1-17-2014

Scene-Dependent Human Intention Recognition for an Assistive Robotic System

Kester Duncan
University of South Florida, kesterduncan@gmail.com

Follow this and additional works at: https://scholarcommons.usf.edu/etd

Part of the Artificial Intelligence and Robotics Commons, and the Robotics Commons

Scholar Commons Citation
Duncan, Kester, "Scene-Dependent Human Intention Recognition for an Assistive Robotic System" (2014). Graduate Theses and Dissertations.
https://scholarcommons.usf.edu/etd/5009

This Dissertation is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Scene-Dependent Human Intention Recognition for an Assistive Robotic System

by

Kester Duncan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Co-Major Professor: Sudeep Sarkar, Ph.D.
Co-Major Professor: Rajiv Dubey, Ph.D.
Dmitry Goldgof, Ph.D.
Rangachar Kasturi, Ph.D.
Thomas Sanocki, Ph.D.

Date of Approval:
January 17, 2014

Keywords: Object-Action Pairs, 3D Object Pose Estimation, Superquadrics, Markov Networks, RGB-D Object Categorization

Copyright © 2014, Kester Duncan
DEDICATION

This dissertation is dedicated to my loving wife, mother, and family for their encouragement and patience over the years.
ACKNOWLEDGMENTS

First and foremost, I would like to thank my Lord and Savior Jesus Christ for being my provider, my strength, my rock, my peace, my shield, my great reward, and my life. I would also like to acknowledge my advisor, Dr. Sudeep Sarkar, who has been patient with me throughout this journey. I thank him for his immeasurable support and guidance as well as the wealth of knowledge and wisdom he imparted to me over the years. Many thanks go out to my other advisor Rajiv Dubey and the committee members Rangachar Kasturi, Dmitry Goldgof, and Thomas Sanocki for their comments and invaluable input. I would also like to extend many thanks to my family for providing the moral support necessary for undertaking this venture, especially my wife and mother. Thanks also to my good friends Sharrine, Lasceeka, Gabe, Jeremy, Darnell, and Colin. Furthermore, I would have never been on this journey if I was not first provided with the opportunity to obtain a sound undergraduate education at Coppin State University as a scholar in the Honors College. On that note, I would like to thank Dechelle Forbes and Ron L. Collins Sr. for their great leadership and encouragement. Last, but not least, I would like to extend thanks to my lab colleagues Ravikiran Krishnan, Fillipe Souza, Rajmadhan Ekambaram, Ravi Panchumarthy, and Aveek Brahmachari for the encouragement and inspiration provided throughout the years.
# TABLE OF CONTENTS

LIST OF TABLES iii

LIST OF FIGURES iv

ABSTRACT vii

CHAPTER 1 INTRODUCTION 1
  1.1 Motivation 1
  1.2 Scope of This Dissertation 4
  1.3 Contributions of This Work 6
  1.4 Outline of Dissertation 7

CHAPTER 2 LITERATURE REVIEW 8
  2.1 Human Robot Interaction 8
  2.2 Human Intention Recognition 10
  2.3 Object Pose Estimation 12

CHAPTER 3 OBJECT SEGMENTATION 15
  3.1 Background 16
  3.2 RGB-D Segmentation 16

CHAPTER 4 OBJECT CATEGORIZATION 20
  4.1 Background 22
    4.1.1 Scale Invariant Feature Transform (SIFT) 22
    4.1.2 Histograms of Oriented Gradients (HOG) 25
    4.1.3 Fast Point Feature Histograms (FPFH) 28
    4.1.4 Bag-of-Words Representation (BoW) 32
    4.1.5 Support Vector Machines (SVMs) 33
  4.2 RGB-D Object Categorization 37

CHAPTER 5 OBJECT POSE ESTIMATION USING SUPERQUADRICS 41
  5.1 Background 43
    5.1.1 Superquadrics 43
    5.1.2 Voxels 48
  5.2 Multi-scale Voxelization 48
LIST OF TABLES

Table 7.1  The effect of voxel size on the recovered pose estimates.  
Table 7.2  Pose estimation results of our proposed algorithm, Biegelbauer et al. (HS), and Solina et al. (Classic) on Dataset 1.  
Table 7.3  Pose estimation results of our proposed algorithm and the classic superquadric fitting algorithm on a subset of Dataset 2.  
Table 7.4  Average shape fitting performance.  
Table 7.5  Grasping test using a Wheelchair-Mounted Robotic Arm (WMRA)  
Table 7.6  All possible objects, actions, and object-action pairs.  
Table 7.7  Groups of object-action pairs evaluated in this work.  
Table 7.8  Examples of scenes used for evaluation.  
Table 7.9  Average number of interactions required for the 15th - 20th sessions for the simultaneous object and position changes in the scene test as shown in Figure 7.13.
LIST OF FIGURES

Figure 1.1 Conceptual plot of the time required to communicate intent versus the communication ability (nature of disability) of an individual. 3
Figure 1.2 Scope of this work. 4
Figure 1.3 Schematic of our complete human-robot collaborative system. 5
Figure 2.1 Control approaches for human-robot interaction. 9
Figure 2.2 Configuration of human-robot interactive systems using intention recognition. 11
Figure 2.3 Failure of pose estimation methods relying on 2D data. 13
Figure 3.1 Segmentation overview: Candidate object hypotheses are extracted from RGB-D images for further processing. 15
Figure 3.2 Segmentation process followed in this work: Given an RGB-D point cloud, we first extract the table plane then extract the point clusters corresponding to candidate objects located on the table. 17
Figure 3.3 Segmentation process adopted in this work. 18
Figure 4.1 Object Categorization Overview: The generic classes for object hypotheses are determined using multi-modal features. 20
Figure 4.2 An example of SIFT descriptor matching. 23
Figure 4.3 Visual example of the HOG descriptor. 27
Figure 4.4 Illustration of the Darboux frame and the PFH features for a pair of points $p_s$ and $p_q$ with their respective normals. 29
Figure 4.5 The influence regions for a Point Feature Histogram and a Fast Point Feature Histogram. 30
Figure 4.6 Illustration of the construction of Bag-of-Words models 32
Figure 4.7 A hyperplane dividing two linearly separable classes (black points - $c_1$, white points - $c_2$). 34
Figure 4.8  Our object categorization approach. 38
Figure 5.1  Object Pose Estimation Overview: The 3D position and orientation of objects are determined using superquadrics. 41
Figure 5.2  Case for superquadrics. 42
Figure 5.3  Examples of superquadrics for different shape parameters $\epsilon_1$ and $\epsilon_2$. 47
Figure 5.4  2D representation of a voxel. 48
Figure 5.5  The computational time required for superquadric fitting using regular sampling grids. 49
Figure 5.6  Execution times for superquadric fitting in milliseconds for different voxel sizes. 50
Figure 5.7  Overview of point cloud downsampling. 51
Figure 6.1  Human Intention Recognition Overview: The most likely object and action pair is determined. 54
Figure 6.2  An overview of our intention recognition approach which uses an object-action intention network, a set of queries for interacting with the user and learning process to improve predictions. 55
Figure 6.3  Example of a small Markov Network with its defined local factors. 58
Figure 6.4  General form of an Object-Action Intention Network. 60
Figure 6.5  Example of an Object-Action Intention Network in (b) for a real 3D scene as captured by an RGB-D camera shown in (a). 61
Figure 7.1  Examples of objects from our RGB-D dataset. 68
Figure 7.2  Segmentation results for typical scenes. 69
Figure 7.3  Segmentation results for scenes with increased clutter and occlusion. 70
Figure 7.4  Categorization results for a typical scene. 70
Figure 7.5  Categorization results for a scene with increased clutter. 71
Figure 7.6  Confusion matrix obtained for our RGB-D Categorization method on our dataset. 72
Figure 7.7  Examples of objects from our dataset (top row) and the RGBD Dataset (bottom row). 73
Figure 7.8  Average computational times for the three algorithms tested in this work on Dataset 1 - Proposed approach: 40ms, Solina et al.: 2.02s, Biegelbauer et al.: 7.88s.

Figure 7.9  Visual results of the superquadric fitting.

Figure 7.10  The Wheelchair-Mounted Robotic Arm (WMRA) system that is used to assist physically-challenged individuals.

Figure 7.11  Intention recognition results for a constant scene on the object-action pair groups listed in Table 7.7: objects and their positions relative to the camera are unchanged.

Figure 7.12  Intention recognition results for object position changes in the scene.

Figure 7.13  Intention recognition results for simultaneous object and position changes in the scene.

Figure 7.14  Random intention selection after learning: A group of intentions are learned over time then the amount of interaction required to choose one of them from the group is determined.

Figure 7.15  Random intention selection for different learning rates $\lambda$. 
ABSTRACT

In order for assistive robots to collaborate effectively with humans for completing everyday tasks, they must be endowed with the ability to effectively perceive scenes and more importantly, recognize human intentions. As a result, we present in this dissertation a novel scene-dependent human-robot collaborative system capable of recognizing and learning human intentions based on scene objects, the actions that can be performed on them, and human interaction history. The aim of this system is to reduce the amount of human interactions necessary for communicating tasks to a robot. Accordingly, the system is partitioned into scene understanding and intention recognition modules. For scene understanding, the system is responsible for segmenting objects from captured RGB-D data, determining their positions and orientations in space, and acquiring their category labels. This information is fed into our intention recognition component where the most likely object and action pair that the user desires is determined.

Our contributions to the state of the art are manifold. We propose an intention recognition framework that is appropriate for persons with limited physical capabilities, whereby we do not observe human physical actions for inferring intentions as is commonplace, but rather we only observe the scene. At the core of this framework is our novel probabilistic graphical model formulation entitled Object-Action Intention Networks. These networks are undirected graphical models where the nodes are comprised of object, action, and object feature variables, and the links between them indicate some form of direct probabilistic interaction. This setup, in tandem with a recursive Bayesian learning paradigm, enables our system to adapt to a user’s preferences. We also propose an algorithm for the rapid estima-
tion of position and orientation values of scene objects from single-view 3D point cloud data using a multi-scale superquadric fitting approach. Additionally, we leverage recent advances in computer vision for an RGB-D object categorization procedure that balances discrimination and generalization as well as a depth segmentation procedure that acquires candidate objects from tabletops. We demonstrate the feasibility of the collaborative system presented herein by conducting evaluations on multiple scenes comprised of objects from 11 categories, along with 7 possible actions, and 36 possible intentions. We achieve approximately 81% reduction in interactions overall after learning despite changes to scene structure.
CHAPTER 1
INTRODUCTION

1.1 Motivation

In order to perform daily self-care activities and interact with their environment, individuals with reduced capabilities often rely on assistants or caregivers. As a result, billions of dollars are spent annually on assistive robotic technologies for application in home, work, and play environments. In so doing, the elderly, persons with disabilities, and injured veterans are provided with opportunities to achieve higher levels of independence, dignity, and quality of life as these technologies can reduce dependence on caregivers and increase self-sufficiency.

However, there are many challenges to developing such robotic technologies. Firstly, seeing a bottle, reaching for it, and picking it up to pour its contents are relatively easy tasks for a fully-functional human; however, this task is difficult for both individuals with reduced capabilities and robots. For a robot to complete such tasks, it must possess the perceptive ability to effectively process the individual’s environment, namely the scene. This involves extracting the object from captured sensor data, determining the object’s type (e.g. bottle vs. cup), and determining its exact position in space, along with manifold mechanical tasks. These individual problems have been considerably addressed in the computer vision and robotics literatures [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], however they still remain largely unsolved. In this dissertation, we leverage recent advances in computer vision with respect to object segmentation and categorization and we directly address the issue of 3D object pose estimation in an effort to improve robotic perception.
Secondly, a great challenge exists when there is a lack of full or reliable communication between the robot and the human for task execution. For instance, it is relatively easy for a fully-functional individual to directly express their intent (in terms of task goals), yet for persons with reduced capabilities this may prove to be quite difficult or impossible depending on their communication ability. As a result, the problem of recognizing one’s intention is brought to the forefront [12]. The term intention is defined as ‘what someone aims at or chooses’ [13] or as ‘short-term goals to which an agent has committed himself’ [14]. Recognizing intentions is therefore a requisite for successful communication and collaboration [15]. These intentions are inferred from actions carried out or from changes in the environment [16, 12, 17]. It is common for humans to observe each other and on the basis of their observations, correctly infer the goals and intentions of others. This ability is often considered ordinary and effortless and is commonly referred to as the “theory of mind” [18, 19] in psychological work.

Therefore, in order to build robots that are competent assistants, we must endow them with this intelligent ability in order to understand the action that is to be performed on the environment, e.g. pick up the red cup. In so doing, we can reduce the need for direct human-robot interaction and thereby maximize robotic task performance. In this dissertation, we present a framework that aims at developing such capabilities.

One of the principal benefits of recognizing intentions is the reduction in required communication time for task execution, as depicted conceptually in Figure 1.1, which is a qualitative plot capturing communication time versus communication ability. In the figure, the solid curve represents the amount of time required by users to communicate intent without intention recognition which is higher than the dotted curve representing the time required with intention recognition. For an individual incapable of physically moving or communicating verbally, such as a locked-in individual, we can expect a notable decrease in the amount of time to communicate their intentions.
Consider as an example, a robot that helps a person incapable of communicating physically choose a task to perform at a breakfast table. The robot is capable of scanning the table and for every item that is found, a group of possible tasks that can be performed with it is recorded in a list. Subsequently, the robot asks the individual to indicate the task they want to perform. Suppose there are $n$ possible tasks on the list. For the extreme case where the user desires the $n^{th}$ task, the robot would have had to prompt the user $n$ times in order to know what to do. With intention recognition, it is possible to reduce the length of this list by only considering items the individual would most likely want to be performed and thereby reduce the amount of time necessary for communicating their intent.

As will be discussed in a subsequent section, most current human intention recognition approaches rely on observing human physical actions for determining their intent. This is
inappropriate for dealing with persons with disabilities wherein it may be quite difficult or impossible for them to perform activities on the environment. In this dissertation, we deviate from this common paradigm and use only the scene information for determining a user’s intent. This has applications beyond assistive devices such as urban search and rescue and surveillance.

1.2 Scope of This Dissertation

Activities of daily living are basic self-care activities ranging from eating or drinking to basic hygiene and cleaning. An activity is primarily a sequence of steps involving objects and actions carried out to accomplish a specific task. As depicted in the example of Figure 1.2, the activity ‘Drink a soda’ consists of steps whereby a soda can (object) is picked up (action) by a robotic arm and poured (action) into a cup (object) followed by moving the cup to facilitate drinking. In this dissertation, we focus on recognizing these individual steps

Figure 1.2. Scope of this work. We determine the object-action pair that represents a user’s intention at a particular step of an activity and attempt to reduce the amount of human-robot interaction necessary to communicate this step.

and actions carried out to accomplish a specific task. As depicted in the example of Figure 1.2, the activity ‘Drink a soda’ consists of steps whereby a soda can (object) is picked up (action) by a robotic arm and poured (action) into a cup (object) followed by moving the cup to facilitate drinking. In this dissertation, we focus on recognizing these individual steps
from captured visual data and we present novel algorithms to accomplish such. A schematic for our complete human-robot collaborative system is depicted in Figure 1.3. The system is partitioned into scene understanding and intention recognition components. We outline novel algorithms for the items highlighted in green.

Figure 1.3. Schematic of our complete human-robot collaborative system. Object category and pose information along with a user’s past decision history are incorporated in our approach for intention recognition. In this dissertation, we present novel algorithms for the items highlighted in green.

For scene understanding, the robotic system flow proceeds as follows. Raw visual data is captured from an RGB-D sensor and the first task is to extract candidate objects via segmentation, which are represented as point cloud clusters, for further processing. Next, the position and orientation of these objects as well as their category identities are ascertained via object pose estimation and object categorization respectively. For pose estimation, a multi-scale superquadric fitting strategy is proposed to rapidly determine object pose and shape characteristics from single view point clouds. For object categorization, we leverage
recent advances in computer vision and adopt a multi-modal approach whereby 2D and 3D features are used to effectively categorize objects.

Object pose and category information are required as input to our intention recognition component. At this stage, the most likely object and action the user desires is determined. At the crux of the intention recognition framework is a novel undirected graphical model formulation entitled Object-Action Intention Networks, which makes it possible to recognize and learn human intentions based only on scene content and learned interaction history.

1.3 Contributions of This Work

In this dissertation, we present novel scene understanding and human intention recognition algorithms for use within an assistive human-robot collaborative system. The principal aim of the system is to allow persons with disabilities to perform tasks on their environment with reduced robot interaction. Our contributions to the state of the art can be summarized as follows:

- First, we present an algorithm for the rapid recovery of position and orientation values of scene objects from 3D data. The algorithm discussed here uses superquadrics, which are a family of parametric shapes, as well as a coarse-to-fine voxelization scheme that allows the fast recovery of the superquadric model parameters with good accuracy.

- Second, we present a novel human intention recognition framework that is appropriate for persons with limited physical abilities. This framework uses a novel probabilistic graphical model formulation entitled Object-Action Intention networks that enables recognition of human intentions from scene information and the incremental learning of past decisions.
• Last, but not least we present a scene-dependent human-robot collaborative system that enables the recognition and learning of human intentions in order to reduce the amount of interactions required to communicate tasks to a robot.

1.4 Outline of Dissertation

The rest of this dissertation is structured as follows. Chapter 2 reviews the human robot interaction, human intention recognition, and object pose estimation approaches reported in the literature and provides some related work pertaining to this research. In Chapter 3, we present some background information on segmentation and present our approach for extracting objects from table-top scenes. Similarly, in Chapter 4, we describe the state of the art algorithms at the core of our object categorization procedure as well as provide the details of our approach. Chapter 5 presents our multi-scale superquadric fitting approach for pose estimation and Chapter 6 presents our scene-dependent human intention recognition approach which uses our object-action intention networks. The results of our intention framework are presented in Chapter 7. We then discuss our conclusions and the future directions for work in Chapter 8.
2.1 Human Robot Interaction

Human Robot Interaction (HRI) is an area of study involving understanding, designing, and evaluating robotic systems that are capable of human-like interactions [20]. This field is comprised of multiple disciplines and includes many challenging problems that if they were to be solved, they would produce solutions that would have great positive social impact. Current HRI research focuses on the development of software and hardware that facilitates a range of tasks including but not limited to urban search and rescue, assistive devices, elderly care, and robotic companions. Recently, this field has received considerable attention with recent advances in sensor technology and the ever increasing need for robots that provide physical assistance for everyday activities that have become difficult or impossible to accomplish for individuals with reduced capabilities.

The HRI problem has evolved over the years. In early HRI research, two paradigms for human-robot interaction were given substantial attention, namely supervisory control and teleoperation. Supervisory control involved a human supervising the behavior of an autonomous system and only intervening when necessary e.g. factory automation, whereas teleoperation involved the human directly controlling every action of a robotic system e.g. controlling a robotic arm. HRI has evolved to allow varying levels of control with an emphasis on human interaction.

One particular facet of human-robot systems that is garnering considerable attention is that of mixed-initiative architectures [21]. For such architectures, the robots and human users
Figure 2.1. Control approaches for human-robot interaction. Circles with ‘H’ represent humans, ‘R’ represents robot, and ‘E’ represents the environment. The arrows depict the communication flow.

share the decision making process. Within this framework, there are different approaches such as adjustable autonomy and collaborative control (see Figure 2.1). With adjustable autonomy, there is a dynamic transfer of control from human to robot and vice versa, whereas for collaborative control, decisions are evaluated by both agents with the aim of achieving a common goal. In this dissertation, we follow the collaborative control approach in an effort to leverage the strengths of both the human and robotic agents.

Furthermore, the need for collaborative systems are manifold. Robots are incapable of performing tasks autonomously with adequate reliability in the uncontrolled and heterogeneous environments of private homes [20]. In some cases, they are only able to execute sophisticated tasks when operated by a human. Also, it is well known that humans and robotic systems possess different skill sets which may complement each other to achieve difficult tasks. Robotic systems may be more precise than humans at certain tasks, however there are tasks that the robot may not be sufficiently equipped to handle thereby requiring human intervention for guidance, such as tasks requiring sophisticated judgment. For complex tasks, it is beneficial to have a human in the loop to monitor, make decisions, or correct
a robot’s actions (for example, [22]). However, continuous monitoring or operation may not be possible as in the case of persons with reduced faculties as it often fatigues a human operator, confines their attention, and prevents them from undertaking other activities.

As a result, one of the core challenges for collaborative systems is determining which tasks are best done by either the human, the robotic system, or a collaboration of both. This has been well addressed in the literature [23, 24] and the focus has shifted from determining which agent performed a task best to discovering how tasks can be best shared by both humans and robots working in concert or in other words, working as a team [25, 26, 27]. The work in this dissertation entails a collaboration of both the human and the robot and is in essence similar to that of [28] whereby we want to minimize the workload of the human while maximizing robotic task performance. However, our focus is not on optimally partitioning the task space, but rather attempting to maximize robotic assistance in a human-centered environment. We accomplish this by performing human intention recognition. With intention recognition, we attempt to reduce the amount of interaction necessary for communicating tasks to the robot.

2.2 Human Intention Recognition

Human intention recognition is defined in general terms as the process of becoming aware of the intention of another and these intentions are inferred from actions carried out or from changes in the environment [16, 12, 17]. In the literature, intention recognition approaches can be classified according to two main configurations hinged on human-robot interactions as depicted in Figure 2.2:

1. *Intention recognition via observation of the user’s physical actions and the environment* (Figure 2.2a): the human and the robot may both perform tasks on the environment and there is some form of interaction between them.

2. *Intention recognition via observation of the user’s environment* (Figure 2.2b): the human interacts with the robot that in turn interacts with the environment.
Figure 2.2. Configuration of human-robot interactive systems using intention recognition. Circles with ‘H’ represent humans, ‘R’ represents robots, and ‘E’ represents the environment. For approaches grouped under (a), the human and robot both perform tasks on the environment whereas for (b), the environment is acted on only by the robot.

For the first configuration (Fig. 2.2 (a)), the human can act directly on the environment. Therefore, both the human and the environment can be observed to infer intentions. Sensors are used to observe the user’s physical actions and determine their intentions [29, 30, 31, 32]. For example, Kelley et al. [29] presented an approach that observed an individual using an RGB-D camera and with a neural network-based method they were able to predict their actions by analyzing their hand positions in relation to objects in the scene. In another work, Kelley et al. [30] proposed a framework that analyzed an individual and the objects they interacted with to improve the intention recognition capabilities of a socially-interactive robot in order to communicate more effectively with humans. Also, Zhu et al. [31] recognized human hand gestures modeled as hidden Markov models in order to command a robot for effective human-robot interaction. Consequently, the main underlying goal of this category of approaches is to develop effective socially-interactive robotic systems and the target population is usually able-bodied individuals.

On the other hand, for the second configuration, as depicted in Figure 2.2 (b), the human cannot act directly on the environment; they act on it via the robot. Sensors are used to observe the environment and the robot interacts with the human for determining their
intentions [33, 34, 35]. For example, Demeester et al. [33] presented a system that estimated the intent of a user using the sensor readings of their environment and the user’s commands so as to take corrective action during wheelchair maneuvering. The system in turn provided assistance that was tailored to the user’s driving ability. Similarly, Carlson and Demiris inferred the user’s intent while operating a powered wheelchair from their joystick input and the affordances of the local environment [35]. The main goal of this category of approaches is the development of robotic systems that function effectively in human environments in order to work in tandem with humans for achieving common goals.

Our survey of the state of the art finds that the second category is not fully explored. It is according to this configuration that the work in this dissertation belongs because it is more appropriate for dealing with persons with disabilities wherein it may be quite difficult for them to perform activities on the environment.

2.3 Object Pose Estimation

For autonomous or semi-autonomous robotic systems to effectively interact with and manipulate objects in their surroundings, accurate and robust perception is necessary. In order to correctly manipulate objects in a scene, accurate position and orientation information is required. Having knowledge of these object properties is of paramount importance for grasp planning and manipulation maneuvers across multiple domains including but not limited to assistive devices, socially-interactive robots, and industrial automation [36].

To this end, object pose estimation has been considerably addressed over the years in the computer vision and robotics literatures from two main categories of approaches. For the first category, the task of determining the pose of unknown objects was addressed by finding correspondences between 2D image features and model features from a database of known objects. The traditional procedure entailed estimating the model position and orientation that best agreed with a set of correspondences between the image and model. This has
been the de facto standard for years, for example [37, 38]. With no knowledge about the 3D information, these methods are error prone and return false positives (see Figure 2.3). Progress has been made by way of reconstructing 3D models from 2D data. For instance,

![Image](image-url)

Figure 2.3. Failure of pose estimation methods relying on 2D data. The image on the computer monitor matches with an object model in the model database. While the match is not inherently incorrect, it does not represent a scene object.

Collet et al. presented an online method for recognizing objects and their poses from images using a combination of RANSAC [39] and Mean Shift clustering [6] of image keypoints. Their method works by extracting image features from a sequence of images taken from different viewpoints and using these features for reconstructing a 3D object model, which is then used for matching to a database. They extended this work in [40] and [7] in an effort to achieve scalability and low latency. A similar approach was outlined by Sun et al. [41] whereby they jointly detected objects, estimated their pose, and recovered their 3D shape information from a single 2D image using a generalized Hough voting-based scheme [41]. Unfortunately, these methods rely on captured 2D information and an off-line training phase that learns
metric 3D models from multiple object views. Therefore, they work reliably only to the scope provided by the training data.

For the second category of approaches, pose estimation is based only on 3D information [42, 43]. With compact, low-cost 3D sensors such as the Microsoft Kinect [44] becoming more available, it is more appropriate and more accurate to utilize 3D data because it provides important geometrical information. Knowing the geometry of an object is important for pose estimation. As an example, Choi et al. [45] presented a voting-based approach that combined geometric and color information from an RGB-D camera and learned object models from multiple point clouds of objects. Also, Ye et al. estimated human body poses from 3D views by matching point cloud data with a database of pre-captured motion exemplars containing body surface models and corresponding skeleton body configurations [46]. However, both of these approaches rely on a model database and multiple object views for pose estimation. Recently, progress has been made whereby model databases were eliminated [43]. In this dissertation, our approach differentiates itself from the state of the art in the sense that we do not use databases containing predefined object models, there is no off-line training stage, and we utilize single view 3D data of unknown objects.
Object segmentation is the first computational step in our system flow. During segmentation, candidate objects are extracted from visual input data. In this chapter, we present some background information on segmentation in Section 3.1 as well as the details of the approach we adopt in this dissertation in Section 3.2.
3.1 Background

Image segmentation is the process of partitioning an image or organizing pixels into compact and expressive representations that emphasize important, interesting, or distinctive properties [47]. In other words, it is the collection of pixels that belong together based on some common property or predicate, for example, grouping proximate pixels that possess the same color or texture. The details of the representation depends on the task at hand, such as functional structures in medical images and objects in natural images. Segmentation is often the first indispensable step for mid-level vision. In terms of object segmentation, the task is to identify and outline individual objects in scenes. This is a challenging and considerably-addressed problem in computer vision as it is an important step in many applications, such as object recognition, object classification, and object retrieval.

Most existing object segmentation methods can be loosely classified into three categories: supervised methods, semi-supervised methods, and unsupervised methods. Supervised segmentation methods require some form of prior knowledge from training data for processing, for example the shape templates required for the approach of [48]. Semi-supervised methods use some form of interaction in order to segment images, for example the interactive graph cut methods of [49, 50]. Unsupervised methods rely on some form of feature grouping or clustering for segmentation such as the approach of [51] and it is according to this category that the segmentation approach adopted in this dissertation falls.

3.2 RGB-D Segmentation

Our aim is to capture a raw point cloud of a scene and extract data points belonging to candidate objects from it for further processing. For this dissertation, we are specifically interested in the segmentation of objects located on the top of tables. Tables are therefore regarded as horizontal planes that can support objects on them. Our segmentation approach (depicted in Figure 3.2) proceeds as follows.
Figure 3.2. Segmentation process followed in this work: Given an RGB-D point cloud, we first extract the table plane then extract the point clusters corresponding to candidate objects located on the table.

1. Range Filtering: Given a single view RGB-D point cloud of a table scene, we first remove points located outside of our target range, generally between 0.5 to 3.0 meters from the camera along the z-axis as indicated by the range filtering step of Figure 3.2. This significantly improves processing time by avoiding unnecessary computations. The gray regions of the images in Figure 3.3 indicate the filtered out areas.

2. Point Cloud Down-sampling: Next, we down-sample the point cloud using voxels in order to reduce the processing time of subsequent steps. During this step the size of a cloud is reduced by decomposing it into regularly-spaced grids. The centroid of all points falling within the boundaries of the grids are used as the new points for the point cloud. In this work, the grid size used for our experiments was $2\text{mm} \times 2\text{mm} \times 2\text{mm}$.

3. Normals Estimation: For every 3D point in the point cloud, an estimate of the normal to the best least-squares plane fit locally is calculated. These normals are then used for retrieving the planar model of the table.

4. Plane Model Estimation: Estimation of the table model coefficients is done using a RANSAC-based [39] approach. We retrieve the plane coinciding with a table-top by finding the plane model parameters that fits best with the captured point cloud, namely the model that possesses the most inliers (3D points satisfying the plane equation: $ax + by + cz + d = 0$). An illustration of this is shown in Figure 3.3 (b). The points belonging to the recovered table plane are used in subsequent processing.

\[ \text{described in more detail in Section 5.1.2} \]
Figure 3.3. Segmentation process adopted in this work. In (b) the green points belong to the table whereas the other points correspond to candidate objects. In (d), these objects are retrieved by performing Euclidean clustering.

5. Convex Hull Estimation & Polygonal Prism Extraction: The convex hull of a set of points $\mathcal{P}$ is the smallest convex set that contains $\mathcal{P}$, which essentially means that the set contains all of the line segments connecting every pair of points in $\mathcal{P}$. We find the convex hull of the inlier points retrieved by plane model estimation. This is then used as input for generating a polygonal prism of a specified height. The points within this
prism correspond to object clusters on the table. In this dissertation, we assume that all relevant objects fall within the height range of 2cm to 50cm.

6. Euclidean Cluster Extraction: In order to separate the set of candidate objects within the polygonal prism obtained from the previous step, Euclidean cluster extraction is performed. The points are split into regions according to an Euclidean distance and $kd$-tree [52] criterion whereby proximate points are grouped into similar clusters. An illustration of the object clusters retrieved is shown in Figure 3.3 (b). These resultant clusters are further processed to determine their object category and pose as described in subsequent sections.
CHAPTER 4

OBJECT CATEGORIZATION

Figure 4.1. Object Categorization Overview: The generic classes for object hypotheses are determined using multi-modal features.

Object categorization is the task of identifying instances of an object category within an image of the real world. It is a fundamental human ability that occurs somewhat effortlessly. For example, a child is able to derive the concept of “bowl” or “bottle” after seeing only a few examples. On the other hand, it is a significantly daunting task for robotic vision systems as there are manifold factors to consider and overcome such as variations in object pose, illumination, texture, clutter, and occlusion. As a result, object categorization is one
of the most difficult and well-studied topics in computer vision. Considerable progress has been attained over the years [11, 53, 54, 55, 56], yet no general and comprehensive solution exists.

Most object categorization approaches work by attempting to exploit the regularities of objects in images under varying viewpoint, illumination, clutter, and texture conditions. In other words, they employ representations or models to capture these characteristics in an effort to ascertain their category identities. These representations can be based on appearance or geometric cues and the models can either be generative or discriminative. However, several models for object categorization are solely built on using appearance information, such as color and texture. Appearance cues can successfully identify object classes up to a certain degree. When objects of different categories only differ in shape but not in texture, appearance-based methods reach their limits. Using this modality solely would not be sufficient for robotic applications that deal with complex scenes. We believe that the most effective approach is to capture various object properties using different modalities, for example, 2D and 3D features [1]. Evidence for the object category, obtained for appearance, color, and shape should be integrated to keep a balance between discrimination and generalization. Our aim is to incorporate recent advances in object representation and use optimal strategies so that we can achieve reliable categorization performance [57].

To this end, in this chapter we describe the various components of our object categorization approach. We first provide some background on the appearance-based features we utilize, namely Scale Invariant Feature Transform (SIFT) [38], described in Section 4.1.1, and Histogram of Oriented Gradients (HOG) [58], described in Section 4.1.2. These features have demonstrated superior performance for manifold tasks including but not limited to human detection [59], face recognition [60], image retrieval [61], and action recognition [62]. Following this, we provide some background information on the geometric feature we use, namely Fast Point Feature Histograms [63] in Section 4.1.3. This is followed by a description
of the Bag of Words (BoW) representation [64] in Section 4.1.4 and the Support Vector Machine (SVM) classifier in Section 4.1.5. We then present the details of our RGB-D Object Categorization approach in Section 4.2.

4.1 Background

4.1.1 Scale Invariant Feature Transform (SIFT)

Scale Invariant Feature Transform (SIFT) [38] is an ubiquitous method for extracting distinctive invariant local features from images that can be used to perform matching between different views of a scene or object. The method was proposed by Lowe in 1999 [65] and shown to be effective for improving object recognition, object tracking, 3D scene reconstruction and a host of other tasks. SIFT has been demonstrated to be robust to scale, rotation, illumination, and viewpoint variations.

SIFT transforms an image into a collection of highly distinctive SIFT keypoint descriptors which can be used for matching against other descriptors for finding similar objects. Particularly, SIFT descriptors are first extracted from a set of template images and stored in a database. A query image is matched by individually comparing each of its descriptors to descriptors from the database and finding candidate matches based on Euclidean distance between descriptor vectors. Correct matches are filtered from the set of candidate matches by identifying subsets of descriptors that are congruent with the object and its location, scale, and orientation in the query image. Figure 4.2 shows an example of SIFT descriptor matching for two images. According to Lowe [38], the major stages of computation for extracting SIFT key-point descriptors are as follows:

1. Scale-space Extrema Detection: The first step of SIFT involves the identification of the location and scale of keypoints. The locations must be repeatable under differing views of an object and this is accomplished by searching for stable features across all
Figure 4.2. An example of SIFT descriptor matching. The coca-cola can on the right is matched to a coca-cola can on a table.

possible scales using a scale space. The scale space of an image is defined as follows:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$  \hspace{1cm} (4.1)

where $L(x, y, \sigma)$ is the scale space function produced by the convolution of a variable-scale Gaussian $G$ with an image $I$. The convolution is executed at different scale and the difference of successive Gaussian-smoothed images $D(x, y, \sigma)$ are taken as follows:

$$D(x, y) = L(x, y, k_i \sigma) - L(x, y, k_j \sigma)$$

$$= (G(x, y, k_i \sigma) - G(x, y, k_j \sigma)) * I(x, y)$$  \hspace{1cm} (4.2)

where $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$, and $k_i$ and $k_j$ are constant multiplicative factors of two adjacent scales $i$ and $j$. The initial keypoints are determined as local mimima
and maxima of the difference of Gaussians $D(x, y, \sigma)$ by comparing each pixel in the scale space to its 26 neighbors: 8 from its own scale, and 18 from the scales above and below. If the pixel’s value is greater or lower than its neighbors, it is selected as a candidate keypoint.

2. Keypoint Localization: Many candidate keypoints are retrieved via the previously outlined step. For keypoint localization, the main goal is to reduce the size of the candidate keypoints set by rejecting those that have low contrast and those that are poorly defined along edges, because such points are unstable. Keypoints with low contrast are rejected by comparing the value of the second-order Taylor expansion $D(\mathbf{x})$ computed at the extremum $\hat{\mathbf{x}}$ with the value of the original candidate, where:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2} \mathbf{x}$$

(4.3)

and $\hat{\mathbf{x}}$ is the derivative with respect to $\mathbf{x} = (x, y, \sigma)$ as follows:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2} \frac{\partial D}{\partial \mathbf{x}}$$

(4.4)

If the value at $\hat{\mathbf{x}}$ is less than a predetermined threshold, the candidate is rejected. Additionally, keypoints detected along edges are eliminated if the ratio of the principal curvatures along the edge is greater than a predetermined threshold.

3. Orientation Assignment: In order to achieve rotation invariance, the keypoint is assigned a consistent orientation based on the gradient directions around the keypoint. This orientation is attained in a scale-invariant manner whereby the closest image scale of the keypoint $L(x, y, \sigma)$ is used for calculations. The gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are calculated from pixel differences for an image sample $L(x, y)$ at
this scale as follows:

\[
m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2} \tag{4.5}
\]

\[
\theta(x, y) = \tan^{-1}((L(x + 1, y) - L(x - 1, y))/(L(x, y + 1) - L(x, y - 1))) \tag{4.6}
\]

A histogram of orientations is constructed from the gradient orientations of points within a neighborhood around the keypoint. This histogram has 36 bins covering the full 360° range of rotations and each sample is weighted by its magnitude and a Gaussian-weighted circular window. The peaks in this histogram correspond to the dominant gradient orientations. The highest peak is chosen as the representative orientation as well as any other peak within 80% of its value, for creating the final keypoint.

4. Keypoint Descriptor Computation: A distinctive and reliable descriptor representation of the keypoints found in the previous step must be computed in order to match the keypoint to other keypoints for recognition. The computation of the SIFT descriptor is carried out in the neighborhood of a keypoint and at the scale and orientation previously determined. A set of 16 orientation histograms with 8 bins each are constructed using the gradient orientation information from 4 × 4 pixel neighborhoods. As previously done, orientation values are weighted by their magnitudes as well as a Gaussian circular window. The values of these histograms are then concatenated to create the 128-element SIFT descriptor vector. This vector is then normalized to unit length to increase its invariance to affine transformations.

4.1.2 Histograms of Oriented Gradients (HOG)

Histograms of Oriented Gradients (HOG) are feature descriptors widely used for object detection and object recognition. They were first proposed by Dalal and Triggs in 2005 [58]
as the feature representation for their human detection system with their primary advantage being that they maintain invariance to geometric and photometric transformations, except object orientation. The essential premise behind HOG is that local object appearance and shape can be described well by the distribution of local intensity gradients without the exact knowledge of the corresponding gradient positions [58]. This is achieved by dividing a RGB image into small spatial regions, called cells, and for each cell, a 1-D histogram of gradient orientations for the pixels within the cell is constructed. The combination of these histograms represents the descriptor. However, for improved accuracy, the local histograms are contrast-normalized by accumulating a measure of the intensity across a larger image region, called a block, and then using this value to normalize all cells within the block. Histograms of oriented gradients can be computed with the following steps:

1. Gradient Computation: The first step of HOG involves calculating the gradient orientations of pixels. An input image is first smoothed using a Gaussian function followed by convolution with a discrete derivative mask. It was shown that simple 1-D \([-1, 0, 1]\) masks worked best as larger masks decreased performance [58]. For color images, the gradient is calculated for each channel, and the one with the largest norm is chosen as the pixel’s gradient vector.

2. Orientation Binning: In this step, each pixel calculates a magnitude-weighted vote for a gradient orientation histogram channel. The votes are accumulated into 9 orientation bins defined over 8 × 8 pixel regions called cells and are spaced evenly over the 0° – 180° range.

3. Descriptor Blocks: Due to the fact that gradient strengths vary over a wide range, they must be locally normalized to improve performance. This involves grouping cells together into larger, spatially connected blocks. The vector of all components of the normalized cell responses from all the blocks forms the final HOG descriptor. The
blocks overlap thereby contributing more than one to the final descriptor. There are
main two types of block geometries: rectangular R-HOG blocks and circular C-HOG
blocks. R-HOG blocks are quite similar to SIFT descriptors [38] and are typically
square grids represented by $3 \times 3$ blocks of $6 \times 6$ pixel cells with 9 histogram bins.
C-HOG blocks are similar to center-surround coding structures and are usually repre-
sented by 2 radial bins and 4 angular bins. A visual example of the HOG descriptor is
shown in Figure 4.3.

![Input Image](image1.png) ![Visual HOG result](image2.png)

Figure 4.3. Visual example of the HOG descriptor.

4. Block Normalization: The blocks previously described are normalized by either of 4
methods. Let $v$ be an unnormalized vector, $\|v\|_k$ its $k$-norm for $k = 1, 2$ and $\epsilon$ be some
small constant. Then, the normalizing factor for HOG blocks can be either of the
following: 1) L2-norm $\rightarrow \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}}$, 2) L1-norm $\rightarrow \frac{v}{\|v\|_1 + \epsilon}$, 3) L1-sqrt $\rightarrow \sqrt{\|v\|_1^2 + \epsilon}$,
and 4) \( L2-hys \rightarrow L2-norm \) followed by limiting the values of \( v \) to 0.2 and re-normalizing.

Except for L1-norm, all the methods demonstrated similar performance [58].

For more details on the HOG descriptor, the reader is referred to the work by Dalal and Triggs [58].

4.1.3 Fast Point Feature Histograms (FPFH)

Fast Point Feature Histograms (FPFH) are point feature representations that capture the neighborhood surface characteristics of a three-dimensional point. They were introduced by Rusu et al. [63] as the optimized counterpart of the computationally-heavy Point Feature Histograms (PFH) [66] which was proposed earlier. To completely understand the inner workings of FPFH, we first provide a description of PFH.

According to Rusu et al. [66], Point Feature Histograms encode the geometrical properties of a neighborhood \( \mathcal{M} \) around a point \( p_i \) by generalizing its mean curvature using a multi-dimensional histogram of values [66]. This higher dimensional space produces a descriptive signature for feature representation that is invariant to the pose of the underlying surface and noise present in the neighborhood. The representation is based on the relationships between points in the neighborhood \( \mathcal{M} \) and their normals\(^1\). More precisely, it captures the surface variations by factoring in all the interactions between the directions of the estimated normals and consequently, becomes dependent on the quality of the surface normal estimations at each point.

The variations of the surface normals are captured by analyzing each pair of points \( p_i \) and \( p_j \) from the neighborhood \( \mathcal{M} \). The first step in producing the Point Feature Histogram for a point \( p_i \) involves estimating all the surface normals \( n_i \) from the points in \( \mathcal{M} \). Subsequently, to calculate the relative difference between two points \( p_i \) and \( p_j \) and their respective normals \( n_i \) and \( n_j \), a Darboux coordinate frame\(^2\) is defined at one of the points. The frame is uniquely

---

\(^1\)A normal is a vector that is perpendicular to the tangent plane of a surface at a 3D point \( p_i \)

\(^2\)A Darboux coordinate frame is a natural moving frame constructed on an oriented surface in 3D space.
defined as follows:

\[
\text{if: } \cos^{-1}(n_i \cdot p_{ji}) \leq \cos^{-1}(n_j \cdot p_{ij}), \quad p_{ji} = p_j - p_i, \quad p_{ij} = p_i - p_j \\
\text{then } \begin{cases} 
    p_s = p_i, n_s = n_i \\
    p_t = p_j, n_t = n_j
\end{cases}
\]

\[
\text{else } \begin{cases} 
    p_s = p_j, n_s = n_j \\
    p_t = p_i, n_t = n_i
\end{cases}
\]  

(4.7)

where \( p_s \) is defined as the source point and \( p_t \) as the target whereby for the source point, the angle between its normal and the line connecting the two points is minimal. The Darboux frame origin is thus defined at \( p_s \) as follows:

\[
\begin{cases} 
    u = n_s \\
    v = u \times \frac{(p_t - p_s)}{||p_t - p_s||_2} \\
    w = u \times v
\end{cases}
\]  

(4.8)

Figure 4.4. Illustration of the Darboux frame and the PFH features for a pair of points \( p_s \) and \( p_q \) with their respective normals [66].
Accordingly, the difference between two normals \( n_s \) and \( n_t \) using the Darboux \( uvw \) frame as shown in Figure 4.4 can be expressed as a set of angular features as follows:

\[
\begin{align*}
\alpha &= v \cdot n_t \\
\phi &= u \cdot \frac{p_t - p_s}{d} \\
\theta &= \tan^{-1}(w \cdot n_t, u \cdot n_t)
\end{align*}
\]

where \( d \) is the Euclidean distance between the two points \( p_s \) and \( p_t \). The quadruplet \( \langle \alpha, \phi, \theta, d \rangle \) is calculated for every pair of points in the neighborhood \( M \). To create the final PFH representation for a query point \( p_i \), the set of all quadruplets is binned into a histogram whereby each feature’s value range is divided into \( b \) subdivisions, and the frequency of occurrences is calculated for each subinterval. The histogram therefore possesses \( b^4 \) bins.

![PFH Influence Neighborhood](image1)

![FPFH Influence Neighborhood](image2)

Figure 4.5. The influence regions for a Point Feature Histogram and a Fast Point Feature Histogram. (adapted from the respective figures in [66] and [63]). (a) For PFH, the query point indicated in red and its \( k \)-neighbors in blue are fully interconnected for calculations. (b) For FPFH, each query point is connected only to its direct \( k \)-neighbors (enclosed by the blue region). Each direct neighbor is then connected to its own neighbors and the constructed histograms are weighted along with the histogram of the query point to form the FPFH.

The computational complexity of calculating Point Feature Histograms for a point cloud with \( n \) points is \( O(nk^2) \), where \( k \) is the number of neighbors for a point. For real-time or near
real-time applications, this is unacceptable; hence the relevance of Fast Point Feature Histograms [63]. FPFH reduces the complexity of the algorithm to $O(nk)$, while still possessing the same discriminative power of PFH. The histogram feature computation is simplified as follows:

1. For each query point $p_q$, a set of triplets $\langle \alpha, \phi, \theta \rangle$ between itself and its neighbors in $\mathcal{M}$ are computed as shown in the Equation 4.9 which is hereby referred to as Simplified Point Feature Histogram (SPFH).

2. For each point, its $k$ neighbors are redetermined and their SPFH values are used to weight the resultant histogram according to Equation 4.10.

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k} \sum_{i=1}^{k} \frac{1}{\omega_k} \cdot SPFH(p_k)$$  \hspace{1cm} (4.10)

The weight $\omega_k$ represents the distance between the query point $p_q$ and a point $p_k$ from the neighborhood $\mathcal{M}$ as shown in Figure 4.5 (b). Unlike the Point Feature Histogram, the algorithm for calculating a FPFH for a query point $p_q$ first estimates the SPFH values for pairs formed between itself and its neighbors, illustrated as red lines in Figure 4.5 (b). This is done for all the points in the point cloud, then a re-weighting of the SPFH values is done using the values of the points in the neighborhood. This creates the FPFH feature which differs from PFH by not considering all the point pairs in the neighborhood $\mathcal{M}$ of a query point $p_q$ but including additional point pairs outside the radius of $\mathcal{M}$ to recover some lost precision. Additionally, the resultant histogram is simplified by de-correlating the values and creating $d$ separate histograms, one for each feature dimension, and concatenating them. FPFH uses 11 binning subdivisions to create an overall histogram with dimension 33. For more details on FPFH, the reader is directed to the original work by Rusu et al. in [66] and [63].
4.1.4 Bag-of-Words Representation (BoW)

Inspired by text categorization methods, the bag-of-words (BoW) model is by far one of the most common representations used for object categorization in the computer vision and robotics literatures [67, 68, 53, 69, 70]. The representation corresponds to a histogram of the number of occurrences of particular features or image patterns (such as HoG or SIFT features described in previous sections) in a given image. Construction of the bag-of-words model is described as follows.

First, features (usually in the form descriptors) are collected from an image either by dense sampling or interest point detection using a detector such as SIFT [38]. Features chosen are often descriptive enough to be discriminative at the category level. These features are often represented by vectors in a high dimensional space and there may be thousands for one image. For efficient handling of these features, a “visual vocabulary” is constructed, which is a set of cluster centers obtained via quantization of the feature vectors. The k-

![Figure 4.6. Illustration of the construction of Bag-of-Words models](image-url)
means algorithm is often used to determine these clusters from training data. Features from multiple images are processed to determine \( k \) representative clusters. These clusters then become the “visual words” of the visual vocabulary. Using this vocabulary, feature descriptors from images are assigned to the closest cluster in terms of some distance metric, usually Euclidean. As shown in Figure 4.6, a histogram of the occurrences of these visual words is constructed and this is the image’s bag-of-words representation. By doing this, images can be compared by determining the distance between bag-of-words vectors or the vectors can be by a classifier such as Support Vector Machines for classification. The latter approach is the one followed in this dissertation.

4.1.5 Support Vector Machines (SVMs)

A Support Vector Machine (SVM) is a discriminative classifier that analyzes data and discovers patterns. It belongs to a specific family of supervised learning algorithms referred to as kernel-based methods. With these methods, linear combinations of a kernel function evaluated on training data forms the foundation of data predictions. SVMs are ubiquitous and commonly used for solving a host of problems in classification and regression analysis. Their current and most common incarnation was proposed by Cortes and Vapnik in 1995 [71]. Although they are fundamentally binary classifiers, namely, they are only capable of distinguishing between two different data classes, extensions have been constructed in the context of multi-class classification.

More formally, an SVM finds the hyperplane\(^3\) (or decision boundary) or set of hyperplanes that linearly separates classes of data. The determination of the SVM hyperplane parameters is formulated as follows. Given a set of training vectors \( \mathbf{x}_i \in \mathbb{R}^n \) and a binary set of classes

---

\(^3\)A hyperplane of an \( n \)-dimensional space is a subset of dimension \( n - 1 \) that separates the space into two half spaces.
Let the two classes $c_1$ and $c_2$ be linearly separable by a hyperplane described by the following equation:

$$w^T x + b = 0 \quad (4.12)$$

The term $w$ is the hyperplane’s normal and $b$ is a bias parameter. There exists at least one choice of $w$ and $b$ such that:

$$\begin{cases} 
  w^T x_i + b > 0, & \text{for } y_i = 1 \\
  w^T x_i + b < 0, & \text{for } y_i = -1 
\end{cases} \quad (4.13)$$

As a result, the decision function becomes $f(x) = sign(w^T x + b)$ with $x$ as test data.

Figure 4.7. A hyperplane dividing two linearly separable classes (black points - $c_1$, white points - $c_2$). The support vectors (data vectors lying on the boundary of the margin between classes) for both classes are highlighted.
The goal of the Support Vector Machine is to find the parameters \( w \) and \( b \) such that the smallest distance from the hyperplane to any of the data samples, referred to as the margin, is maximized. By expanding on the constraints provided in Equation 4.13, we have:

\[
\begin{align*}
\begin{cases}
w^T x_i + b &\geq d_+, \text{ for } y_i = 1 \\
w^T x_i + b &\leq d_-, \text{ for } y_i = -1
\end{cases}
\end{align*}
\] (4.14)

where \( d_+ \) and \( d_- \) represent the outer boundary of the margin. The maximal distance between \( w^T x_i + b \) is \( \pm 1 \). By choosing \( w \) and \( b \) with the maximal margin, \( d_+ = 1 \) and \( d_- = 1 \), the margin becomes:

\[ d_+ + d_- = 2 \] (4.15)

We obtain Equation 4.16 by combining Equations 4.14 and 4.15 to obtain:

\[ y_i (w^T x_i + b) \geq 1 \] (4.16)

With Equation 4.16, the misclassification of data samples is penalized in proportion to their distances from the hyperplane or decision boundary.

Thus, the margin is ascertained by the points that are located closest to the hyperplane. Maximizing the margin therefore entails building a better separation boundary between classes so that the classification error is minimal. The subset of data points lying on this boundary are known as the support vectors of the classification. In the example of Figure 4.7, the margin is \( \frac{2}{||w||} \). In order to find the maximum margin, \( ||w|| \) has to be minimized, which in turn is equivalent to minimizing \( ||w||^2 \). This problem is formally written as an optimization problem as shown in the following equation:

\[
\min_{\frac{1}{2} ||w||^2} \\
\text{subject to: } y_i (w^T x_i + b) \geq 1 (\forall \text{ data points } x_i)
\] (4.17)
This optimization problem is solved best as a Lagrangian formulation because the constraints are easier to handle and the training data only appears in the form of dot products between vectors [72]. Using Lagrangian multipliers $\alpha_i \geq 0$, the Lagrangian function is formulated as:

$$L_P = \frac{1}{2} ||w||^2 - \sum_i \alpha_i y_i (w^T x_i + b) + \sum_i \alpha_i$$  \hspace{1cm} (4.18)

With respect to $w$ and $b$, minimization proceeds by setting the derivatives of $L_p$ for $w$ and $b$ to 0 to produce:

$$\begin{cases} 
  w = \sum_i \alpha_i y_i x_i \\
  0 = \sum_i \alpha_i y_i
\end{cases} \Rightarrow L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$  \hspace{1cm} (4.19)

$L_D$ is commonly known as the dual form of the primary Langrangian function $L_P$ and $K(x_i, x_j)$ is known as a kernel function.

As previously mentioned, SVMs belong to a specific algorithmic family known as kernel-based methods. This is so because data sets are not always linearly separable. As a result, SVMs map the data from its original data space to a higher dimension feature space which may have a high or even infinite dimension in order to determine the separating hyperplane between classes. This is done using a kernel function as shown in Equation 4.19. A kernel function is any non-negative real-valued function that satisfies the following properties:

1. Symmetric: $K(x_i, x_j) = K(x_j, x_i)$

2. Positive Semi-definite $\forall \ x$ in kernel matrix $K_{i,j}$

The most commonly used kernels are as follows:

- Linear kernel: $K(x_i, x_j) = x_i^T x_j$

- Polynomial kernel: $K(x_i, x_j) = (x_i^T x_j + \delta)^d$, with $\delta > 0$
• Sigmoidal kernel: \( K(x_i, x_j) = \tanh(\kappa x_i^T x_j + \delta)^d \), with \( \kappa > 0, \delta > 0 \)

• Radial Basis Function (RBF) kernel: \( K(x_i, x_j) = e^{-\gamma \cdot d(x_i, x_j)} \) with \( \gamma > 0 \), where \( d \) is a distance metric which can take any of the following forms:
  
  \[-d(x_i, x_j) = \sum_k |x_{i,k} - x_{j,k}|^{\frac{1}{2}} \text{ for the Sublinear RBF kernel,} \]
  
  \[-d(x_i, x_j) = \sum_k |x_{i,k} - x_{j,k}|^1 \text{ for the Laplacian RBF kernel,} \]
  
  \[-d(x_i, x_j) = \sum_k |x_{i,k} - x_{j,k}|^2 \text{ for the Gaussian RBF kernel.} \]

By using kernel functions, SVMs can be formulated entirely in terms of scalar products in the higher dimension feature space. Thus, the decision function becomes:

\[
f(x) = \text{sign}(w^T x + b) \Rightarrow \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\]  

(4.20)

In this dissertation, we use a Support Vector Machine with a Radial Basis Function kernel for our experiments. SVM works in a vector space, therefore using the previously-described Bag-of-Words approach is highly applicable because it provides a vector representation for objects of interest. Furthermore, in order to apply SVMs for multi-class problems, we adopt the one-against-all strategy for classification.

### 4.2 RGB-D Object Categorization

We process the RGB-D object point clusters obtained via object segmentation to identify instances of object categories. This process is difficult because an object category must be determined despite inhibiting factors like occlusion, noise, background clutter, and multiple objects. Several models for object categorization are solely built on using appearance information. Appearance cues can successfully identify object classes up to a certain degree. When objects of different categories only differ in shape but not in texture, appearance-based methods reach their limits.
To address this issue, we employ a categorization method based on multiple cues: appearance, 2D contour shape, and 3D shape. This allows us to keep a balance between discrimination and generalization. Our procedure as depicted in Figure 4.8 is as follows. From a 2D projection of the object point cloud, we extract SIFT [38] and HoG [58] features for appearance and contour shape respectively. The 3D shape properties are obtained by using Fast Point Feature Histograms (FPFH) [63]. For the final object representation, the Bag-of-Words (BOW) model is employed [64] for each cue. The BOWs are produced using
visual vocabularies constructed offline on features from point clouds in our RGB-D Dataset described in detail in Chapter 7.

For classification, we use Support Vector Machines (SVMs) with a radial basis function kernel for the BOW vectors obtained. There is a classifier for each cue outlined above, namely a FPFH classifier, a SIFT classifier, and a HOG classifier. These classifiers are trained on point cloud data from our dataset which contains data for 11 object categories, which is done offline. For cue integration, we adopt the ensemble of classifiers paradigm. With this paradigm, the $k$ class confidence outputs provided by each cue classifier are concatenated to create a new $3k$-element data vector. Using this data along with the correct category labels, another Support Vector Machine is trained and it is this classifier that is responsible for making the final classification decision.

As previously mentioned, object categorization is a challenging issue in computer vision as there can be many variations to an object’s appearance. For the intention recognition approach described in this dissertation, correct category labels are necessary as incorrect predictions can lead to communication of the wrong task to the robot. To ensure that the correct category labels are obtained, we include a confidence evaluation component (or failure recovery as in [73]) whereby we determine the overall classifier confidence in its predictions. If there is high confidence in the classification, the predicted object category is simply returned for the given point cloud. However, if the confidence is low, the user is prompted to select the correct object class from a list of category choices determined by the closest $k$ matches.

We base this confidence evaluation component as shown in Figure 4.8 on the entropy of the classifier output vector, which is in itself a probability distribution over the 11 possible object classes. Entropy is a measure of the amount of uncertainty or disorder in a distribution and is given by the following equation:

$$H_\alpha(P) = \frac{1}{1 - \alpha} \log_2 \left( \sum_{i=1}^{n} p(x_i)^\alpha \right)$$  \hspace{1cm} (4.21)
This particular form of entropy is known as Rényi’s Entropy of order $\alpha$, where $\alpha \geq 0$ and we utilized an $\alpha$ value of 2. High entropy values indicate uniformity whereas low values indicate uncertainty. Equation 4.21 is a generalization of the more common Shannon entropy. We believe that finding the entropy of the class prediction distribution $P(c)$, where $c$ is the category, is a reasonable measure of the combined classifier’s confidence. Thus, the confidence function $\Phi(\cdot)$ is defined as $\Phi = 1 - H_\alpha[\hat{P}(c)]$. Higher values of $\Phi$ indicate greater confidence and vice versa. We establish an acceptable confidence by using a threshold $T$. Therefore, if $\Phi > T$, the classifier confidence is acceptable whereas if $\Phi \leq T$, the user is prompted for the correct class.
CHAPTER 5

OBJECT POSE ESTIMATION USING SUPERQUADRICS

Figure 5.1. Object Pose Estimation Overview: The 3D position and orientation of objects are determined using superquadrics.

Using 3D data brings its challenges\(^1\). Massive amounts of 3D data must be processed rapidly in order for robotic systems to be responsive. Autonomous robotic systems equipped with 3D sensors can acquire point cloud data at an increasingly high rate. Executing common tasks such as scene segmentation and 3D reconstruction on massive amounts of 3D data is computationally expensive and requires a lot of computational time for processing, which is

\(^1\)Portions of this work have been published (Duncan et al. [9]) and are utilized with the permission of the publisher as shown in Appendix A.
unacceptable for real-time or near real-time robotics [74]. For a responsive robotic system, the latency of frequently executed tasks should be low. For example, as the number of points in a point cloud increases, the computational time required for processing greatly increases. For a system requiring user interaction, this is unacceptable.

![RGB Image](image1.png) ![RGBD Point cloud](image2.png) ![Missing points of object due to occlusion](image3.png) ![Recovered superquadric](image4.png)

Figure 5.2. Case for superquadrics. With their tri-axis symmetry, superquadrics are able to recover the occluded portions of point clouds.

In this chapter, we address the issue of finding the shape and pose information of unknown objects in a rapid manner. This is done from single view point clouds where only the front part of the object is visible and assumptions must be made about the back side in order to correctly manipulate the object [75]. We attempt to handle this issue by employing superquadrics, which are compact parametric shapes with tri-axis symmetry that are appropriate for modeling frequently encountered objects in domestic settings. Moreover, the symmetric quality of superquadrics facilitates the prediction of the shape of occluded object parts by assuming global symmetry [76] (see Figure 5.2).

Over the years, different models have been introduced for 3D shape recovery such as spherical harmonics and geometric icons, but superquadrics are conceivably the most appropriate for our tasks [42]. Their compact shape can be described with a small set of parameters.
thereby facilitating the description of a wide variety of different basic shapes such as spheres, cylinders, and cuboids (see Figure 5.3). Superquadrics have been used for object approximation [76, 77], object detection [42], novelty detection [78], object segmentation [79, 80, 81], and collision detection [82]. Our goal is to quickly find the superquadric that best fits an unorganized point cloud representing an object hypothesis. This is a major issue inherent with employing superquadrics. Their parameters must be minimized in a least-squares fashion and this process can be computationally expensive if point clouds are large. We address this by proposing a multi-scale voxelization strategy. With this strategy, we are able to estimate the pose of an object using superquadrics in a robust and computationally-efficient manner without sacrificing accuracy.

5.1 Background

5.1.1 Superquadrics

Superquadrics are a family of parametric shapes that include superellipsoids, super-toroids, and superhyperboloids with one and two parts. They were introduced for use in the computer graphics community by Barr [83] in 1981 and are appealing for robotic applications by nature of their definition. In this work, we focus on the superellipsoid which is useful for a volumetric part-based description. The terms superellipsoid and superquadric are often used interchangeably in the literature and we follow the same terminology here. Given the parameters that define a superquadric, the shape and pose information can be easily extracted as well as volumes and moments of inertia. They are compact in shape and have a closed surface. Moreover, superquadrics exhibit tri-axis symmetry, which is a characteristic well approximated by many household objects [75].

Superquadrics can be defined in an object centered coordinate system with five variables and in a general coordinate system by eleven independent variables. Equation 5.1 defines a superquadric surface by a 3D vector $\mathbf{x}$ which originates in the coordinate center and sweeps
out a closed surface when the angles $\eta$ and $\omega$ change within their given ranges.

$$\mathbf{x}(\eta, \omega) = \begin{bmatrix}
a_1 \cos^{\epsilon_1}(\eta) \cos^{\epsilon_2}(\omega) \\
a_2 \cos^{\epsilon_1}(\eta) \cos^{\epsilon_2}(\omega) \\
a_3 \sin^{\epsilon_1}(\eta)
\end{bmatrix}$$

$$\frac{-\pi}{2} \leq \eta \leq -\frac{\pi}{2},$$

$$-\pi \leq \omega \leq \pi$$

(5.1)

These angles correspond to the latitude and longitude angles of the vector $\mathbf{x}$ when expressed in spherical coordinates. The parameters $a_1, a_2,$ and $a_3$ define the superquadric size along the $x, y,$ and $z$ dimensions, whereas $\epsilon_1$ and $\epsilon_2$ correspond to the squareness in the latitudinal and longitudinal planes respectively. In this work, $\epsilon_1$ and $\epsilon_2$ range from 0.1 to 1.0. Equation 5.1 is commonly used for rendering superquadrics.

Using the equality $\cos^2(\alpha) + \sin^2(\alpha) = 1,$ the parameters $\eta$ and $\omega$ can be eliminated to obtain the following implicit form of the superquadric equation for an object-centered coordinate system:

$$\left( \left( \frac{x}{a_1} \right)^{\frac{2}{\epsilon_1}} + \left( \frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left( \frac{z}{a_3} \right)^{\frac{2}{\epsilon_1}} = 1$$

(5.2)

As a result, the function $F$ can be defined as shown in Equation 5.3.

$$F(x, y, z) = \left( \left( \frac{x}{a_1} \right)^{\frac{2}{\epsilon_1}} + \left( \frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left( \frac{z}{a_3} \right)^{\frac{2}{\epsilon_1}}$$

(5.3)

This function is referred to as the inside-outside function because it determines where a given 3D point lies relative to the superquadric surface. For a given point $\mathbf{x} = [x, y, z]$, $F(\mathbf{x}) = 1$ if the point is on the superquadric surface, $F(\mathbf{x}) < 1$ if the point is inside the superquadric, and $F(\mathbf{x}) > 1$ if the point is outside the superquadric.
In order to recover superquadrics from real world scenes, they must be represented in
general position and orientation. This requires additional parameters for expressing the
rotation and translation of the superquadric relative to the center of the world coordinate
system. The homogeneous coordinate transformation $T$ is used to transform the 3D points
in the object-centered coordinate system into the world coordinates as shown in Equation
5.4:

$$
\begin{bmatrix}
    x_w \\
    y_w \\
    z_w \\
    1
\end{bmatrix}
= T
\begin{bmatrix}
    x_o \\
    y_o \\
    z_o \\
    1
\end{bmatrix}
$$

(5.4)

where

$$
T =
\begin{bmatrix}
    n_x & o_x & a_x & p_x \\
    n_y & o_y & a_y & p_y \\
    n_z & o_z & a_z & p_z \\
    0 & 0 & 0 & 1
\end{bmatrix}
$$

(5.5)

Therefore, for any given point, the transformation $T$ first rotates that point (by the pa-
rameters $n, o$ and $a$) and then translates it by $(p_x, p_y, p_z, 1)^T$. However, the points must
be expressed in object-centered coordinates, so the transformation $T^{-1}$ must be performed
where $T^{-1}$ is

$$
T^{-1} =
\begin{bmatrix}
    n_x & n_y & n_z & -(p_x n_x + p_y n_y + p_z n_z) \\
    o_y & o_y & o_z & -(p_x o_x + p_y o_y + p_z o_z) \\
    a_z & a_y & a_z & -(p_x a_x + p_y a_y + p_z a_z) \\
    0 & 0 & 0 & 1
\end{bmatrix}
$$

(5.6)
By substituting Eqs 5.4 and 5.6 into Eq. 5.3, the expanded implicit form of the superquadric equation for objects in general position and orientation is given by:

\[
F(x_w, y_w, z_w) = \left[ \left( \frac{n_x x_w + n_y y_w + n_z z_w - p_x n_x - p_y n_y - p_z n_z}{a_1} \right)^{\frac{2}{\epsilon_2}} + \right. \\
\left. \left( \frac{o_x x_w + o_y y_w + o_z z_w - p_x o_x - p_y o_y - p_z o_z}{a_2} \right)^{\frac{2}{\epsilon_1}} + \right. \\
\left. \left( \frac{a_x x_w + a_y y_w + a_z z_w - p_x a_x - p_y a_y - p_z a_z}{a_3} \right)^{\frac{2}{\epsilon_1}} \right] \epsilon_1 \epsilon_2 \tag{5.7}
\]

where the variables \((a_1, a_2, a_3)\) are the scaling dimensions along the \(x\), \(y\), and \(z\) axes of the superquadric, \((\epsilon_1, \epsilon_2)\) are the factors which determine the superquadric’s shape ranging from from 0.1 to 1, and \((n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z, p_x, p_y, p_z)\) are the twelve parameters of the homogeneous transformation matrix that is a result of a rotation and translation of the world coordinate frame. The Euler angles \(\phi, \theta, \psi\) are used to represent the rotational part of the transformation matrix \(T\). This is the form used for minimization in this work. Therefore, the eleven variables that define a superquadric in general position and orientation are \(\Lambda = \{a_1, a_2, a_3, \epsilon_1, \epsilon_2, \phi, \theta, \psi, p_x, p_y, p_z\}\). To handle global superquadric deformations, the tapering parameters \(k_x\) and \(k_y\) are used, but we do not exploit this feature in this work. Examples of superquadrics for different values of \(\epsilon_1\) and \(\epsilon_2\) are shown in Figure 5.3, demonstrating how they can be used to describe a wide range of shapes.

The expression in Equation 5.8 must be minimized where the multiplier \(\sqrt{a_1 a_2 a_3}\) enforces the recovery of the smallest superquadric and the exponent \(\epsilon_1\) promotes faster convergence as it makes the error metric independent of the shape factor [76].

\[
\min_k \sum_{k=0}^{n} \left( \sqrt{a_1 a_2 a_3} \left( F^{\epsilon_1} \left( x_{k}; \Lambda \right) - 1 \right) \right)^2 \tag{5.8}
\]
The Levenberg-Marquardt algorithm [84] is used to recover the parameter set \( \Lambda \) that best fits a given set of points \( \mathbf{x}_k \) in a least-squares minimization. The Levenberg-Marquardt algorithm is a standard technique used to solve non-linear least squares problems and it is a combination of the gradient descent and the Gauss-Newton methods. An important aspect to this minimization is the initial parameter set used. A good initialization is crucial to the success of the superquadric fitting process. Therefore, we use the initial pose given via the eigenvalue decomposition of the point cloud. The initial shape used is an ellipsoid \( (\epsilon_1 = 1, \epsilon_2 = 1) \) and the superquadric scale factors are based on the dimensions of the cloud itself.

Figure 5.3. Examples of superquadrics for different shape parameters \( \epsilon_1 \) and \( \epsilon_2 \). In this work, we focus on recovering shapes found within the range 0.1 to 1.0.
5.1.2 Voxels

Voxels (volumetric pixels) are 3D elements representing values on a regularly-spaced grid in 3D space. Each voxel value represents an approximation of the centroid of all the points falling within the bounds of the corresponding grid cell (see Figure 5.4). Voxels are commonly used for terrain representation in games and volumetric imaging in medicine.

![2D representation of a voxel](image)

Figure 5.4. 2D representation of a voxel. Voxels divide the 3D space into uniform cells, typically cubes, and their values represent the centroid of the points falling within its bounds.

5.2 Multi-scale Voxelization

In this section, we introduce an automatic coarse-to-fine voxelization scheme for superquadric fitting. Fitting superquadrics to point cloud data is a computationally expensive task and the bottle neck of this process is the iterative Levenberg-Marquardt [84] algorithm minimizing eleven parameters. This is a time consuming algorithm that is heavily dependent on the number of points to be fitted. Simple sampling grids can be used to alleviate this issue, whereby every $i$th point is chosen, but this may not be sufficiently effective for reducing latency, as shown in Figure 5.5. Our proposed scheme significantly reduces the size of point clouds while maintaining their general shape and space relations. To execute this task, we use voxels which are volumetric pixels that divide the 3D space into uniform 3D cells,
Figure 5.5. The computational time required for superquadric fitting using regular sampling grids. Every $i$th data point is chosen, such as every 5 points or every 20 points and so on.

typically cubes. In each voxel, all the points present will be approximated or downsampled using their centroid. By doing this, the surface is represented more accurately. Voxels are perfectly justifiable as a means of downsampling data, having been extensively used in the medical imaging and computer graphics communities (see Figure 5.6).

By the same token, setting the dimensions of a voxel beforehand may not be the best solution to this problem. If the dimensions of the voxel are set too high relative to the point cloud, certain shape details can be lost. On the other hand, if the dimensions are too low, the benefits of downsampling may not be harnessed. To this end, our scheme performs a multi-scale voxelization of the point cloud data so that there is a good balance between speed of computation and accuracy. Figure 5.7 gives an overview of this process. For our scheme, there are four main parameters:

- $s_{\text{max}}$, which represents the maximum voxel size,
- $s_{\text{min}}$, representing the minimum voxel size,
Figure 5.6. Execution times for superquadric fitting in milliseconds for different voxel sizes. The voxel sizes range from 3.0 cm to 0.5 cm. The larger the voxel size, the lower the computational time.

- $N$, denotes the number of scales, and
- $\tau$, which represents the threshold for error change respectively.

The input to our scheme is a point cloud representing a segmented object from a 3D scene as shown in Algorithm 1. At the first scale $\sigma_1$, the cloud is voxelized using a voxel size of $s_{\text{max}}$ which significantly reduces its size. A superquadric fitting at this scale is performed. Its fitting error $e_1$ is compared to that of an initial error $e_0$ that is computed using unoptimized superquadric parameters acquired via the eigenvalue decomposition of the original point cloud as mentioned in Section 5.1.1. If the difference between these error values is less than $\tau$, the process stops and the superquadric parameters recovered at this scale are accepted.
Likewise, if this difference is greater than $\tau$, we proceed down to the next scale initializing the fit using the acquired superquadric parameters of the previous scale. At this level, the
**Algorithm 1: Multi-scale Superquadric Fitting Algorithm**

**Input**: ObjectPointCloud, \( N, s_{\text{max}}, s_{\text{min}}, \tau \)

**Output**: \( \Lambda \)

**begin**

/* Initialization */
\[
\delta \leftarrow \frac{s_{\text{max}} - s_{\text{min}}}{N};
\]
\[
e_0 \leftarrow 0;
\]
\[
\Delta_e \leftarrow 0;
\]
\[
i \leftarrow 1;
\]
preProcessPointCloud(ObjectPointCloud);

/* Process */

while \( i \leq N \) and \( \Delta_e \geq \tau \) do

\[
s_i \leftarrow s_{\text{max}} - [(i - 1) * \delta];
\]

DownSampledCloud \( \leftarrow \) downSampleCloud(ObjectPointCloud, \( s_i \));

estimateInitialSuperquadricParameters(ObjectPointCloud);

\( \hat{\Lambda} \leftarrow \) recoverSuperquadricParameters(DownSampledCloud);

\( e_s \leftarrow \) calculateFittingError(ObjectPointCloud, \( \hat{\Lambda} \));

\[
\Delta_e = \text{abs}(e_0 - e_s);
\]

\[
e_0 \leftarrow e_s;
\]

\[
\Lambda \leftarrow \hat{\Lambda};
\]

\[
i = i + 1;
\]

**end**

Voxel size \( s_i \) to be used is determined according to Equation 5.9.

\[
s_i = s_{\text{max}} - [(i - 1) * \delta],
\]

\[
\delta = \frac{s_{\text{max}} - s_{\text{min}}}{N},
\]

\[
i = 1, ..., N
\]

The same process continues until there is no significant change in the error, whereby the process stops or it would proceed until the \( \sigma_N \) scale is encountered.

\[
\sum_{k=0}^{n} ((F^{e_1}(x_k; \Lambda) - 1))^2
\]

(5.10)
The error metric we use is given by Equation 5.10 which is similar to Equation 5.8 without the constraint for the smallest superquadric. If the parameters of this scheme are set appropriately, this may never be necessary. However, including a minimum downsampling stage ensures that some degree of data downsampling is done so that latency is lowered. This scheme can be employed for much more than superquadric fitting by replacing the fitting step, the error metric, and the initialization components.
CHAPTER 6
INTENTION RECOGNITION VIA OBJECT-ACTION INTENTION NETWORKS

Figure 6.1. Human Intention Recognition Overview: The most likely object and action pair is determined.

6.1 Overview

An overview of our intention recognition framework is shown in Figure 6.2. From scene understanding, object information is used as input for construction of our object-action intention network. Based on the network, a set of queries is generated and proposed to the

54
user. A query is simply a yes-or-no question. In an ideal situation, the first query proposed to the user coincides with the user’s intention. If this does not occur, the query set is modified and another query is presented until the user’s intention is communicated as depicted by the query loop in Figure 6.2. The user’s selections are learned via the learning loop in order to adapt to the user’s preferences. Notably, the ability of our system to learn a user’s preferences over time differentiates it from the state of the art. Based on our understanding, there are no major works which simultaneously infer human intentions and learn them. With learning, we are able to improve the intention predictions which in effect reduces the need for many rounds of user interaction. The individual components of this framework are unveiled in the following sections.

### 6.2 Object-Action Intention Networks

Our aim is to determine the user’s intention which in this work is represented as object-action pairs. Hence, we infer the action the human most likely wants the robot to perform on
an object in an effort to answer the question “what to do next?” However, there are many possible answers to this question and uncertainty exists with regards to knowing which object-action pair a user desires because the user may not be able to explicitly communicate their intent, e.g. move a box versus opening a box. Therefore, we use probability theory to capture this uncertainty and perform reasoning in order to arrive at an appropriate answer. In so doing, we formulate our intention recognition problem as follows.

For each 3D scene that is captured and processed, there are $n$ objects present. These objects are represented by the binary random variables $O = \{O_1, \ldots, O_n\}$ where their values $o_i$ indicate whether the user wants to manipulate the object or not i.e. $o_i = ‘Yes, I want to manipulate this object’$ or $o_i = ‘No, I do not want to manipulate this object’$. Associated with each object are binary action variables $A = \{A_1, \ldots, A_m\}$ whose values $a_j$ indicate whether the user wants to perform the action on the object or not i.e. $a_j = ‘Yes, I want to perform this action’$ or $a_i = ‘No, I do not want to perform this action’$. In addition to these object and action variables are object feature variables $F = \{F_1, \ldots, F_c\}$, which are also binary random variables representing some intrinsic property of the object. Features can range from object distance from the camera to object color. There can be $c$ feature variables per object and they can potentially bias an object for selection by the user e.g. user’s preference for red objects.

Under this formulation, our task is to infer the most probable object that the user wants to manipulate as well as the most probable action that the user intends to perform on the object. Thus, our goal is to find the highest-probability joint assignment of object and action variables of the form $P(o_i = yes, a_j = yes)$, which represents the intent of the user. With a joint distribution of these variables in our model, we can answer questions about the observed scene. These questions can vary from the standard conditional probability query $P(O = o, A = a | F = f)$, where we want to determine the probability distribution over the values $o, a$ of the sets of random variables $O$ and $A$ given an instantiation of $f$ to
to finding the most probable assignment to some subset of variables. We are particularly interested in determining the maximum a posteriori (MAP) probability, whereby the task is to infer the most likely assignment to the variables in $O$ and $A$ given the evidence $F = f$: 

$$\arg \max_{o,a} P(o, a | f).$$

In order to represent the joint distribution in an efficient manner and perform fast inference, we employ Markov Networks, which are probabilistic graphical models that encode relationships between random variables.

### 6.2.1 Markov Networks

Markov networks are undirected graphical models that efficiently capture a joint distribution $P$ over a set of random variables by exploiting existing independence properties that exist between them. The nodes in the graph represent the random variables, and the edges represent some from of direct probabilistic interaction between neighboring variables [85]. They are highly applicable as the foundation of our intention recognition system because they can model the relationships between the object and action variables where directional influence between them cannot be naturally ascribed.

In order to quantitatively represent the joint distribution, the structure of the graphical network must be associated with a set of parameters. This parameterization is achieved by factors $\Phi$, which are functions that map the values of a set of random variables $d$ to positive real numbers $\mathbb{R}^+$. Factors capture the compatibilities between related variables and are defined on the cliques of the graph (a collection of nodes that are all pairwise neighbors). A small example is depicted in Figure 6.3 where factors are defined over the cliques of the network. For instance, for each possible value of the factor defined over the action ‘pick up’ and the object ‘bottle,’ there is a positive real number. The joint distribution is then defined by taking the product of these local factors, then normalizing the product to obtain a legal distribution as shown in Equations 6.1 to 6.3.
Figure 6.3. Example of a small Markov Network with its defined local factors. Factors are defined over the cliques of the graph and maps the values that a set of random variables can take to positive real numbers.

\[
P(o_1, \ldots, o_n, a_1, \ldots, a_m, f_1, \ldots, f_{cn}) = \frac{1}{Z} \Phi(o_1, \ldots, o_n, a_1, \ldots, a_m, f_1, \ldots, f_{cn}),
\]

where
\[
\Phi(o_1, \ldots, o_n, a_1, \ldots, a_m, f_1, \ldots, f_{cn}) = \phi_1(d_1) \ldots \phi_n(d_n),
\]

and
\[
Z = \sum_{o_1, \ldots, o_n, a_1, \ldots, a_m, f_1, \ldots, f_{cn}} \Phi(o_1, \ldots, o_n, a_1, \ldots, a_m, f_1, \ldots, f_{cn})
\]

\(Z\) is known as the partition function and the elements of \(d\) represents subsets of variables. Also, \(\phi(d)\) can be any of the following two forms: \(\phi(o_i, a_j)\) or \(\phi(o_i, f_{ci})\).

Using the joint distribution represented by the Markov network, we can acquire the MAP as shown in Equation 6.4. We believe that by using this formulation, the MAP is likely to converge to a user’s intent over time.

\[
MAP(O, A | f) = \arg \max_{o, a} P(o, a, f)
\]

To handle this inference problem in an efficient manner, approximate and optimized inference methods are used. In this work, we adopt the Loopy Belief Propagation method [85] which
uses message passing to approximate marginal probability distributions for variables and factors.

### 6.3 Network Construction

In this work, the structure of these Markov networks depends on the captured 3D scene, particularly which objects are present in the scene, the actions that can be performed using them, and contributing object properties such as the relative distances of objects from the camera, color etc. In this work, we refer to these networks as *object-action intention networks* because they change according to the scene under consideration. In so doing, they are dynamically configured to calculate the likelihood of human intentions.

The sets of object and action variables comprise the corresponding set of object and action nodes in the network as depicted in Figure 6.4. The edge link between an object and action node signifies that the action can be performed on the object and that a direct probabilistic relationship exists between them. For example, one can perform the ‘open’ action on a ‘bottle’ object. Moreover, for each object, there is a predetermined finite set of actions that can be attributed to it. For each object-action pair, its corresponding network factor $\phi_i$ receives its values from stored factors which we refer to as *template factors*. These template factors are used for learning the user’s intentions over time.

Additionally, the set of feature variables correspond to the feature nodes in the network, thereby representing some intrinsic property of the objects themselves (see Figure 6.4). For every object, there is a set of $c$ features which is represented in Figure 6.4 as a vector. In this work, we only utilize an object’s distance from the camera as a contributing object feature (i.e. $c = 1$) for reasoning and these variables indicate whether an object is near or far from the camera based on a dynamically-determined distance threshold. These object, action, and feature variables are related and their symmetrical dependencies are captured by the network with the overall joint probability distribution $P(o_1, \ldots, o_n, a_1, \ldots, a_m, f_1, \ldots, f_{cn})$. Objects
in the scene are resolved via object categorization and the object distances from the camera are calculated via object pose estimation as described in Sections 4 and 5 respectively. An example of an object-action intention network is shown in Figure 6.5 (b) for the scene shown in Figure 6.5 (a).

6.4 Query Selection

In an effort to acquire a solution to Equation 6.4 that coincides with the user’s intent, a human-robot interaction must take place. The user is prompted with a series of queries based on the marginal probabilities of variables and factors in the network. A query in this case is simply a yes-or-no question involving an object variable, an action variable, or a combination of both. For every network \( G \) that is constructed, there is a set of \( s \) generated queries \( Q = \{Q_1, ..., Q_s\} \) sorted according to their probabilities. As mentioned previously, our aim is to minimize the amount of human interaction in our system. This in turn translates into getting the first query \( Q_1 \) of \( Q \) to match the intent of the human. For this
to be achieved, the user’s preferences must be learned and this is described in a subsequent section. Moreover, an individual query $Q_t$ may represent an attempt to determine if the user wants to manipulate an object $Q_t = \{O_i\}$, perform an action $Q_t = \{A_j\}$, or perform an action on a specific object $Q_t = \{O_i, A_j\}$ as follows:

1. *Do you want to manipulate $O_i$?*
2. Do you want to perform $A_j$?

3. Do you want to perform $A_j$ on $O_i$?

The user’s response, regardless of whether it is positive or negative, leads to a modification of the set $Q$, eventually resulting in a set that only contains the query which corresponds to the user’s intention. For instance, if a query of Type 1 is proposed which asks if the user wants to manipulate an object $O_i$ and the user responds negatively, all queries involving object $O_i$ are removed from $Q$. On the other hand, if the user responds positively, only queries involving object $O_i$ are kept in $Q$. Feedback from the user is treated as observations or evidence in the network and the network is updated accordingly. This process is depicted in Figure 6.2.

6.5 Mapping of Queries to Different Interfaces

The yes-or-no query framework outlined in this work can be generalized to various types of human-robot interfaces. We can consider interfaces to be different from one another in terms of the communication bandwidth. For instance, a touch screen interface has higher bandwidth for communication than a brain computer interface (BCI).

For a touch screen that is being used by a fully-functional individual, we can display a screen image highlighting the most probable objects the user may want to use and prompt them to indicate the desired one. This is equivalent to asking many yes/no types of queries simultaneously, i.e. asking ‘Do you want to use the bottle?’, ‘Do you want to use the cup?’, ‘Do you want to drink from the bottle?’, ‘Would you like to pour from something?’ etc. For instance, if the user clicks on the image of a highlighted bowl, they are indicating that they want to manipulate just the bowl and not any of the other scene objects ($bowl = yes$, $cup = no$, $bottle = no$). Note that this is different from what we can assert for a yes/no answer about a single object. An answer of no to the query ‘Do you want to use the cup?’ simply means $cup = no$. It does not allow us to assert anything about the other objects in the
scene, such as the bowl or the bottle. Therefore, touch screen interfaces permit responding to multiple or complex queries in one instant. These responses are used as input for modifying the query set. Consequently, we expect fewer interactions for complex queries rather than simple ones.

On the other hand, for a Brain Computer Interface used by a person with reduced capabilities, we can display a screen image divided into sectors to represent different object or action queries such as ‘Do you want to use the cup?’, ‘Do you want to drink something?’, ‘Do you want to pick up something?’ etc. The person’s measured brain response can then be used to determine whether \((\text{cup} = \text{no})\), \((\text{drink} = \text{yes})\), or \((\text{pick up} = \text{no})\). It is important to note that it may only be possible to answer simple queries in such cases. The responses to these queries would be used to modify the query set. As a result, more interactions may be necessary for determining complete intentions. However, with our framework, the aim is to reduce these interactions using intention recognition.

6.6 Learning

In this work, the ideal scenario for intention recognition involves having a query set \(Q\) where the first query proposed to the user is actually the user’s intent. For this to occur, we learn the past decision history of the user in an attempt to adapt to the user’s preferences. This in turn translates into modifying the probability distribution over all object-action pairs so that the user’s preferences are captured. To this end, we cast our learning framework as a form of Recursive Bayesian Incremental Learning [86] which is described as follows.

Let \(\theta\) represent the collection of multinomial parameter vectors for the factors defined over all object and action variables, hence \(\theta_i = \phi_i\). Furthermore, let \(D^k = \{x_1, \ldots, x_k\}\) explicitly represent \(k\) observed user choices where \(x_i = \{\text{No}, \text{No}, \ldots, o_i = \text{Yes}, \ldots, \text{No}, a_j = \text{Yes}, \ldots, \text{No}\}\) indicating a user’s selection of the \(i^{th}\) object and the \(j^{th}\) action as captured via our object action intention network. Our aim is to determine the most likely object-action
pair based on information we have already acquired. Therefore, by using Bayes formula, the posterior probability for the distribution over all object and action variables satisfies the recursive relation given in Equation 6.5.

\[
P(\theta | D^k) \propto P(x^k | \theta)P(\theta | D^{k-1})
\]  

(6.5)

Equation 6.5 allows us to incrementally learn a user’s preferences as they repeatedly interact with our system and data are collected. Given that \( P(\theta | D^0) = P(\theta) \), we can use this equation repeatedly to produce the sequence of probabilities \( P(\theta), P(\theta | x_1), P(\theta | x_1, x_2), \) and so on.

The term \( P(x^k | \theta) \) in Eq. 6.5 is known as the likelihood function and it represents the probability of the observed data given the parameters values \( \theta \). Its value is given in Equation 6.6 whereby the probability of each datum \( x_i \) given the parameters \( \theta \) is proportional to the product of the factors \( \phi_i \) defined over the subset of random variable values \( d_i \) (see Eq. 6.2).

\[
P(x^k | \theta) \propto \prod_{i=1}^{k} \phi_i(d_i | \theta)
\]

(6.6)

\[
\propto \prod_{i=1}^{k} \prod_{j=1}^{s} \theta_{ij}^{N_{ij}}
\]

(6.7)

Each object-action factor \( \phi_i \) has \( s \) values (in our case 4), therefore \( \theta_{ij} \) represents the \( j^{th} \) value of the \( i^{th} \) factor as shown in Eq 6.7. \( N_{ij} \) represents the count on the respective factor value which can be 1 or 0 in our case.

Additionally, the term \( P(\theta | D^{k-1}) \) in Equation 6.5 represents the prior probability distribution based on the set of data samples that were previously observed. The prior is updated as data are collected thereby producing the posterior distribution which then serves as the prior for the subsequent observation. We assume that the distributions under consideration in this work are of the Dirichlet form [85]. Dirichlet distributions are probability distribu-
tions for multivariate random variables parameterized by a vector of hyperparameters $\alpha$, where $\alpha_j > 0$ determines the shape of the distribution and corresponds to counts on data values observed. This translates into each factor $\phi_i$ defined over a subset of variable values $d_i$ being Dirichlet. Equation 6.8 defines a Dirichlet distribution over the parameter vector $\theta$.

$$Dir(\theta; \alpha) = \frac{1}{C(\alpha)} \prod_{i=1}^{k} \prod_{j=1}^{s} \theta_{ij}^{\alpha_{ij}-1}$$

(6.8)

$C(\alpha)$ is a normalizing constant. By using a Dirichlet prior, the posterior distribution is also Dirichlet which allows us to update the distributions using sufficient statistics from the data. Sufficient statistics are functions of the data samples $D$ that contain all of the information relevant to estimating the parameter $\theta$ [86]. We can then combine Equations 6.5, 6.7, and 6.8 to produce Eq. 6.9.

$$P(\theta | D^k) \propto \prod_{i=1}^{k} \prod_{j=1}^{s} \theta_{ij}^{N_{ij}} \prod_{i=1}^{k} \prod_{j=1}^{s} \theta_{ij}^{\alpha_{ij}-1}$$

$$\propto \prod_{i=1}^{k} \prod_{j=1}^{s} \theta_{ij}^{N_{ij}+\alpha_{ij}-1}$$

(6.9)

What follows is that the maximum a posterior estimate for $\theta$ is given according to sufficient statistics as shown in Eq. 6.10.

$$\hat{\theta}_{ij} = \frac{N_{ij} + \alpha_{ij} - 1}{\sum_{j=1}^{s} N_{ij} + \sum_{j=1}^{s} (\alpha_{ij} - 1)}$$

(6.10)

The Dirichlet hyperparameter $\alpha_{ij}$ stores the prior count observed for the value $j$ of factor $\phi_i$ whereas $N_{ij}$ represents its current count. Thus, the parameters $\theta$ are updated as more information becomes available. As a result, when the user selects an object to manipulate and an action to perform on that object, the value of the corresponding parameter value $\theta_{ij}$ is updated.
If all \( \alpha_{ij} \)'s in Eq. 6.10 are set to the same value \( \alpha \) and the numerator and denominator are divided by \((\alpha - 1)\), we are left with the equivalent form shown in Eq. 6.11.

\[
\hat{\theta}_{ij} = \frac{\lambda N_{ij} + 1}{\sum_{j=1}^{s} \lambda N_{ij} + s} \tag{6.11}
\]

\[
\lambda = \frac{1}{\alpha - 1} \tag{6.12}
\]

This is our learning equation where \( \lambda \) determines how quickly we learn intentions\(^1\). In all of the experiments in this paper, \( \lambda \) is set to 1.

Subsequently, the values of every possible object-action factor \( \phi_i \) is stored as a means of tuning the factor parameters to the user’s preferences over time. To accomplish this, we use stored template factors \( \Phi_T \) for each permutation of an object and its associated actions. Hence, for each factor \( \phi_i \) occurring in an object-action intention network, its corresponding template factor \( \phi_i^T \) is updated when a user completes their selection according to Eq. 6.11.

\(^{1}\alpha > 1\)
CHAPTER 7
RESULTS

In this chapter\(^1\), we present experiments to demonstrate the efficacy of the scene-dependent human-robot collaborative system outlined in this dissertation. These experiments were performed on RGB-D scenes of common household objects captured by a Microsoft Kinect sensor [44]. All of the objects are located on tabletops. The algorithms used in this work were all implemented using C++ in conjunction with the Point Cloud Library[87] for point cloud processing and libDAI[88] for discrete approximate probabilistic inference. The evaluations were performed on a PC equipped with a 2.13GHz Intel Core Duo processor and 4.00GB of memory. We first outline the main dataset used in this work followed by results for segmentation, categorization, and pose estimation. We then describe the experimental set up used for evaluating the intention recognition framework and present the corresponding results.

7.1 Dataset

For object categorization purposes, we use our own RGB-D dataset of common household objects which consists of 110 objects divided into 11 object categories with 10 object instances per category. The dataset contains many objects that are similar in shape and appearance. For each object instance, 6 segmented RGB-D point clouds from different camera viewpoints are collected totaling 660 point clouds. It is important to note that these point clouds are meant to replicate real-world conditions where there is no post-processing to deal with imperfect object segmentation and holes in the data where the Kinect is incapable of

\(^1\)Portions of the results shown here have been published (Duncan et al. [9]) and are utilized with the permission of the publisher as shown in Appendix A.
determining depth due to transparent or reflective objects. Some examples are shown in Figure 7.1.

7.2 Segmentation Evaluation

In this section, we present a few visual results depicting the performance of our segmentation procedure. Figure 7.2 (a) and (b) depicts two typical tabletop scenes with common household objects. For both scenes, every object present is properly extracted from the RGB-D point clouds provided by the Kinect [44]. The green regions signify the extracted points belonging to candidate objects. Segmentation is never perfect and errors do occur. As with many vision problems, clutter and occlusion are inhibiting factors. Figure 7.3 provides results to demonstrate the performance of our procedure under these conditions. As can be seen from the figure, even though some of the objects present are properly extracted, there are instances where two separate objects are extracted as one and others are not extracted because they are not fully visible. Despite these limitations, the results we obtain are sufficient for our current tasks. For future work, we intend to address these issues in order to create a more robust system.
Figure 7.2. Segmentation results for typical scenes. Original images are shown in (a) and (b). The extracted object points are highlighted in green in (c) and (d).

7.3 Categorization Evaluation

In this section, we briefly present a few results to demonstrate the performance of our categorization procedure using objects from the dataset described in Section 7.1. Figure 7.4 (a) depicts a scene comprised of object instances from 10 object categories (bottle, bowl, box, can, carton, mug, spray-can, tin, tube, tub) located on a tabletop. Our procedure is able to accurately determine the correct category for each object present. This is depicted by the confusion matrix in Fig. 7.4 (b), which is a visual representation of the performance of our procedure where each column of the matrix represents the instances of the predicted category, while each row represents the instances of the actual category. High values along the diagonal of this matrix indicate good performance.

Figure 7.5 (a) depicts a scene with the same object composition where the procedure is not as accurate due to variations in object placement, poor point cloud quality etc. Here, a
Figure 7.3. Segmentation results for scenes with increased clutter and occlusion. Original images are shown in (a) and (b). The extracted object points are highlighted in green in (c) and (d).

Figure 7.4. Categorization results for a typical scene. Visual results are presented on the left while a confusion matrix showing the exact results is provided on the right.

box is incorrectly classified as a mug and a spray-can is incorrectly classified as a can due to inherent similarities in appearance. We have observed that within our dataset, shape is the strongest cue. This is expected as the majority of the categories exhibit the greatest similarity
according to shape (e.g., rectangular box vs. cylindrical spray-can) whereas appearance differences are responsible for the intra-class variations. We present a confusion matrix in Figure 7.6 showing results obtained for the 110 objects in our dataset. Our procedure performs well and this establishes the efficacy of our approach for recognizing household object categories.

### 7.4 Pose Estimation Evaluation

In this section, we present experiments to demonstrate the efficacy of our proposed multi-scale voxelization scheme for fitting superquadrics. It is important to note that the 3D point clouds used are captured from a single view and that segmentation is not perfect. This makes the shape and pose recovery of an object difficult because of the lack of information. Our algorithm recovers the superquadric that best fits the data that it is given, and for the majority of cases it performs well despite noisy or spurious data. Nonetheless, we first outline the datasets used in this work, followed by our evaluations.
7.4.1 Evaluation Datasets

We use two datasets to perform our evaluations; our own dataset of common household objects described in Section 7.1 in different poses which we refer to as Dataset 1 and Lai’s RGBD Dataset [89, 90] which consists of RGB and depth images of 300 common everyday objects taken from multiple view angles organized into 51 categories which we refer to as Dataset 2. These objects are primarily cylindrical, spherical, or box-like in shape thereby rendering the superquadric as the ideal parametric model for recovering their shapes because of its tri-axis symmetry characteristic. We provide the principal axis orientation of the objects in our dataset (Dataset 1) as the ground truth pose information. We believe that
this is an appropriate metric for determining pose when objects are to be handled. For spherical objects where there may be more than one principal axis, we choose the one that is closer to the world z-axis. Examples of images for objects used from both datasets are shown in Figure 7.7.

![Examples of objects from our dataset (top row) and the RGBD Dataset [89] (bottom row).](image)

**Figure 7.7.** Examples of objects from our dataset (top row) and the RGBD Dataset [89] (bottom row).

### 7.4.2 Effect of Changing Voxel Sizes

In this section, we present quantitative results on *Dataset 1* demonstrating the effect of changing the voxel size for point cloud downsampling on pose estimation accuracy. We measure this accuracy by calculating the Euclidean distance between the ground truth location of the centroid of an object and the centroid position values recovered by the superquadric (i.e. \( p_x, p_y, p_z \)). We also investigate the accuracy with regards to principal axis estimation. This is measured by calculating the angle between the ground truth principal axis vector and the recovered one. Table 7.1 shows the results of this experiment. The first row shows the average distance between the recovered object centroid position and ground truth, the second row shows the median distance, the third row shows the average angle differences between the recovered principal axes and ground truth, and the fourth shows the median.
Table 7.1. The effect of voxel size on the recovered pose estimates.

<table>
<thead>
<tr>
<th></th>
<th>3.0cm.</th>
<th>2.5cm.</th>
<th>2.0cm.</th>
<th>1.5cm.</th>
<th>1.0cm.</th>
<th>0.5cm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average location distance</td>
<td>3.35cm.</td>
<td>3.36cm.</td>
<td>3.33cm.</td>
<td>3.36cm.</td>
<td>3.31cm.</td>
<td>3.29cm.</td>
</tr>
<tr>
<td>Median location distance</td>
<td>2.2cm.</td>
<td>2.52cm.</td>
<td>2.7cm.</td>
<td>2.5cm.</td>
<td>2.6cm.</td>
<td>2.24</td>
</tr>
<tr>
<td>Average angle difference</td>
<td>31.2°</td>
<td>22.9°</td>
<td>23.1°</td>
<td>23.7°</td>
<td>22.5°</td>
<td>27.2°</td>
</tr>
<tr>
<td>Median angle difference</td>
<td>2.1°</td>
<td>3.02°</td>
<td>3.02°</td>
<td>3.80°</td>
<td>3.1°</td>
<td>3.6°</td>
</tr>
</tbody>
</table>

Notice that the average location distance is relatively the same across the different scales, which demonstrates that for these voxel sizes, pose estimation is not severely affected. This also tells us that for this dataset, it may not be necessary to use all of the point cloud information to determine the approximate location of an object. Conversely, determining the exact principal axis orientation may require using more data in this case. We report the median angle difference due to the severe effect that a miscalculated orientation value can have on the overall average error. One advantage of our multi-scale voxelization approach is that it can implicitly determine the appropriate downsampling scale for 3D data, hence reducing the adverse effects of having one predetermined value.

7.4.3 Pose Recovery Results

In this section, we present the pose recovery results of our proposed approach in comparison to two other methods: the classical superquadric fitting approach introduced by Solina et al. [76], and the approach of Biegelbauer et al. [42]. In the classical algorithm, the full point cloud is used for recovering the parameters to the superquadric model. Therefore, the computational time of this method is directly proportional to the amount of object points provided. Conversely, Biegelbauer and Vincze’s method uses a hierarchical RANSAC-based search and a sorted quality-of-fit criteria to find superquadric models for objects in a scene[42].
7.4.3.1 Experiments Using Dataset 1

Table 7.2 displays the results of pose estimation on Dataset 1 using all three algorithms. We include the median pose accuracies as well as the median absolute deviation because the distribution across objects is irregular. For instance, if the estimated principal axis is incorrect, the angle difference from the ground truth can in some cases be approximately 90° producing massive error. Also, we use the median absolute deviation because it is a robust statistic that is resilient to data irregularities. Nonetheless, it can be seen that our algorithm is most similar in performance to the classical approach even though the computational time required by our algorithm is significantly lower as a result of the multi-scale downsampling (see Figure 7.8). Beigelbauer and Vincze’s approach [42] produced less accurate but relatively similar results, but at great computational expense, which may be unacceptable for a responsive robotics system.

<table>
<thead>
<tr>
<th></th>
<th>Proposed</th>
<th>Solina</th>
<th>Biegelbauer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median location distance</td>
<td>2.23cm.</td>
<td>2.24cm.</td>
<td>3.85cm.</td>
</tr>
<tr>
<td>Median Absolute Deviation (distance)</td>
<td>0.89</td>
<td>0.86</td>
<td>0.461</td>
</tr>
<tr>
<td>Median angle difference</td>
<td>2.85°</td>
<td>3.23°</td>
<td>13.46°</td>
</tr>
<tr>
<td>Median Absolute Deviation (angles)</td>
<td>1.77°</td>
<td>2.52°</td>
<td>10.52°</td>
</tr>
<tr>
<td>Average time (s)</td>
<td>0.04</td>
<td>2.1</td>
<td>7.9</td>
</tr>
</tbody>
</table>

7.4.3.2 Experiments Using Dataset 2

Due to the similarity in performance of our approach and the one of Solina et al. [76], we perform a more detailed analysis of the pose estimation performance of both algorithms. We conducted pose estimation experiments on a subset of Dataset 2 consisting of over 3500 total image views of 5 different objects. We show the results in Table 7.3. We proceed with this experiment from the standpoint that for different views of the same object, the principal axis orientation should relatively be the same. The RGB-D dataset used in this test
Figure 7.8. Average computational times for the three algorithms tested in this work on Dataset 1 - Proposed approach: 40ms, Solina et al.: 2.02s, Biegelbauer et al.: 7.88s.

Table 7.3. Pose estimation results of our proposed algorithm and the classic superquadric fitting algorithm [76] on a subset of Dataset 2.

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ball</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.29</td>
<td>0.30</td>
<td>0.30</td>
<td>0.09</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.33</td>
<td>0.34</td>
<td>0.33</td>
<td>0.23</td>
</tr>
<tr>
<td>Solina et al.</td>
<td>0.29</td>
<td>0.31</td>
<td>0.33</td>
<td>0.09</td>
</tr>
<tr>
<td>2</td>
<td>0.26</td>
<td>0.32</td>
<td>0.32</td>
<td>0.26</td>
</tr>
<tr>
<td>Cereal box</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.33</td>
<td>0.31</td>
<td>0.32</td>
<td>1.1</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.26</td>
<td>0.25</td>
<td>0.29</td>
<td>4.1</td>
</tr>
<tr>
<td>Solina et al.</td>
<td>0.26</td>
<td>0.25</td>
<td>0.29</td>
<td>4.1</td>
</tr>
<tr>
<td><strong>Food</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>can</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.30</td>
<td>0.28</td>
<td>0.28</td>
<td>0.18</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.26</td>
<td>0.29</td>
<td>0.29</td>
<td>0.52</td>
</tr>
<tr>
<td>Solina et al.</td>
<td>0.26</td>
<td>0.29</td>
<td>0.29</td>
<td>0.52</td>
</tr>
<tr>
<td><strong>Soda</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>can</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.29</td>
<td>0.30</td>
<td>0.31</td>
<td>0.18</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.21</td>
<td>0.09</td>
<td>0.22</td>
<td>0.61</td>
</tr>
<tr>
<td>Solina et al.</td>
<td>0.21</td>
<td>0.09</td>
<td>0.22</td>
<td>0.61</td>
</tr>
</tbody>
</table>

contains multiple views of the same object as it rotates on a turntable, hence only changing the orientation of the object around the upright axis. Therefore, an appropriate test for pose accuracy in this case is to calculate both the standard deviation $\sigma$ and median absolute deviation of the principal axis orientation of multiple views of the same object. Ideally, these
values should be low, indicating that the pose estimations are less dispersed and mostly like
the mean. Our subset consists of the varying views of a ball, cereal box, soda can, and food
can. These results confirm that we are not sacrificing accuracy for computational savings.
Rather, we obtain comparable accuracy to the state of the art and achieve it in considerably
less time. We achieved less speedup with this dataset as compared to Dataset 1 because
the object point clouds were smaller. With larger point clouds, the computational savings
become more noticeable and vice versa.

7.4.4 Shape Fitting Estimation

In this section, we report the effect of multi-scale downsampling on the quality of the
recovered superquadric model. Table 7.4 demonstrates the shape fitting quality of our ap-

<table>
<thead>
<tr>
<th></th>
<th>Error</th>
<th>No. of Points per Cloud</th>
<th>No of Points Processed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our approach</td>
<td>15.22</td>
<td>15641</td>
<td>110</td>
</tr>
<tr>
<td>Solina et al.</td>
<td>14.1</td>
<td>15641</td>
<td>15641</td>
</tr>
</tbody>
</table>

approach in comparison to that of Solina et al. [76]. These values were calculated for objects
of Dataset 1 according to Equation 5.10. They are given in terms of the average fitting
error, the average number points per object point cloud, and the average number of points
processed per cloud. As can be seen, with downsampling the fitting error is remarkably
similar to the value acquired by using the complete object point cloud. Using this dataset,
we discovered that only approximately 1% of the point cloud is necessary for determining
the relative pose of an object. Visual examples of our shape fitting results are shown in
Figure 7.9.

7.4.5 Grasping Application

In this section we present quantitative results to gauge the effectiveness of our pose esti-
mation approach for manipulating a robotic arm. To execute this task, we use a wheelchair-
Figure 7.9. Visual results of the superquadric fitting. Left column: RGB Image, Middle column: Noisy segmented point cloud, Right Column: Recovered superquadric

mounted 9 degree of freedom robotic arm system known as the USF WMRA, which is shown in Figure 7.10. This system is designed to assist the physically challenged in manipulating objects in their living environments without the assistance of other human beings. The grasping tests were performed by placing a single object on a table within the arm’s workspace.
Figure 7.10. The Wheelchair-Mounted Robotic Arm (WMRA) system that is used to assist physically-challenged individuals.

For each grasping attempt, the object is placed in a new position. As shown in Table 7.5, our results demonstrate an overall grasping success rate of 92%, thus confirming that our approach for pose estimation is accurate enough to enable robotic manipulation of relevant objects.

Table 7.5. Grasping test using a Wheelchair-Mounted Robotic Arm (WMRA)

<table>
<thead>
<tr>
<th></th>
<th>Food Can</th>
<th>Tea Box</th>
<th>Juice Box</th>
<th>Spray Can</th>
<th>Air Freshener</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attempts</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>Successful grasps</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>23</td>
</tr>
</tbody>
</table>

7.5 Intention Recognition Evaluation

This section details the results acquired to validate the effectiveness of our approach at reducing the number of interactions between a human and robot using intention recognition.
It should be noted that for all of the following experiments, the object-action pairs of the constructed networks are initialized with equivalent probabilities. We first outline our evaluation set up, then describe the baseline method we use for comparisons followed the results we obtained for different scene changes.

### 7.5.1 Evaluation Set Up

Our evaluation scenarios are based on the individual steps that constitute activities of daily living (see Figure 1.2) and consist of objects and action pairs that coincide with a user’s intention. To determine each object-action pair, human-robot interaction must take place and we refer to these rounds of interactions as *sessions*. This means that at the end of each session, a human intention is recognized, namely the object the human wants to manipulate and the action they want to perform on the object. Table 7.6 lists the 11 possible object categories, 7 possible actions, and 36 possible object-action pairs. Table 7.7 further lists the various groupings of object-action pairs that were tested in this work. We assume that

<table>
<thead>
<tr>
<th><strong>Objects</strong> (11)</th>
<th>Bottle, Bowl, Box, Can, Carton, Cup, Mug, Spray-can, Tin, Tube, Tub</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actions</strong> (7)</td>
<td>Drink, Grasp, Move, Open, Pour, Push, Squeeze</td>
</tr>
</tbody>
</table>
these groups correspond to steps belonging to basic activities of daily living such as eating breakfast or drinking a soda.

Table 7.7. Groups of object-action pairs evaluated in this work.

<table>
<thead>
<tr>
<th>Group</th>
<th>Object-Action Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Grasp-Box, Open-Box, Grasp-Carton, Open-Carton, Pour-from-Carton</td>
</tr>
<tr>
<td>2</td>
<td>Grasp-Can, Move-Can, Pour-from-Can, Drink-from-Cup</td>
</tr>
<tr>
<td>3</td>
<td>Grasp-Carton, Move-Carton, Pour-from-Carton, Drink-from-Cup</td>
</tr>
<tr>
<td>4</td>
<td>Grasp-Bottle, Move-Bottle, Pour-from-Bottle, Drink-from-Cup</td>
</tr>
</tbody>
</table>

The main performance metric we use for our evaluations is the number of human-robot interactions per session. To understand the significance of this metric, the reader is referred to the Query Loop of Figure 6.2. The ideal scenario occurs when this value is 1, which indicates that the first query proposed to the user is their actual intent. We conduct experiments to ascertain this value over multiple sessions and first present the performance of our scene-dependent intention recognition framework under the following conditions:

- Constant scene between sessions
- Same objects, different object positions between sessions
- Simultaneous object and position changes between sessions

This leads into our main result exposing how well our framework responds after learning a group of intentions that coincide with a user’s preferences.

7.5.2 Naïve Baseline Intention Recognition Approach

To the best of our knowledge, no algorithm exists in the state of the art whereby we can perform a direct comparison. The simplest baseline approach in our case would be one that selects an object-action pair from the set of all possible configurations of objects and actions, but this is impractical. Thus, we compare the results of our framework with a naïve
Table 7.8. Examples of scenes used for evaluation. Within a row, the scenes either have no difference (row 1), differ by object positions (row 2), or a combination of different objects and varying positions (row 3).

<table>
<thead>
<tr>
<th>Scene Difference</th>
<th>Session scene examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td><img src="image1" alt="Session 1" /> ⇒ <img src="image2" alt="Session 2" /> ⇒ <img src="image3" alt="Session i" /></td>
</tr>
<tr>
<td>Object Positions</td>
<td><img src="image1" alt="Session 1" /> ⇒ <img src="image2" alt="Session 2" /> ⇒ <img src="image3" alt="Session i" /></td>
</tr>
<tr>
<td>Objects &amp; Positions</td>
<td><img src="image1" alt="Session 1" /> ⇒ <img src="image2" alt="Session 2" /> ⇒ <img src="image3" alt="Session i" /></td>
</tr>
</tbody>
</table>

Baseline approach hinged on object and action probabilities outlined as follows. For every object in a scene, the probability of its selection by the user is expressed as the ratio of object-action pairs involving the object to the total number of object-action pairs possible in the scene. Similarly, for every action, the probability of its selection is the ratio of the object-action pairs that it is a member of to the total number of object-action pairs possible in the scene. The probabilities for each object-action pair is simply the reciprocal of the total number of object-action pairs possible in the scene. Using these probabilities, a set of queries is generated and proposed the user.

This approach captures the underlying object and action relationships in a pure manner. It is a reduced version of our approach where there is no partiality to object-action pairs due to learning or object feature influence. Furthermore, this baseline approach essentially captures the most likely object and action based only on scene content. Suppose that there is a scene composed of 4 bottles and 1 cup, this approach sets the probability that the user wants
to use a bottle higher than the probability that the user wants to use the cup. Therefore, a corresponding query suggesting that the user’s intention involves a bottle is intuitive and corroborated by the captured scene. By employing this approach for comparisons, we are able to test how reliant our framework is on the scene and the effect of learning.

7.5.3 Results for Constant Scenes

This section demonstrates how our intention recognition framework performs for the case when there are no changes to a scene, namely objects and their positions relative to the camera are unchanged. Our test scene is composed of 8 objects with set positions from the following categories: can, cup, mug, carton, tub, box, bowl, and bottle as shown in the first row of Table 7.8. Figure 7.11 displays the result of this experiment for the 4 object-action pair groups listed in Table 7.7. Each pair is chosen as the desired intention for 20 sessions and the average number of interactions required to communicate this intention is plotted along with their standard deviations as error bars (please note that when the standard deviation is 0, there are no visible error bars).

We can see from the figure that over the span of 20 sessions, the average number of interactions required by our framework monotonically decreases ending with an average value less than or equal to 2 for all groups. After 20 sessions, the average reduction in interactions is 81%. This behavior is expected as the same intention is chosen and learned for the same scene multiple times. Our framework outperforms the baseline method for which the average number of interactions per session is constant because the composition of the scene is constant.

7.5.4 Results for Object Position Changes

It is common in household environments for the placement of objects to be altered over time. Frequently-used objects are often placed in positions where they are easily accessible
Figure 7.11. Intention recognition results for a constant scene on the object-action pair groups listed in Table 7.7: objects and their positions relative to the camera are unchanged. Please note that the presence of error bars indicates a standard deviation value that is greater than 0.

or “within reach” whereas rarely-used objects are placed “out of the way.” In this section, we show how our framework performs in such cases. The scenes evaluated are composed of 8 randomly-positioned objects from the following categories: can, cup, mug, carton, tub, box, bowl, and bottle. Examples can be found in the second row of Table 7.8. For each session, each object is placed in a different position which results in a total of 340 scenes for this experiment. Figure 7.12 illustrates the result of this experiment for the object-action
pair groups listed in Table 7.7. The experimental procedure is similar to the one presented in Section 7.5.3 whereby each pair is chosen as the desired intention for 20 sessions and the average number of interactions required for recognition is calculated.

Figure 7.12. Intention recognition results for object position changes in the scene. Each session corresponds to a different scene where the object composition of the scene is the same, but their positions vary. Please note that the presence of error bars indicates a standard deviation value that is greater than 0.

The figure shows that our framework consistently reduces the number of interactions and outperforms the baseline method over the full span of 20 sessions for all groups. Additionally, the interactions were reduced by an average of 83.3% after 20 sessions. Based on our intention
recognition formulation, placing objects closer to the camera increased their likelihood for selection and vice versa. We observed that by moving desired objects away from the camera, the average number of interactions increased. However, this effect is significantly reduced after multiple sessions and is of negligible impact as a result of learning. Nonetheless, our framework consistently outperforms the baseline method for which the average number of interactions per session is the same as the previous experiment in Section 7.5.3 because it is not affected by object placement.

7.5.5 Results for Simultaneous Object and Position Changes

In this section, we demonstrate how our framework performs under considerable scene changes. As an example, consider the difference between a bathroom counter-top scene and a kitchen counter-top scene. Our test scenarios involve at most 8 randomly-selected and randomly-positioned objects, as well as 2 preselected randomly-positioned objects. These two objects may be involved in the object-action pair representing the desired intention, thus they are always present in the scene for every session. Altogether, 340 different scenes are tested and the third row of Table 7.8 presents some examples. Figure 7.13 displays the result of this experiment for the object-action pair groups listed above.

Table 7.9. Average number of interactions required for the 15\textsuperscript{th} - 20\textsuperscript{th} sessions for the simultaneous object and position changes in the scene test as shown in Figure 7.13.

<table>
<thead>
<tr>
<th></th>
<th>15\textsuperscript{th}</th>
<th>16\textsuperscript{th}</th>
<th>17\textsuperscript{th}</th>
<th>18\textsuperscript{th}</th>
<th>19\textsuperscript{th}</th>
<th>20\textsuperscript{th}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>Proposed</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>1.0</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>Baseline</td>
<td>5.4</td>
<td>5.0</td>
<td>5.8</td>
<td>5.8</td>
<td>5.2</td>
</tr>
<tr>
<td>Group 2</td>
<td>Proposed</td>
<td>2.0</td>
<td>2.25</td>
<td>2.25</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Baseline</td>
<td>5.5</td>
<td>5.5</td>
<td>7.0</td>
<td>7.0</td>
<td>6.25</td>
</tr>
<tr>
<td>Group 3</td>
<td>Proposed</td>
<td>2.5</td>
<td>2.0</td>
<td>2.25</td>
<td>1.5</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>Baseline</td>
<td>6.0</td>
<td>5.5</td>
<td>6.75</td>
<td>7.0</td>
<td>6.25</td>
</tr>
<tr>
<td>Group 4</td>
<td>Proposed</td>
<td>2.0</td>
<td>1.5</td>
<td>1.75</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Baseline</td>
<td>4.5</td>
<td>3.75</td>
<td>4.25</td>
<td>5.0</td>
<td>4.5</td>
</tr>
</tbody>
</table>
Figure 7.13. Intention recognition results for simultaneous object and position changes in the scene. Each session corresponds to a different scene where the object composition of the scene and their positions vary. The amount of interactions for the baseline fluctuates as a result. Please note that the presence of error bars indicates a standard deviation value that is greater than 0.

The figure shows that despite considerable modifications to the scene, our framework still manages to reduce the number of human-robot interactions over time, namely by 78% after 20 sessions. Only for the first session in Figures 7.13 (b) and (d) does the baseline method fare better than ours. At worst, our approach should perform similar to the baseline. Therefore, these values occur as a result of the desired objects being positioned further away from the camera, which in effect reduces their probability for selection. As mentioned in the previous
section, this effect is mitigated after multiple sessions as a result of learning. Table 7.9 lists the exact values for the proposed approach compared against the baseline method for the last 6 sessions to clearly demonstrate the reduction in the number of interactions. It is important to note that the baseline values are not constant across sessions as in previous experiments due to changes in scene composition. Also, we have determined via (Analysis of Variance) ANOVA [91] tests that the average interaction values presented for our approach along with the baseline are significantly different after at least 5 sessions, where the F-ratios are at least 8.23 with a critical value of 5.98.

7.5.6 Learning a Group of Intentions

Consider the following scenario: a person wakes up, brushes their teeth, drinks a cup of coffee, and eats a bowl of cereal before they head off to work. They repeat this sequence of events every morning. For humans, it takes relatively no effort to determine this person’s morning routine after some time. For instance, if the individual’s spouse wanted to help them get to work faster, all they have to do is put toothpaste on the toothbrush, make coffee, and prepare the cereal ahead of time because they are cognizant of their spouse’s routine.

For this reason, this section presents the results of learning a group of intentions over time then determining the amount of interaction required to choose one of them from the group. This is somewhat analogous to learning the person’s morning routine as previously described. Ideally, the selection likelihood of the intentions in the group should be higher than all other possible intentions, therefore the amount of interaction required to select one of them should be small.

The experiment is performed on scenes where the objects and their positions vary over the span of at least 50 sessions. Each intention in the group is selected at most 10 times in no particular order given a conducive scene. At the conclusion of this “training” period,
one of these intentions is randomly chosen and the average number of interactions necessary for choosing it is calculated. Figure 7.14 illustrates the results of this experiment. It shows that our framework reduces the necessary amount of interaction for all intentions tested and that it consistently outperforms the baseline. This behavior is desired because we want our framework to be able to capture a user’s preferences over time in order to simultaneously reduce human interaction and maximize robot task performance.
7.5.7 Effect of Different Learning Rates

With respect to learning, one may ask the question “what is the effect of increasing or decreasing the value of the learning rate $\lambda$ discussed in Chapter 6?” Intuitively, it is expected that increasing the learning rate should speed up learning but as shown in Figure 7.15, this may not necessarily be the case in this work. The figure demonstrates the results for the previously-discussed experiment in Section 7.5.6 with the learning rates of 0.5, 1, 10, and 1000\(^2\). Only for the ‘Pour Carton’ intention does increasing the learning rate consistently reduce the average number of interactions. Conversely, for the other intentions, the effect is the exact opposite. Interestingly, for ‘Grasp Carton’, the lowest learning rate actually fares

![Figure 7.15](image)

**Figure 7.15.** Random intention selection for different learning rates $\lambda$.

\(^2\)As noted earlier, for all the other experiments in this work, the learning rate is set to 1
better than the higher values. Overall, these results indicate that the performance of our framework is not solely dependent on the learning rate, but on other factors such as the object-action relationships induced by the scene and object positioning. High learning rates are only effective at reducing the amount of interactions for determining an intention if that intention is the only one being repeatedly chosen out of all possible intentions, but not the case for when groups of intentions are to be learned.
CHAPTER 8
CONCLUSION AND FUTURE WORK

We presented in this dissertation a novel scene-dependent human robot collaborative system capable of recognizing and learning human intentions based on scene objects, the actions that can be performed on them, and human interaction history. The system is partitioned into scene understanding and intention recognition modules in an effort to provide the robot with the ability to collaborate effectively with humans for completing everyday tasks. The system is able to segment objects from RGB-D data, determine their poses in 3D space, acquire their category labels, and determine the most likely object and action pairs desired by the user. We have demonstrated through our results in Chapter 7 how this system is capable of reducing the amount of human interactions necessary for communicating tasks to a robot. We believe that this in turn can maximize robotic task performance. We have addressed two important existing problems in this dissertation, namely, 3D pose estimation from single views, and human intention recognition for individuals with limited physical capabilities.

First, our approach for 3D pose estimation rapidly acquires the shape and pose information of unknown objects from single view point cloud data. We use a low latency multi-scale voxelization strategy that is capable of accurately estimating the shape and pose parameters of relevant objects in a scene. A reconstructed 3D model of the object is computed by fitting parametric superquadrics to the data which provides us with the underlying shape and pose, as well as volume and moments of inertia. We obtained results comparable to two 3D pose estimation algorithms and did so in significantly less time.
Second, our proposed intention recognition framework uses information acquired from the scene and learned interaction history for inferring human intentions. To the best of our knowledge, there are no existing frameworks for recognizing human intentions that only utilize scene content. The commonly-accepted approach in the literature is the observation of human physical actions, which is not appropriate for individuals with reduced physical capabilities, on whom our work is focused. At the core of this framework is our Object-Action Intention Networks, which are undirected graphical models where the nodes are comprised of object, action, and object feature variables, and the links between them indicate probabilistic interaction. With these networks and a recursive Bayesian learning paradigm, we have demonstrated how our system can adapt to a user’s preferences over time. This ability differentiates our work from the state of the art as there is a lack of major works which simultaneously infer human intentions and learn them. One major conclusion that can be drawn from this framework is that despite significant changes to the scene, in terms of objects being removed or repositioned between sessions with the system, the amount of interactions is reduced. This trend persists in all of the experiments conducted where our framework outperforms a naïve baseline intention recognition approach by achieving an average of 81% reduction in interactions after learning.

We also presented methods for object segmentation and categorization from RGB-D point clouds, which leverage recent advances in computer vision. For segmentation, we presented a method that acquires candidate objects from tabletops, and for categorization we presented a method that balances discrimination and generalization to handle intra and inter-class variations. By adopting these methods, we are able to effectively decompose scenes comprised of common household objects.

For future work, there are manifold paths that can be taken. First, we plan to extend our object-action intention networks to handle scene contexts as well as sequences of steps which coincide with complete activities. One way of capturing context is by modifying our
network formulation to include links between neighboring objects, which would facilitate
the prediction of activities where these objects are used, for example, a bowl and a spoon
used to eat cereal. To account for the temporal dynamics inherent with sequences of steps
in activities, a transition model is necessary. Therefore, we plan to adopt the architecture
of either Dynamic Bayesian Networks or Hidden Markov Models similar to the intention
recognition approaches of Tahboub [12] and Kelley [32] respectively. This would enable us
to capture an RGB-D snapshot of a scene, predict the overarching activity such as “making
oatmeal” or “drinking a soda,” as well as the individual steps of the activity in an effort to
reduce the cognitive workload and interactions with a user.

Second, we plan to extend the functionality of our system’s scene understanding compo-
nents. Our pose estimation approach can be augmented to recover multiple superquadrics
for a single object. In turn, this would facilitate pose estimation of a wider variety of shapes
present in human environments and complex objects such as objects with handles like pots,
panns, and jugs. Multiple object views can also be used for improved pose estimates. Our
categorization component can be altered to make use of incremental learning. This can be
implemented using incremental support vector machines [92] whereby additional training
data can be incorporated without re-training from scratch. This would allow the learning of
previously unseen object views and thereby improve categorization performance. Further-
more, we plan to include object affordances as another cue for categorization. The concept of
affordances focuses on how objects can be used in the world, relating their functionality. We
believe that this is an exploitable concept that is quite meaningful for robotics applications.

In terms of applications, the intention recognition framework presented herein is not
restricted to assistive devices. Thus, we believe that it could be adapted for multiple robotic
applications. For example, it can be incorporated in robots such as the USAR Whegs©
[93] for urban search and rescue tasks in order to infer the best courses of action based
on captured visual data. Such devices may have communication limitations which make it

94
difficult to transmit large data such as images of debris or a human trapped in building rubble etc. Consequently, this can lead to a reliance on yes or no questions for high-level decision making. Our query framework outlined in Chapter 6 is appropriate for such scenarios because it caters to different types of interfaces with either low or high bandwidths.
REFERENCES


APPENDICES
Appendix A Copyright Permission for Work Shown in Chapters 5 and 7

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © [year of original publication] IEEE. Reprinted, with permission, from [original publication title, and month/year of publication]
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author’s approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
3) In placing the thesis on the author’s university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity’s name goes here]’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.
ABOUT THE AUTHOR

Kester Duncan received the Bachelor of Science Degree in Computer Science from Coppin State University in Baltimore, MD U.S.A. in December 2007, where he was an Honors College scholar in the accelerated bachelors program since 2004. Following this, he received the Master of Science degree in Computer Science from the University of South Florida in Tampa, FL U.S.A. under the supervision of Dr. Sudeep Sarkar. Currently, he is a Ph.D. candidate in the Computer Science department of the University of South Florida and a member of the Center for Assistive, Rehabilitation, and Robotics Technologies at U.S.F. His research interests include saliency detection in images and videos, image retrieval, sign-language recognition, multi-modal object categorization, object pose estimation, and human intention recognition.