessing the current deformation. This can be done by further involvement of force sensing or the addition of tactile sensing at the fingertips. A re-planning stage could also be added to the system to re-plan initial contact points based on the initial grasp and deformation results, in order to increase the robustness of the solution when objects slide or rotate when initially grasped, or when the initial grasp attempt fails, two potential situations that were demonstrated in Chapter 5. Multiple viewpoints were introduced in the assessment phase of this work to generate complete 3D data of the deformed object. The system could be improved further by introducing multiple sensors that could be used to generate 3D data of the object as it is being deformed. This stage can also include estimating object behaviour or model

properties based on the deformation assessment. Another exciting area to explore would be a comparison with how a human operator would plan and execute a three-fingered tripod grasp to complete the same shape control problem as the robot and how the similarity results compared.

There is still significant room for novel solutions to be developed and many unsolved problems in the area of 3D deformable object grasping and manipulation. Other potential areas of future work include exploring the limitations uncovered in this thesis and designing innovative techniques to overcome the unsolved problems discussed above.

References

- [1] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
- [2] M. Sezgin and B. Sankur, “Survey over image thresholding techniques and quantitative performance evaluation,” *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–166, 2004.
- [3] R. Muthukrishnan and M. Radha, “Edge detection techniques for image segmentation,” *International Journal of Computer Science & Information Technology*, vol. 3, no. 6, p. 259, 2011.
- [4] R. Adams and L. Bischof, “Seeded region growing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.
- [5] Z. Zivkovic, “Improved adaptive gaussian mixture model for background subtraction,” in *Proc. 17th International Conference on Pattern Recognition (ICPR)*, vol. 2, (Cambridge, United Kingdom), pp. 28–31, IEEE, 2004.
- [6] Y. Deng and B. S. Manjunath, “Unsupervised segmentation of color-texture regions in images and video,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 800–810, 2001.
- [7] V. Caselles, F. Catté, T. Coll, and F. Dibos, “A geometric model for active contours in image processing,” *Numerische mathematik*, vol. 66, no. 1, pp. 1–31, 1993.
- [8] S. Osher and J. A. Sethian, “Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations,” *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, 1988.
- [9] Y. Shi and W. C. Karl, “Real-time tracking using level sets,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, (San Diego, California, USA), pp. 34–41, 2005.
- [10] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
- [11] K. Lai, L. Bo, X. Ren, and D. Fox, “A large-scale hierarchical multi-view rgb-d object dataset,” in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 1817–1824, 2011.

- [12] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [13] I. I. Fouad, S. Rady, and M. G. Mostafa, “Efficient image segmentation of rgb-d images,” in *12th International Conference on Computer Engineering and Systems (ICCES)*, (Cairo, Egypt), pp. 353–358, 2017.
- [14] A. K. Mishra, A. Shrivastava, and Y. Aloimonos, “Segmenting “simple” objects using rgb-d,” in *IEEE International Conference on Robotics and Automation (ICRA)*, (St. Paul, Minnesota, USA), pp. 4406–4413, 2012.
- [15] A. K. Mishra and Y. Aloimonos, “Visual segmentation of “simple” objects for robots,” *Robotics: Science and Systems VII*, vol. 217, 2012.
- [16] A. Mishra, Y. Aloimonos, and C. L. Fah, “Active segmentation with fixation,” in *IEEE 12th International Conference on Computer Vision (ICCV)*, (Kyoto, Japan), pp. 468–475, 2009.
- [17] H. Liu, M. Philipose, and M.-T. Sun, “Automatic objects segmentation with rgb-d cameras,” *Journal of Visual Communication and Image Representation*, vol. 25, no. 4, pp. 709–718, 2014.
- [18] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.
- [19] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze, “Segmentation of unknown objects in indoor environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Vilamoura, Portugal), pp. 4791–4796, 2012.
- [20] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze, “Learning of perceptual grouping for object segmentation on rgb-d data,” *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 64–73, 2014.
- [21] A. Ückermann, R. Haschke, and H. Ritter, “Real-time 3d segmentation of cluttered scenes for robot grasping,” in *Proc. 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, (Osaka, Japan), pp. 198–203, 2012.
- [22] H. Y. Yalic and A. B. Can, “Automatic object segmentation on rgb-d data using surface normals and region similarity,” in *Proc. 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISI-GRAPP)*, (Funchal, Portugal), pp. 379–386, 2018.
- [23] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, “Learning rich features from rgb-d images for object detection and segmentation,” in *European Conference on Computer Vision (ECCV)*, (Zürich, Switzerland), pp. 345–360, Springer, 2014.

- [24] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Boston, Massachusetts, USA), pp. 3431–3440, 2015.
- [25] S.-J. Park, K.-S. Hong, and S. Lee, “Rdfnet: Rgb-d multi-level residual feature fusion for indoor semantic segmentation,” in *IEEE International Conference on Computer Vision (ICCV)*, (Venice, Italy), pp. 4980–4989, 2017.
- [26] R. Schnabel, R. Wahl, and R. Klein, “Efficient ransac for point-cloud shape detection,” *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007.
- [27] P. V. Hough, “Method and means for recognizing complex patterns,” 1962. US Patent 3,069,654.
- [28] J. Wang and M. M. Oliveira, “Improved scene reconstruction from range images,” *Computer Graphics Forum*, vol. 21, no. 3, pp. 521–530, 2002.
- [29] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, and A. Morales, “Mind the gap-robotic grasping under incomplete observation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 686–693, 2011.
- [30] A. H. Quispe, B. Milville, M. A. Gutiérrez, C. Erdogan, M. Stilman, H. Christensen, and H. B. Amor, “Exploiting symmetries and extrusions for grasping household objects,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3702–3708, 2015.
- [31] C. Papazov and D. Burschka, “An efficient ransac for 3d object recognition in noisy and occluded scenes,” in *Asian Conference on Computer Vision*, (Berlin, Germany), pp. 135–148, Springer, 2010.
- [32] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, “Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes,” in *IEEE International Conference on Computer Vision (ICCV)*, (Barcelona, Spain), pp. 858–865, 2011.
- [33] Y. Li, A. Dai, L. Guibas, and M. Nießner, “Database-assisted object retrieval for real-time 3d reconstruction,” *Computer Graphics Forum*, vol. 34, no. 2, pp. 435–446, 2015.
- [34] L. A. Alexandre, “3d descriptors for object and category recognition: a comparative evaluation,” in *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1-3, (Vilamoura, Portugal), p. 7, 2012.
- [35] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3d shapenets: A deep representation for volumetric shapes,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Boston, Massachusetts, USA), pp. 1912–1920, 2015.

- [36] D. T. Nguyen, B.-S. Hua, K. Tran, Q.-H. Pham, and S.-K. Yeung, “A field model for repairing 3d shapes,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, Nevada, USA), pp. 5676–5684, 2016.
- [37] A. Sharma, O. Grau, and M. Fritz, “Vconv-dae: Deep volumetric shape learning without object labels,” in *European Conference on Computer Vision (ECCV)*, pp. 236–250, Springer, 2016.
- [38] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, “Shape completion enabled robotic grasping,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Vancouver, Canada), pp. 2442–2447, 2017.
- [39] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, “Yale-cmu-berkeley dataset for robotic manipulation research,” *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.
- [40] D. Kappler, B. Bohg, and S. Schaal, “Leveraging big data for grasp planning,” in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, (Seattle, Washington, USA), 2015.
- [41] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, “3d-r2n2: A unified approach for single and multi-view 3d object reconstruction,” in *European Conference on Computer Vision (ECCV)*, (Amsterdam, The Netherlands), pp. 628–644, Springer, 2016.
- [42] A. Dai, C. R. Qi, and M. Nießner, “Shape completion using 3d-encoder-predictor cnns and shape synthesis,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Honolulu, Hawaii, USA), 2017.
- [43] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu, “High-resolution shape completion using deep neural networks for global structure and local geometry inference,” in *Proc. IEEE International Conference on Computer Vision (ICCV)*, (Venice, Italy), 2017.
- [44] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, “Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision,” *arXiv preprint arXiv:1612.00814*, 2016.
- [45] E. J. Smith and D. Meger, “Improved adversarial systems for 3d object generation and reconstruction,” in *Proc. 1st Annual Conference on Robot Learning*, (Mountain View, California, USA), pp. 87–96, 2017.
- [46] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni, “3d object reconstruction from a single depth view with adversarial learning,” in *Proc. IEEE International Conference on Computer Vision Workshops*, pp. 679–688, 2017.
- [47] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, “Pcn: point completion network,” in *IEEE International Conference on 3D Vision (3DV)*, (Verona, Italy), pp. 728–737, 2018.

- [48] S. M. Udupa, *Collision detection and avoidance in computer controlled manipulators*. PhD thesis, California Institute of Technology Pasadena, CA, 1977.
- [49] T. Lozano-Perez, "Spatial planning: A configuration space approach," in *Autonomous Robot Vehicles*, pp. 259–271, Springer, 1990.
- [50] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [51] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [52] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," tech. rep., Carnegie-Mellon Univ Pittsburgh PA Robotics Inst, 1993.
- [53] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility-polygon search and euclidean shortest paths," in *26th Annual Symposium on Foundations of Computer Science (SFCS)*, (Portland, Oregon, USA), pp. 155–164, IEEE, 1985.
- [54] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*, (London, United Kingdom), pp. 203–209, Springer, 1998.
- [55] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Anchorage, Alaska, USA), pp. 2902–2908, 2010.
- [56] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems*, pp. 767–774, 2004.
- [57] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [58] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
- [59] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [60] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
- [61] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2, (San Francisco, California, USA), pp. 995–1001, 2000.

- [62] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, “An integrated approach to inverse kinematics and path planning for redundant manipulators,” in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, (Orlando, Florida, USA), pp. 1874–1879, 2006.
- [63] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, “Randomized path planning for redundant manipulators without inverse kinematics,” in *7th IEEE-RAS International Conference on Humanoid Robots*, (Pittsburgh, Pennsylvania, USA), pp. 477–482, 2007.
- [64] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner, “Manipulation planning with workspace goal regions,” in *IEEE International Conference on Robotics and Automation (ICRA)*, (Kobe, Japan), pp. 618–624, 2009.
- [65] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Autonomous Robot Vehicles*, pp. 396–404, Springer, 1986.
- [66] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “Chomp: Gradient optimization techniques for efficient motion planning,” in *IEEE International Conference on Robotics and Automation*, (Salt Lake City, Utah, USA), pp. 489–494, IEEE, 2009.
- [67] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “Stomp: Stochastic trajectory optimization for motion planning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 4569–4574, 2011.
- [68] F. Nadon, A. J. Valencia, and P. Payeur, “Multi-modal sensing and robotic manipulation of non-rigid objects: A survey,” *Robotics*, vol. 7, no. 4, p. 74, 2018.
- [69] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, “Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey,” *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.
- [70] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, and J. L. Wyatt, “Modeling of deformable objects for robotic manipulation: A tutorial and review,” *Frontiers in Robotics and AI*, vol. 7, 2020.
- [71] S. Tokumoto and S. Hirai, “Deformation control of rheological food dough using a forming process model,” in *Proc. IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2, (Washington, District of Columbia, USA), pp. 1457–1464, 2002.
- [72] M. Shibata and S. Hirai, “Soft object manipulation by simultaneous control of motion and deformation,” in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, (Orlando, Florida, USA), pp. 2460–2465, 2006.

- [73] M. Higashimori, K. Yoshimoto, and M. Kaneko, "Active shaping of an unknown rheological object based on deformation decomposition into elasticity and plasticity," in *IEEE International Conference on Robotics and Automation(ICRA)*, (Anchorage, Alaska, USA), pp. 5120–5126, 2010.
- [74] K. Yoshimoto, M. Higashimori, K. Tadakuma, and M. Kaneko, "Active outline shaping of a rheological object based on plastic deformation distribution," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (San Francisco, California, USA), pp. 1386–1391, 2011.
- [75] J. Smolen and A. Patriciu, "Deformation planning for robotic soft tissue manipulation," in *Second International Conferences on Advances in Computer-Human Interactions (ACHI)*, (Washington, District of Columbia, USA), pp. 199–204, IEEE, 2009.
- [76] J. Das and N. Sarkar, "Autonomous shape control of a deformable object by multiple manipulators," *Journal of Intelligent & Robotic Systems*, vol. 62, no. 1, pp. 3–27, 2011.
- [77] J. Das and N. Sarkar, "Robust shape control of deformable objects using model-based techniques," *Advanced Robotics*, vol. 25, no. 16, pp. 2099–2123, 2011.
- [78] J. Das and N. Sarkar, "Passivity-based target manipulation inside a deformable object by a robotic system with noncollocated feedback," *Advanced Robotics*, vol. 27, no. 11, pp. 861–875, 2013.
- [79] D. Navarro-Alarcon, Y.-H. Liu, J. G. Romero, and P. Li, "Model-free visually servoed deformation control of elastic objects by robot manipulators," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1457–1468, 2013.
- [80] D. Navarro-Alarcon, Y.-H. Liu, J. G. Romero, and P. Li, "On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments," *The International Journal of Robotics Research*, vol. 33, no. 11, pp. 1462–1480, 2014.
- [81] D. Navarro-Alarcon, H. M. Yip, Z. Wang, Y.-H. Liu, F. Zhong, T. Zhang, and P. Li, "Automatic 3-d manipulation of soft objects by robotic arms with an adaptive deformation model," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 429–441, 2016.
- [82] D. Navarro-Alarcon and Y.-H. Liu, "Fourier-based shape servoing: a new feedback method to actively deform soft objects into desired 2-d image contours," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 272–279, 2017.
- [83] A. Delgado, C. A. Jara, and F. Torres, "In-hand recognition and manipulation of elastic objects using a servo-tactile control strategy," *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 102–112, 2017.

- [84] A. Delgado, C. A. Jara, and F. Torres, “Adaptive tactile control for in-hand manipulation tasks of deformable objects,” *The International Journal of Advanced Manufacturing Technology*, vol. 91, no. 9-12, pp. 4127–4140, 2017.
- [85] A. Delgado, J. A. Corrales, Y. Mezouar, L. Lequievre, C. Jara, and F. Torres, “Tactile control based on gaussian images and its application in bi-manual manipulation of deformable objects,” *Robotics and Autonomous Systems*, vol. 94, pp. 148–161, 2017.
- [86] F. Nadon and P. Payeur, “Automatic selection of grasping points for shape control of non-rigid objects,” in *IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, (Ottawa, Canada), pp. 1–7, 2019.
- [87] F. Nadon and P. Payeur, “Grasp selection for in-hand robotic manipulation of non-rigid objects with shape control,” in *IEEE International Systems Conference (SysCon)*, (Virtual Conference), pp. 1–8, 2020.
- [88] L. Zaidi, B. Belhassen-Chedli, L. Sabourin, and M. Youcef, “Modeling and analysis of 3d deformable object grasping,” in *23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, (Smolenic, Slovakia), pp. 1–8, IEEE, 2014.
- [89] L. Zaidi, J. A. Corrales, B. C. Bouzgarrou, Y. Mezouar, and L. Sabourin, “Model-based strategy for grasping 3d deformable objects using a multi-fingered robotic hand,” *Robotics and Autonomous Systems*, vol. 95, pp. 196–206, 2017.
- [90] L. Zaidi, J. A. C. Ramon, L. Sabourin, B. C. Bouzgarrou, and Y. Mezouar, “Grasp planning pipeline for robust manipulation of 3d deformable objects with industrial robotic hand + arm systems,” *Applied Sciences*, vol. 10, no. 23, p. 8736, 2020.
- [91] Z. Hu, T. Han, P. Sun, J. Pan, and D. Manocha, “3-d deformable object manipulation using deep neural networks,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4255–4261, 2019.
- [92] F. F. Khalil, P. Curtis, and P. Payeur, “Visual monitoring of surface deformations on objects manipulated with a robotic hand,” in *IEEE International Workshop on Robotic and Sensors Environments*, (Phoenix, Arizona, USA), pp. 1–6, 2010.
- [93] A.-M. Cretu, E. M. Petriu, P. Payeur, and F. F. Khalil, “Deformable object segmentation and contour tracking in image sequences using unsupervised networks,” in *Canadian Conference on Computer and Robot Vision (CRV)*, (Ottawa, Canada), pp. 277–284, IEEE, 2010.
- [94] A. J. Valencia, F. Nadon, and P. Payeur, “Toward real-time 3d shape tracking of deformable objects for robotic manipulation and shape control,” in *IEEE SENSORS*, (Montreal, Canada), pp. 1–4, 2019.
- [95] A. J. Valencia and P. Payeur, “Combining self-organizing and graph neural networks for modeling deformable objects in robotic manipulation,” *Frontiers in Robotics and AI*, vol. 7, 2020.

- [96] F. Hui, P. Payeur, and A.-M. Cretu, “Visual tracking of deformation and classification of non-rigid objects with robot hand probing,” *Robotics*, vol. 6, no. 1, p. 5, 2017.
- [97] I. Cuiral-Zueco and G. López-Nicolás, “Rgb-d tracking and optimal perception of deformable objects,” *IEEE Access*, vol. 8, pp. 136884–136897, 2020.
- [98] A. Malti, A. Bartoli, and T. Collins, “Template-based conformal shape-from-motion from registered laparoscopic images,” in *Medical Image Understanding and Analysis Conference (MIUA)*, vol. 1-2, (London, United Kingdom), p. 6, 2011.
- [99] T. R. dos Santos, A. Seitel, T. Kilgus, S. Suwelack, A.-L. Wekerle, H. Kenngott, S. Speidel, H.-P. Schlemmer, H.-P. Meinzer, T. Heimann, and L. Maier-Hein, “Pose-independent surface matching for intra-operative soft-tissue marker-less registration,” *Medical Image Analysis*, vol. 18, no. 7, pp. 1101–1114, 2014.
- [100] L. Han, H. Wang, Z. Liu, W. Chen, and X. Zhang, “Vision-based cutting control of deformable objects with surface tracking,” *IEEE/ASME Transactions on Mechatronics*, 2020.
- [101] E. De Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, “Performance capture from sparse multi-view video,” in *SIGGRAPH: International Conference on Computer Graphics and Interactive Technique*, (Los Angeles, California, USA), pp. 1–10, ACM, 2008.
- [102] A. Jordt and R. Koch, “Fast tracking of deformable objects in depth and colour video,” in *22nd British Machine Vision Conference (BMVC)*, (Dundee, United Kingdom), pp. 1–11, 2011.
- [103] D. T. Ngo, J. Östlund, and P. Fua, “Template-based monocular 3d shape recovery using laplacian meshes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 172–187, 2015.
- [104] F. Nadon, A. J. Valencia, N. Sambandam, S. Rowlands, J. Dickens, and P. Payeur, “Toward a general framework for 3d deformable object grasping and manipulation,” in *the Workshop on Robotic Manipulation of Deformable Objects (ROMADO) for IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Las Vegas, Nevada, USA, Virtual Conference), 2020.
- [105] Intel, *Intel® RealSense™ 400 Series Product Datasheet*, 1 2019. Rev. 5.
- [106] A. Grunnet-Jepsen, J. N. Sweetser, and J. Woodfill, “Best-known-methods for tuning Intel® Realsense™ depth cameras d400 series for best performance,” Tech. Rep. Rev 2.0, Intel, 2020.
- [107] CRS Robotics Corporation, *F3 Robot Sysem User Guide*, 6 2000. Rev. 1.
- [108] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

- [109] Barrett Technology, LLC, *BarrettHand BH8-280 User Manual*, 8 2019. Ver. 45.
- [110] W. Kabsch, “A solution for the best rotation to relate two sets of vectors,” *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, pp. 922–923, 1976.
- [111] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.
- [112] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic, “The grasp taxonomy of human grasp types,” *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 66–77, 2015.
- [113] M. A. Roa and R. Suárez, “Grasp quality measures: review and performance,” *Autonomous Robots*, vol. 38, no. 1, pp. 65–88, 2015.
- [114] ANSYS Inc., *ANSYS Workbench (Ver. R3) [Academic Student Licence]*, 7 2019. Retrieved from <https://www.ansys.com/academic/students/ansys-student>.
- [115] J.-P. Berrut and L. N. Trefethen, “Barycentric lagrange interpolation,” *SIAM Review*, vol. 46, no. 3, pp. 501–517, 2004.
- [116] D. Berenson, “Manipulation of deformable objects without modeling and simulating deformation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Tokyo, Japan), pp. 4525–4532, 2013.
- [117] W. A. Wolovich and H. Elliott, “A computational technique for inverse kinematics,” in *23rd IEEE Conference on Decision and Control*, (Las Vegas, Nevada, USA), pp. 1359–1363, 1984.
- [118] R. Macknojjia, A. Chávez-Aragón, P. Payeur, and R. Laganière, “Experimental characterization of two generations of kinect’s depth sensors,” in *IEEE International Symposium on Robotic and Sensors Environments Proceedings (ROSE)*, (Magdeburg, Germany), pp. 150–155, 2012.
- [119] CloudCompare, *CloudCompare (Ver. 2.11.1) [GPL Software]*, 7 2020. Retrieved from <http://www.cloudcompare.org/>.
- [120] M. Kazhdan, M. Bolitho, and H. Hoppe, “Poisson surface reconstruction,” in *Proc. 4th Eurographics Symposium on Geometry Processing*, vol. 7, (Goslar, Germany), 2006.
- [121] Creaform, *HandySCAN3D™*, 2 2021. https://www.creaform3d.com/sites/default/files/assets/brochures/files/handyscan3d_brochure_en_hq_20210202_1.pdf.
- [122] M.-A. Drouin, G. Godin, M. Picard, J. Boisvert, and L.-G. Dicaire, “Structured-light systems using a programmable quasi-analogue projection subsystem,” in *Emerging Digital Micromirror Device Based Systems and Applications XII*, vol. 11294, p. 112940O, International Society for Optics and Photonics, 2020.

- [123] J. R. Bartels, J. Wang, W. Whittaker, and S. G. Narasimhan, “Agile depth sensing using triangulation light curtains,” in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, (Seoul, South Korea), pp. 7900–7908, 2019.
- [124] N. Sambandam, S. Rowlands, A. J. Valencia, P. Payeur, and A.-M. Cretu, “Building 3d deformable object models in partially observable robotic environments,” in *the Workshop on Managing Deformation (MaDef) for IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Las Vegas, Nevada, USA, Virtual Conference), 2020.

APPENDICES

Appendix A

Supplementary Results

This appendix presents additional results obtained during the completion of this work. These results are in addition to the more in-depth analysis and results presented throughout Chapter 4 and in Section 5.1, and make up the remainder of the data set for the results in Section 5.2.

Figure A.1 presents the initial setup to grasp and deform an object for three additional experiments, labelled (a) to (c). These results represent the remaining experiments for the small deformation category.

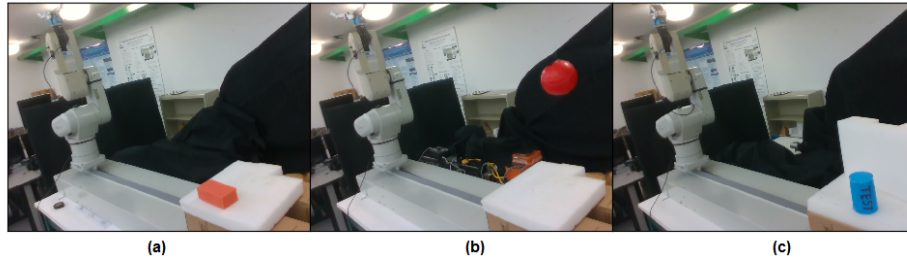


Figure A.1: Initial setup for grasping and deforming an object, supplementary results (a) to (c) for objects that require small deformations compared to the desired 3D shape.

Figure A.2 presents the deformation results for each of the additional experiments presented in Figure A.1.

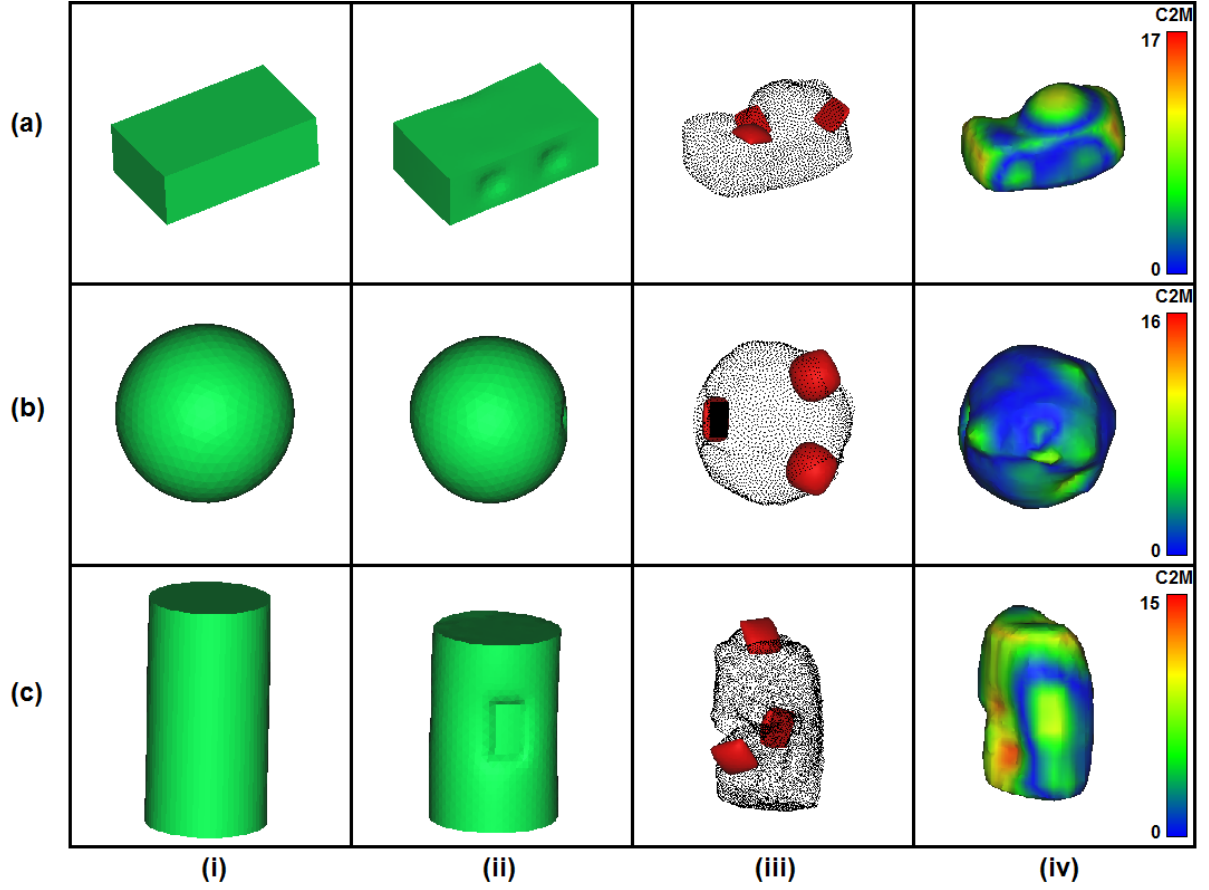


Figure A.2: Deformation assessment for supplementary results (a) to (c). (i) measured undeformed object, (ii) desired 3D shape, (iii) deformed object point cloud with fingertip positions identified, and (iv) measured deformed object with C2M distance to desired shape shown on surface.

Figures A.3 and A.4 present the initial setup and deformation results for the remaining experiments. The remaining experiments all belong to the category of large deformations.

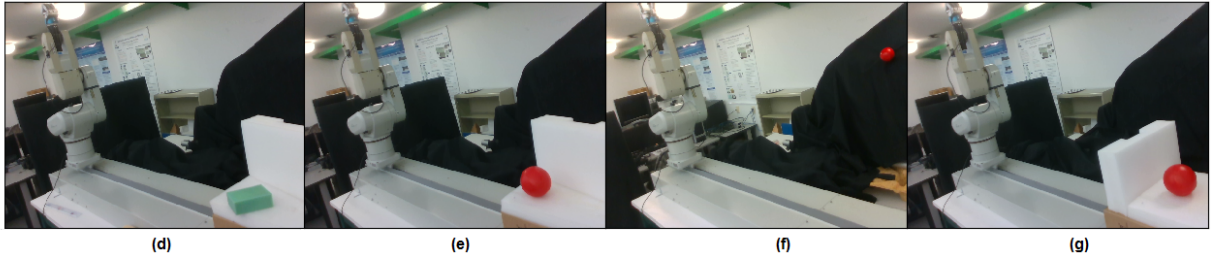


Figure A.3: Initial setup for grasping and deforming an object, supplementary results (d) to (g) for objects that require large deformations compared to the desired 3D shape.

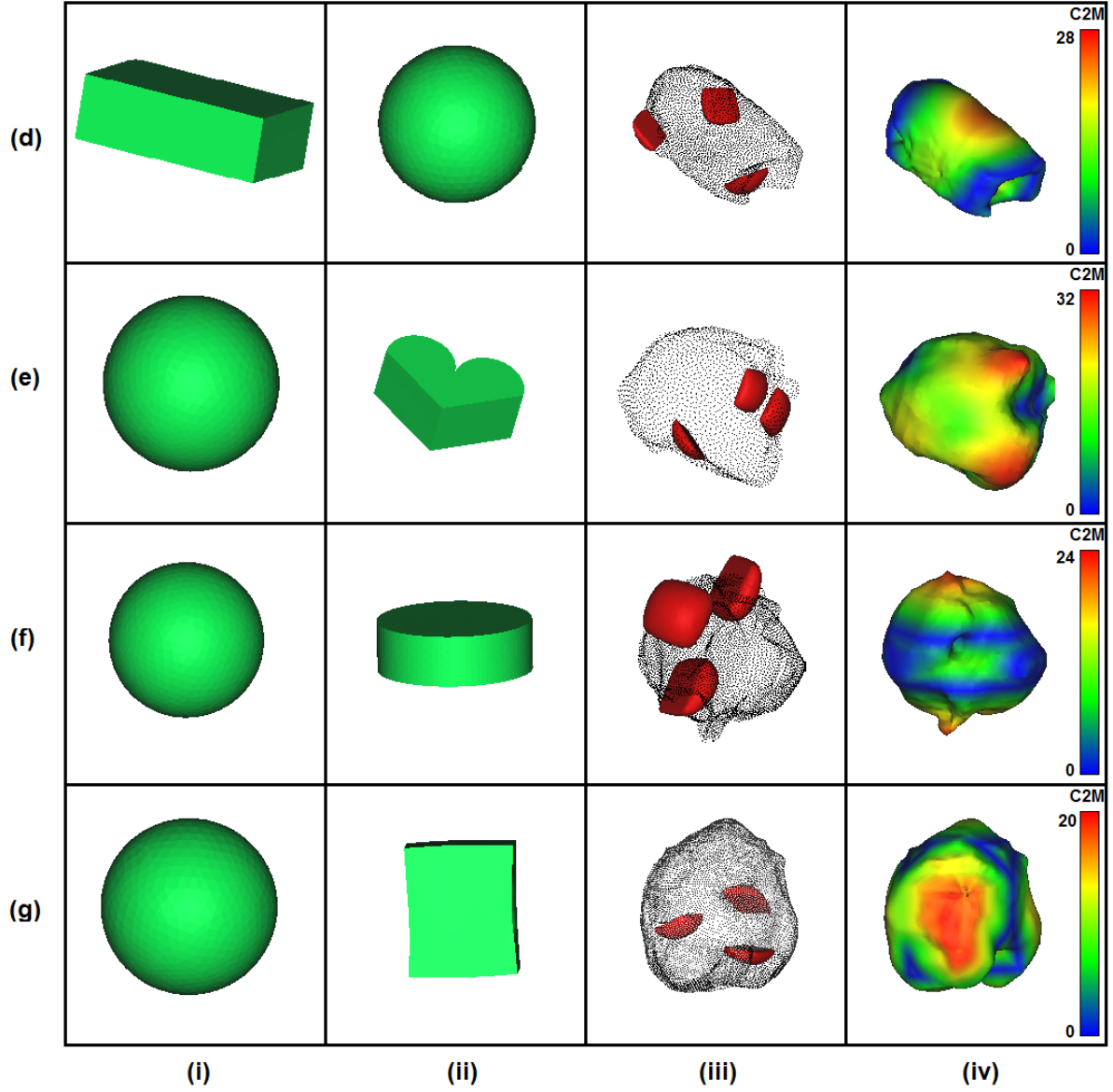


Figure A.4: Deformation assessment for supplementary results (d) to (g). (i) measured undeformed object, (ii) desired 3D shape, (iii) deformed object point cloud with fingertip positions identified, and (iv) measured deformed object with C2M distance to desired shape shown on surface.