12-2009

Interactive Perception for Cluttered Environments

Robert Willimon
Clemson University, rwillim@clemson.edu

INTERACTIVE PERCEPTION FOR CLUTTERED ENVIRONMENTS

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

by
R. Bryan Willimon
December 2009

Advisor: Dr. Stanley Birchfield
Co-Advisor: Dr. Ian Walker
Committee Member: Dr. Adam Hoover

Abstract

Robotics research tends to focus upon either non-contact sensing or machine manipulation, but not both. This paper explores the benefits of combining the two by addressing the problem of extracting and classifying unknown objects within a cluttered environment, such as found in recycling and service robot applications. In the proposed approach, a pile of objects lies on a flat background, and the goal of the robot is to sift through the pile and classify each object so that it can be studied further. One object should be removed at a time with minimal disturbance to the other objects. We propose an algorithm, based upon graph-based segmentation and stereo matching, that automatically computes a desired grasp point that enables the objects to be removed one at a time. The algorithm then isolates each object to be classified by color, shape and flexibility. Experiments on a number of different objects demonstrate the ability of classifying each item through interaction and labeling them for further use and study.

TABLE OF CONTENTS

Table of Contents (Continued)

Page

List of Tables

Table Page

LIST OF FIGURES

Chapter 1

Introduction

Visual sensing and machine manipulation are well-studied topics within robotics research. Most of this effort, however, concentrates on only one topic or the other without considering the significant coupling of the two. To be sure, an important body of work has been aimed at using remote sensing to assist in real time with manipulation, e.g., visually-guided manipulation [4][14]. However, there has been relatively little work aimed at the reverse problem, namely, using manipulation to guide non-contact sensing in meaningful ways [26] [12] [7] [5].

Yet, humans routinely adopt this latter approach of “manipulation-guided sensing.” For example, we routinely shuffle through papers on a desk or sift through objects in a drawer to more quickly and efficiently identify items of interest. In such cases, it is our interaction with the environment that increases our understanding of the surroundings, in order to more effectively guide our actions to achieve the desired goal. In a similar manner, animals such as racoons [25] and cats also adopt this approach.

Cats are known to use their front paws to poke and swat at objects to better understand them, whether it involves playing with a toy or trying to catch a rodent. In these activities, one mode of information acquisition is via contact sensing, i.e., haptics. Haptics has been a topic of significant interest and activity in the past few years [10]. However, relatively little attention has been given to the use of the potentially much richer source of information available via vision during the environmental interaction.

As a first step in addressing this problem, Katz and Brock [12] describe a system in which a manipulator learns about the environment by interacting with it. Video available from an overhead camera is analyzed by tracking feature points on an object in order to determine the number, location, and type (revolute or prismatic) of joints on an articulated object lying on a table. To describe this process, they introduce the term "interactive perception." Interactive perception is a new approach towards autonomous manipulation. Rather than separating action from perception and solving each independently, this new methodology argues they should both be addressed at the same time.

Kenney, Buckley, and Brock [13] used an approach that involves a manipulator interacting with the environment to learn more about it. In this approach, a robotic arm interacts with objects on a table, and an overhead camera captures the scene. The video is used to subtract the current image from the previous image to locate the robotic arm or objects that have moved within that video frame. This information is used to locate objects and track them as they move around the table.

Inspired by the above work, this thesis introduces a new approach to interactive perception (or manipulation-guided sensing), in which successive manipulations of objects in an environment are used to increase vision-based understanding of that environment, and vice versa. The paper represents a development of the concept of interactive perception and demonstrates its usefulness in solving the problem of retrieving an object within a pile. We show that deliberate actions can change the state of the world in a way that simplifies perception and consequently future interactions. This interactive perception process should eventually lead to increased understanding of the environment and provide a robust and reliable solution to the problem of accurately retrieving and using objects within a cluttered environment.

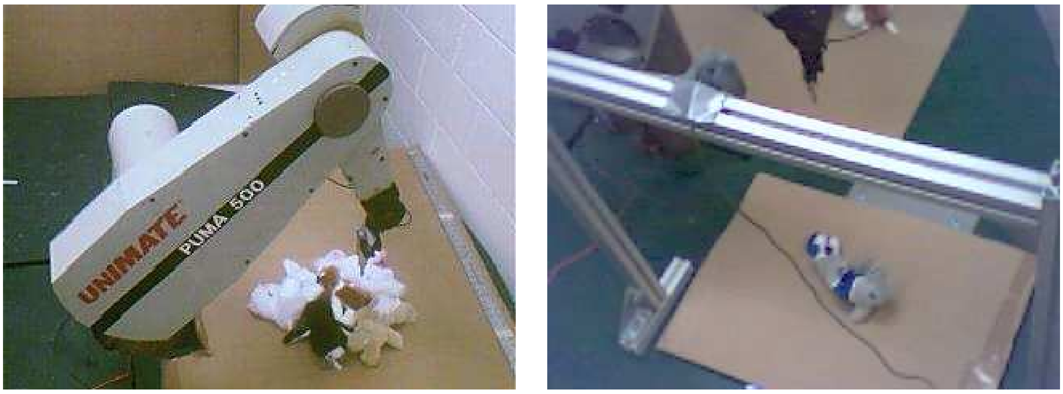


Figure 1.1: The proposed setting for manipulation-guided sensing. Left: A robotic arm interacts with a pile of unknown objects (stuffed animals) to segment the individual items one at a time. An overhead stereo pair of cameras (Logitech QuickCam Pro webcams) is not shown. Right: The system then learns about each object's characteristics by automatically isolating it from the pile and interacting with it. A third overhead camera is used for sensing the isolated object.

Based upon graph-based segmentation and stereo matching, our system examines a pile of unknown objects (see Figure 1.1) and determines the one that is most likely to be on top, for which a desired grasp point is then computed. Successive objects are removed from the pile and isolated for further study. Appearance models built of the isolated objects are then used to guide future interactions. The motivation for this work includes not only the extraction of harmless objects such as a sock in a drawer but also the discovery of delicate and possibly dangerous objects that are initially hidden from the field of view. For example, in searching for an explosive device a robot must be careful to gently remove the obstructing objects while disturbing the rest of the pile as little as possible.

Our work differs from that of Katz and Brock [12] in its purpose and scope. Rather than recovering geometric properties of a single unobstructed rigid articulated structure, our concern is with sifting through a cluttered pile of objects [22], many of which are occluding one another, and properly classifying each object. Unlike [12], our objects can be either rigid or non-rigid, and our results provide a skeleton of the object along with some geometric properties to more fully determine how the object can be classified and interacted with for further use.

Another piece of related work is that of Saxena et al. [19]. In that work, information about a scene is gathered to generate a 3D model of each object in the scene. The 3D model is then compared against a database of previously created models with grasping locations already determined. Their work explores more fully how to grasp the object instead of learning more about it, like in [2]. Our method is different in that the objects in the pile being examined in this work are unknown a priori up to a given scale. Therefore, we do not have access to any prior database of the objects being examined.

The work that we are proposing is being treated on a general level. Our approach involves “pushing” objects at desired locations once they are isolated. This idea of interacting with different actions (i.e. pushing, tapping, grabbing) is used to discover different attributes of the object. These attributes are then used to describe what the object is as well as what the object can do. In [15] [16] [21], various researchers use interactions to describe how objects act to different stimuli from the robot manipulator.

In [15] [16], for example, a robot was used to interact with a ball and the operater told the robot that the ball is green and it rolls when “pushed”. The robot learns about the color green and what the word “rolling” means by watching what the ball did after interaction with the robot. In [16], the robot captured how the object interacted to one of three interactions: “grasp”, “tap”, or “touch”. The robot was then used to imitate what it had learned from interacting with the object. In [21], their work was similar to that of [15] because they addressed the problem of learning about visual properties and spatial relations. Their work involved vision, communication and manipulation subsystems. Our work coincides with the same ideology in these works but without using communication or any verbal input. We use interactions to group like items together in a sorting method.

The work in this thesis can be extended and used for learning about an environment with service robots. Previous work involving service robot applications has been geared towards grasping [9] and folding clothes [18] [17]. Clothing is considered to be at the opposite end of the spectrum far away from rigid objects. Manipulating and interacting with non-rigid objects is still a largely unsolved problem for robotics. Most of the previous work involves interacting with rigid objects, because rigid items in a 2D environment are easy to characterize. That is why we explored this approach and the following algorithm is generalized to handle both rigid and non-rigid objects.

Chapter 2

Extracting and Isolating Unknown Objects within a Pile

2.1 Extraction overview

The extraction and isolation process is the first half of the entire system that is involved with learning about unknown objects for further study. We found that interacting with and learning about objects individually provide more accurate information regarding the characteristics of each unknown item. Figure 2.1 presents an overview of the extraction process. Each of the boxes is discussed in more detail in the following subsections. Graph-based segmentation [6] starts the process to divide the area into regions based on color. Next, stereo matching is used to determine the initial region in which to concentrate. After the selected region is chosen from the graph-based segmentation and stereo matching, a grasp point is selected to determine where and how to extract the object. Lastly, the manipulator sets the object aside to be classified to aid identification and manipulation of the object in future interactions.

To determine if there are no more objects in the scene, the currently selected region is tested to determine if it is smaller than a predetermined threshold of b_{min}. b_{min} was computed as

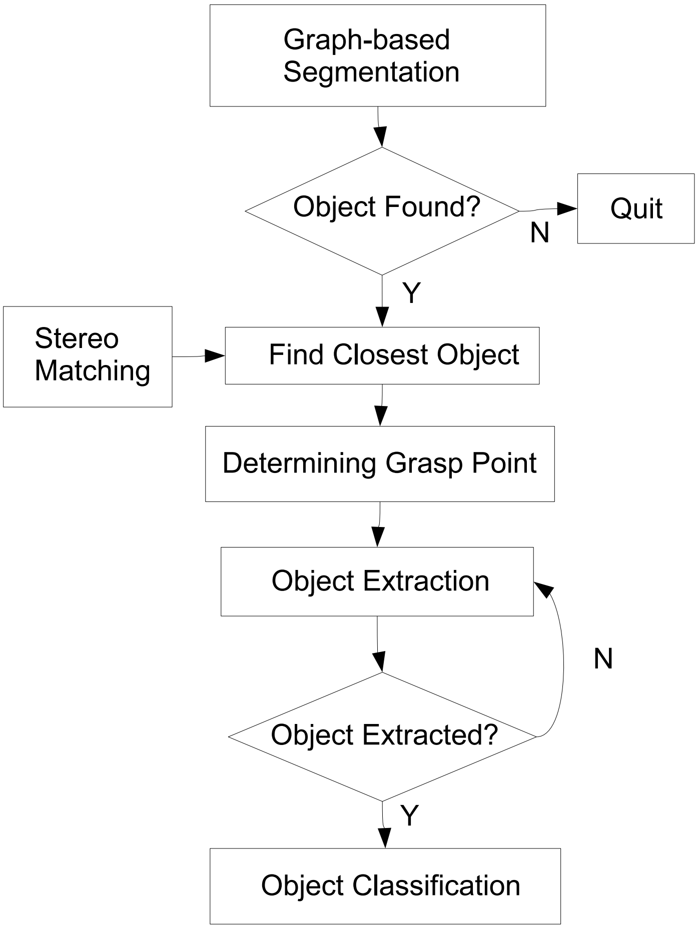


Figure 2.1: Overview of our system for vision-guided extraction of a pile of unknown objects.

the average size of an area that results from noise in the graph-segmentation algorithm. If the region is less than b_{min}, then the algorithm decides that there are no more objects in the image since the objects in our scenarios are on average larger than a size of b_{min}.

2.2 Graph-based segmentation

Graph-based segmentation [6] is an algorithm used to take an image and segment it into different regions based on feature vectors associated with the pixels (e.g., color). This algorithm forms the first step in the extraction process. Figure 2.2 shows the results of graph-based segmentation on an example image taken in our lab. The graph-based segmentation gives a layout of the original image broken up into various regions. Those regions are then examined to calculate the area of each region, whether the region is touching the border of the image, and the mean color value in the region.

Our initial goal is to determine which object is on top of the pile, so that a single object can be extracted from the scene while minimizing the chance of disturbing the surrounding objects. There are many monocular image cues that can provide a hint as to which object is on top, such as the size of the object, its concavity, T-junctions, and so forth. The graph-based segmentation facilitates the separation of background and foreground regions, thereby removing the former from consideration. The background is considered to be all the colors associated with pixels located on the border of the image. To decide among the foreground, we rely upon stereo matching using a pair of cameras. The idea behind looking for the object on top of a pile is so that the other regions (or objects) in the pile would not be disturbed when extracting the selected region.

2.3 Stereo matching

Stereo matching is the process in visual perception leading to the sensation of depth from two slightly different projections of an environment [20]. With rectified cameras, the difference in image coordinates between two corresponding points in the two images arise from the cameras' different positions along the baseline. This image difference is called horizontal

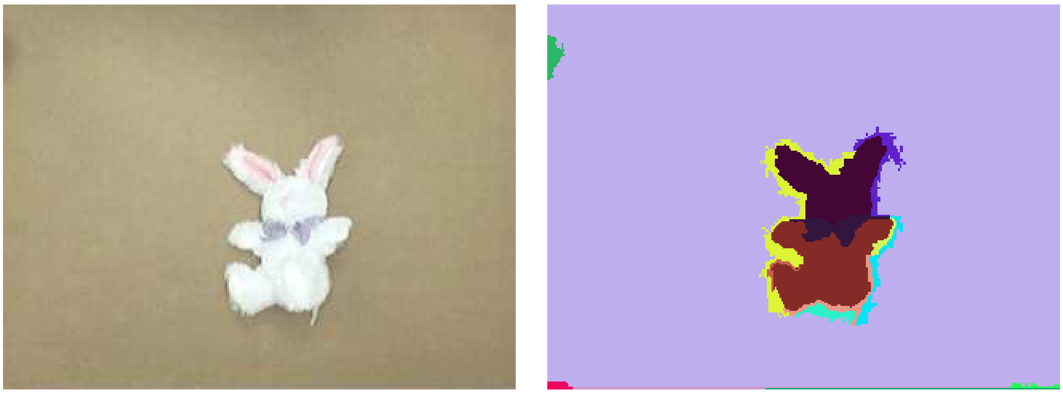


Figure 2.2: LEFT: An image taken in our lab. RIGHT: The results of applying the graph-based segmentation algorithm. Despite the over-segmentation, the results provide a sufficient representation for grasping the object.

disparity, and it is inversely proportional to the distance from the camera to the point. We implemented a window-based sum-of-absolute differences (SAD) stereo algorithm [20] for its computational efficiency, utilizing MMX/SSE2 SIMD operations to increase the speed of computation. The disparity image that results from stereo matching is used to estimate the relative distance to each segmented area found in the previous section.

The disparity image contains noise from misalignment of the cameras, reflections in the scene (non-Lambertian surfaces), and occlusion. To reduce the effects of this noise, we employ a left-right consistency check [8] to retain only those disparities that are consistent in both directions. We also use the graph-based segmentation image to separate the background from the foreground. Among the foreground disparities, connected constant-disparity regions whose area exceeds a threshold, a_{min}, are retained while smaller regions are discarded. (We set $a_{min} = 0.04\%$ of the image.) Figure 2.3 shows the results of the stereo matching before and after reducing the effects of noise. Photometric inconsistency between the cameras is handled by converting to grayscale, followed by adjusting the gain of one image to match the other.

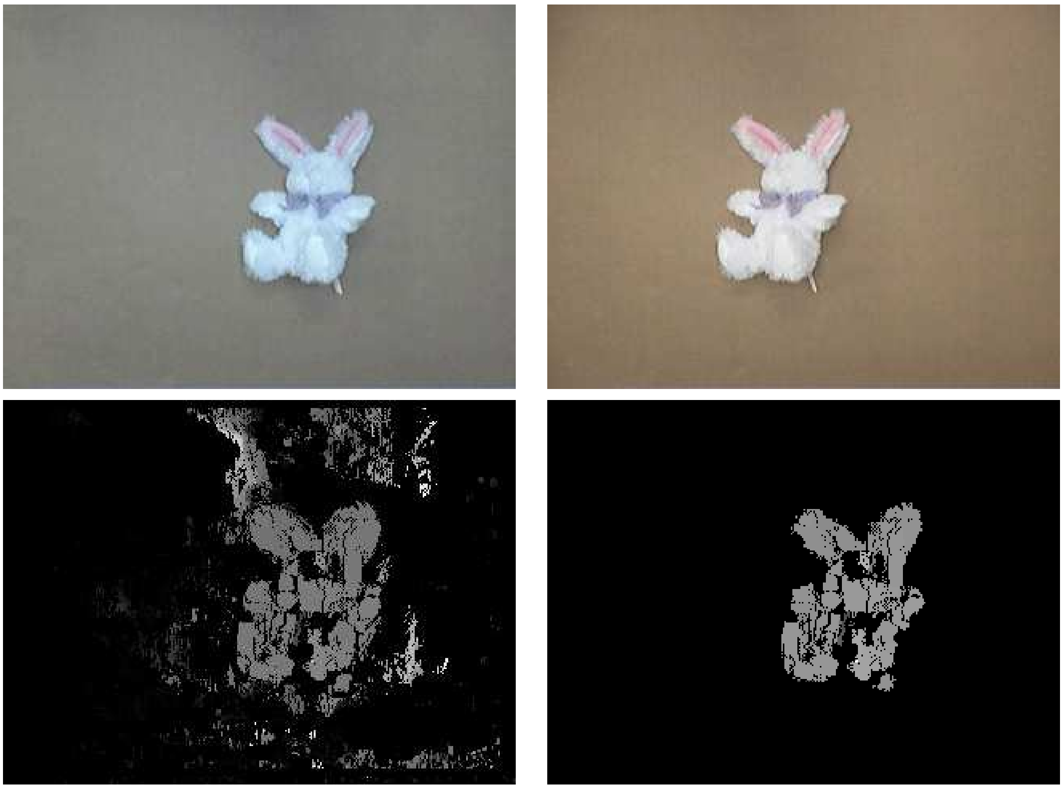


Figure 2.3: TOP: A stereo pair of images taken in our lab, showing the large amount of photometric inconsistency. BOTTOM: The disparity image obtained by SAD matching (left), and the result after masking with the foreground and removing small regions (right).

2.4 Determining grasp point

Determining the grasp point is a critical step for picking up the selected object. If the object itself or surrounding items are fragile or sensitive, then it is important to minimize the amount of disturbance. Because some of our objects are non-rigid and irregularly shaped, we cannot use the approach of [19] because they only involve rigid objects and train their system from a database of known objects. Instead, we calculate the 2D grasp point as the geometric center of the object, defined as the location whose distance to the region boundary is maximum. This point, which can be computed efficiently using chamfering [3], is much more reliable than the centroid of the region, particularly when the region is concave. The nature of the objects and grasp scenarios considered in this paper motivates contact at their geometric center, see section 4.

Figure 2.4 shows the grasp point found by the maximum chamfer distance for a selected object. Once the grasp point has been found, the robot arm is moved over the pile of objects, the end effector is positioned above the grasp point, the arm is positioned two inches away from the expected height given by the stereo cameras, and the arm lowers the end effector orthogonally to the image plane to grab the object and remove it from the scene. The arm is positioned closely to the expected height because when the arm is lowered in the orthogonal direction, the trajectory generator used in the robotic computer adds a small curvature to move from one point to another.

Once the arm has retreated from the camera's field of view, the images before and after the extraction procedure are compared to determine whether, in fact, the object was removed. This decision is made by comparing the number of pixels whose absolute difference in intensity exceeds a threshold and the size of the object. As long as the object has not been successfully removed, the arm is successively lowered a small amount to try again, until either the object is taken or the end effector has reached a maximum distance (to avoid collision with the table on which the object is sitting). These repeated attempts overcome the lack of resolution in our stereo imaging configuration. The extraction process continues until all of the objects in the image have been removed (i.e., when all of the segmented image regions that do not touch the image boundary are smaller than the minimum threshold, b_{min}).

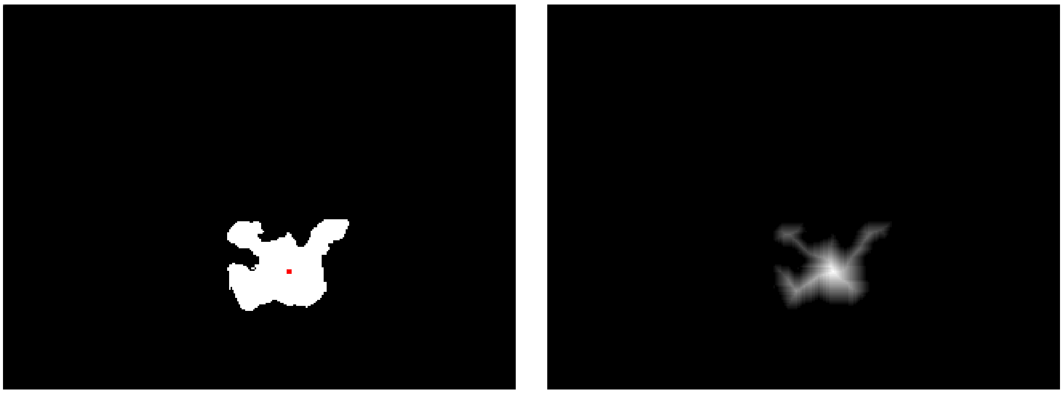


Figure 2.4: LEFT: The binary region associated with an object. The grasp point (red dot in the center) is the location that maximizes the chamfer distance. RIGHT: The chamfer distance of each interior point to the object boundary.

Chapter 3

Classifying and Labeling Individual Unknown Objects

3.1 Classification overview

The classification and labeling process is the second half of the entire system that is involved with learning about unknown objects for further study. We found that classifying objects by color, shape, and geometric properties provide a way to learn more accurately about what each object can do through interactions with the robot. Figure 3.1 presents an overview of the classification process. Each of the boxes is discussed in more detail in the following subsections. Color histogram labeling [23] begins the process by giving the object a color label. Color histogram labeling is just one tool to classify an unknown object with a specific label. This label can be used to identify the object in future encounters and separate the item based on its distribution of the red, green, and blue color values of the object within the image. Skeletonization is the next step in the classification process. Skeletonization provides a 2D model of the unknown object to be used in determining locations of revolute joints and interaction points.

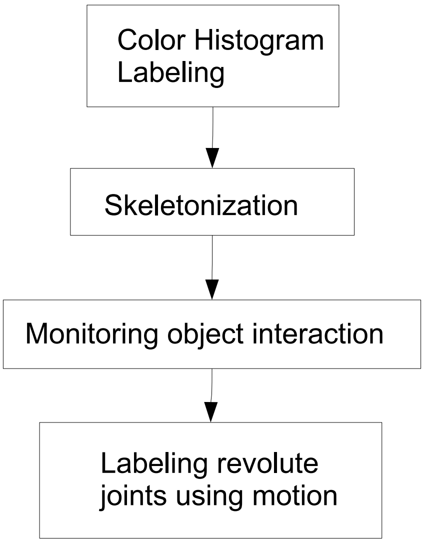

Color Histogram Labeling
Skeletonization
Monitoring object interaction
Labeling revolute joints using motion

Figure 3.1: Overview of our system for manipulation-guided classification of a pile of unknown objects.

Once the skeleton is constructed, the robotic arm is then used to interact with the object to further study it. Monitoring object interaction is the step that observes how the object reacts to the robotic arm's involvement. Finally, labeling revolute joints using motion uses the monitored interactions to determine joints on the object that are movable and others that are rigid. After the final step, a revised skeleton is created with the revolute joints labeled. This revised skeleton provides a more accurate classification of the unknown object and also gives insight towards calculating an educated grasp point and orientation to grab it with minimal error.

3.2 Color histogram labeling

When an object has been extracted from the pile, it is set aside in order to study it further. This procedure involves constructing a model of the object and comparing that model with

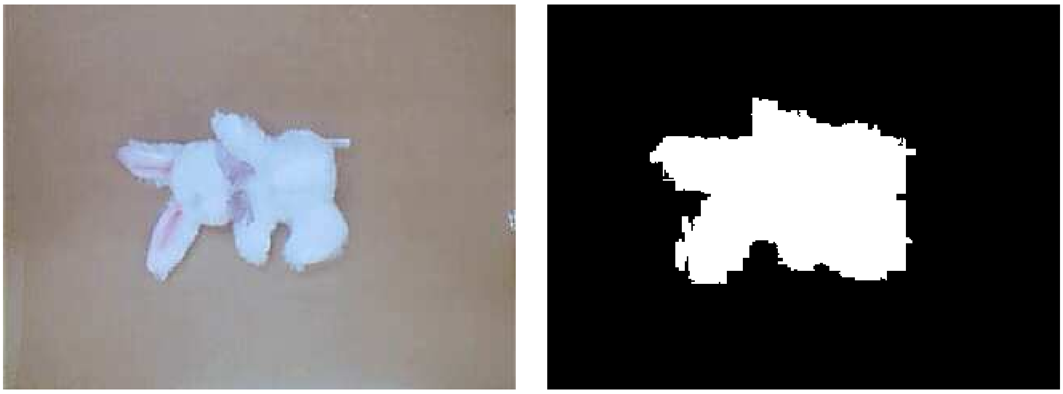


Figure 3.2: LEFT: The isolated object to be classified. RIGHT: The binary mask of the object used in constructing its color histogram model.

the models (if any) of previous objects that have been encountered. To compare objects we use a color histogram, both for its simplicity and for its robustness to geometric deformations. A color histogram is a representation of the distribution of colors in an image, derived by counting the number of pixels with a given set of color values [23]. The color space is divided into discrete bins containing a range of colors each. We use eight bins for each color range of red, green, and blue, leading to 512 total bins.

Objects are matched by comparing their color histograms. For this step, we use histogram intersection [23] which is conveniently affected by subtle differences in small areas of color while at the same time being guided by the dominant colors. The histogram intersection is normalized by the number of pixels in the region, leading to a value between 0 and 1 that can be interpreted as the probability of a match. By comparing the color histogram of the currently isolated object with histograms of previously encountered objects, we are able to determine the identity of the object. If the probability is high enough, then the objects are considered to be the same; otherwise, the current object has not been seen before and its histogram is therefore added to the database. (We set the minimum probability for this decision to be 70%.) Figure 3.2 shows the isolated object to be classified, along with its binary mask.

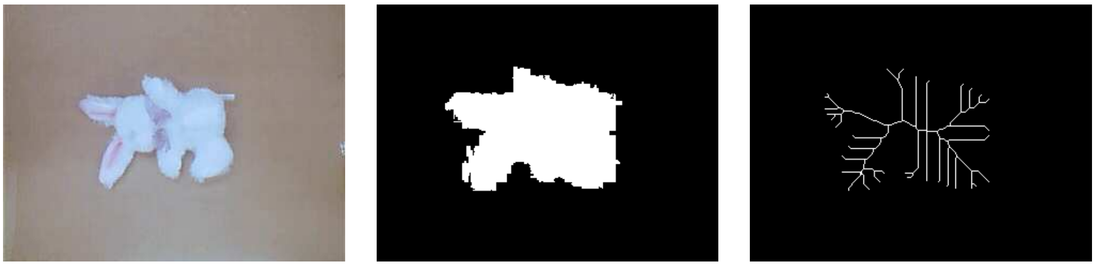


Figure 3.3: LEFT: The original object. MIDDLE: The isolated object to be classified. RIGHT: The skeleton of the object.

3.3 Skeletonization

Skeletonization is the process of designing the internal outline of an object. The general way of describing how a skeleton is formed is by using the prairie-fire analogy. The boundary of an object is set on fire and the skeleton is the loci where the fire fronts meet and quench each other [1]. The skeleton is a single-pixel wide outline of the object. This outline gives a representation that illustrates where the extremities or limbs of the item are located. The locations where the extremities meet the torso or inner-most line are considered intersection points, and locations where extremeties begin are considered end points. The intersection points are the initial guesses of the locations of the objects joints. End points are positions that are used to interact with the robotic arm. Figure 3.3 gives an example of a box with its skeleton.

3.4 Monitoring object interaction

Monitoring object interaction involves tracking feature points within an image to understand the movements and overall makeup of the object. This process is facilitated by using the Kanade-Lucas-Tomasi (KLT) feature tracker [24]. The KLT algorithm selects feature points in an image and then maintains their location in the image as they move due to scene changes resulting from camera or object motion. Detecting features in the image is used when the selected region is found by separating the foreground from the background using graph-based segmentation.

After the KLT algorithm provides features within the whole image, only the features within the selected region are kept, and the other features are discarded. Figure 3.4 shows the KLT features for the whole image on the left and only the features within the selected region on the right. Before KLT feature detection is applied, the region is dilated to include the features along the edge of the object. Dilation is the process of taking an object within an image and increasing the size of the object by 1 or more pixels, depending on how many times the algorithm is run. Finally, the end points mentioned previously are used to instruct the arm where to interact with the object.

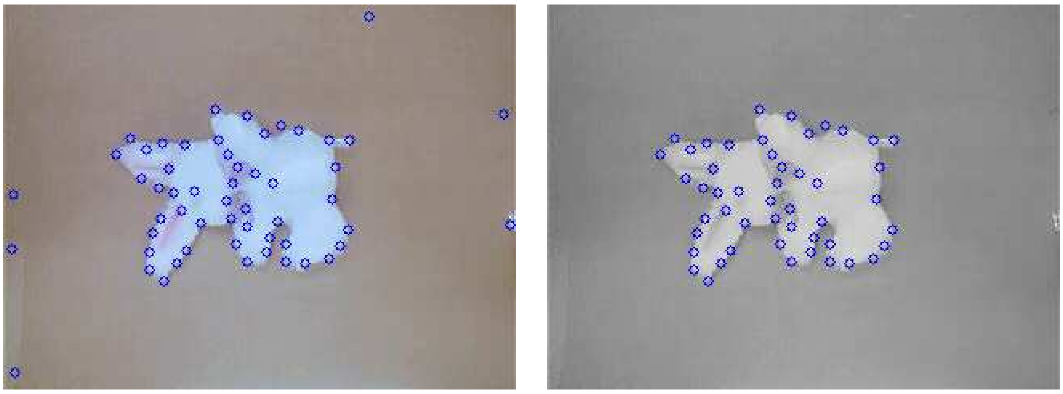


Figure 3.4: Left: KLT features selected in the whole image. Right: Features that are located within the region selected by graph-based segmentation.

The interaction of the arm “pushes” the object at each end point orthogonal to the object. The end effector of the arm is placed two inches away from the object in the direction of the vertical or horizontal axis of the image plane depending on the location of the end point. Each end point is placed within a vector of coordinates to decide which end points will give the most information. Each end point is compared against the rest of the vector to determine if any one point is close to another point. An end point is close to another point if and only if the

Euclidean distance between the two points is below a threshold, we used a threshold of 5. The remaining end points were grouped based on if the point was closer to the top of the image or to the left of the image. If the point was closer to the top of the image, then the end effector will move in the vertical direction of the image plane, otherwise it will move in the horizontal direction of the image plane.

As the arm interacts with the object, the algorithm groups various clusters of features based on distance and direction values. The process then checks to see if the current number of frames in the video feed from the camera is a multiple of f_{length}. If the number is a multiple of f_{length}, then the algorithm segments parts of the image using the clustering algorithm. If not, then it continues monitoring the item. In our experiments, we set f_{length} to be 5 frames.

Figure 3.5 shows an example of clustering feature points. On the left side, the figure shows the feature points of the original object before clustering occurs. On the right side, the figure shows the feature points after f_{length} frames of video and the separation of two different groups. The boundary line indicates the distances between the two groups that are greater than the prespecified threshold (which we set to 10). This example could also result in 3 separate groups emerging after the clustering process occurs.

In the work of [12], small groups with three or fewer features are discarded from the image. However, in our approach we have found that such groups hold some significant value in the case that some feature points attached to the object have been lost, resulting in a small group of features remaining. Therefore, in our algorithm all groups with at least two features are retained. As for groups of size one, the single feature point is then attached to the nearest group, using the Euclidean distance in the image.

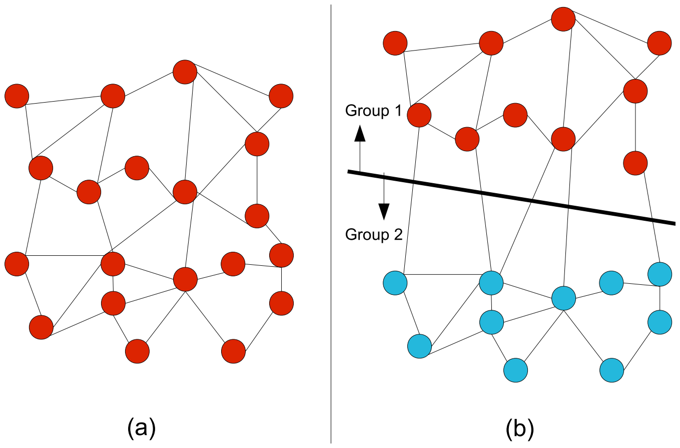


Figure 3.5: Example of clustering feature points according to inter-distance values, in Euclidean space. (a) Before clustering and (b) after clustering with decision boundary.

3.5 Labeling revolute joints using motion

Labeling revolute joints using motion coincides with the monitoring process using KLT tracking. The features are clustered based on the Euclidean distances between them in the image plane. Features with similar motion vectors have relatively constant inter-feature distances and are therefore placed in the same group, while features with different motion vectors are separated into distinct groups. Using features to monitor the object gives the characteristic of which areas are moving and which ones are not.

After the selected region has been divided into one or more groups, an arbitrary feature point in each group is analyzed to decide if that group has moved more than the rest of the groups in the image. The assumption is that the interacted region is moving, while the other areas remain relatively stationary. The group whose computed motion is greater than the prespecified threshold is then determined to be movable and connected to a revolute joint.

If the feature group size is larger than one, then the surrounding ellipse of the feature group is used to calculate the end points along the major axis, illustrated in Figure 3.6, using principle component analysis (PCA) [11]. The end points of the major axis are considered the location of the revolute joint and the end of the rigid link. If the feature group size is only one point, then the intersection point closest to that feature is considered to be a revolute joint. Figure 3.7 gives an example of the mapping between feature points and intersection points within an image.

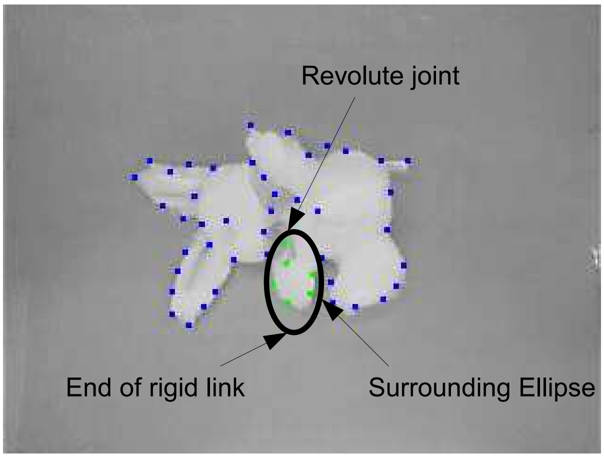


Figure 3.6: Example of grouping feature points to locate revolute points near the endpoints of the major axis.

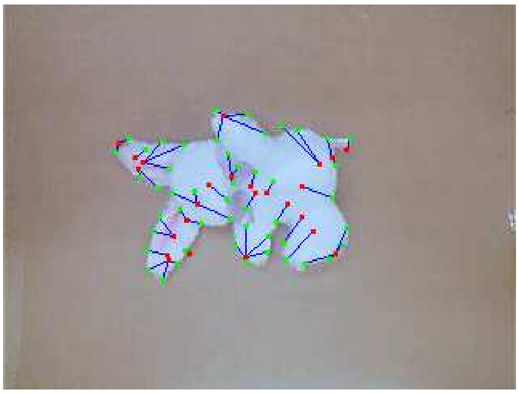


Figure 3.7: Example of mapping feature points to the nearest intersection point. The red dots represent the intersection points (possible revolute joints) of the skeleton. The green dots represent the feature points gathered by KLT. The blue lines connecting various points together represents the mapping of the feature point to the closest intersection point of that feature.

Figure 3.8 illustrates the initial skeleton labeled with intersection points and end points and the revised skeleton with revolute joints labeled after several interactions with the robotic arm. The new skeleton is designed with movable joints labeled and all other joints deemed rigid. The end points associated with rigid joints are removed because any branches that do not have a revolute joint are considered noise in the skeleton. Only branches with movable joints are considered extremities of the object.

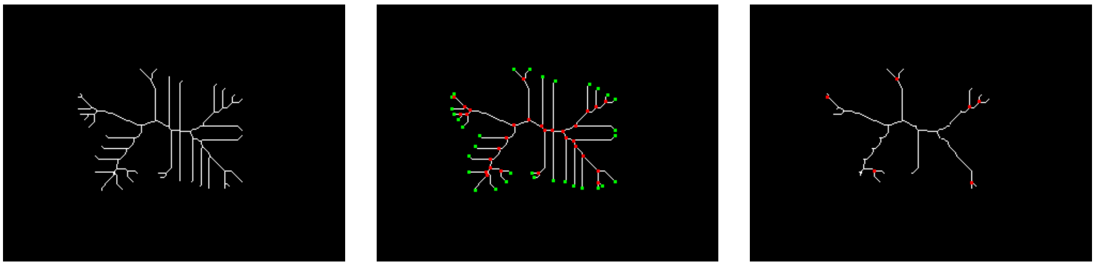


Figure 3.8: Left: The original skeleton. Middle: Skeleton with intersection points and end points labeled. Right: Revised skeleton after multiple interactions. The red dots represent the intersection points (possible revolute joints) of the skeleton. The green dots represent the end points (interaction points) of the skeleton.

Chapter 4

Experimental Results

4.1 Platform

The proposed approach was applied in a number of different scenarios to test its ability to perform practical interactive perception. These scenarios were created to simulate real world experiences and interactions. Using our approach, a PUMA 500 robotic arm was used in these experiments when extracting the objects from a pile of unknown items. In each experiment, a pile of objects rested upon a flat, uniform background. The objects themselves, their type, and their number were unknown beyond being above a specific minimum size. The system did not contain a database of known items, in contrast with [19].

The hardware used in our lab to run these experiments consisted of three Logitech QuickCam Pro webcams, a PUMA 500 robotic arm, a computer to operate the robot arm using QNX operating system, and a computer to run the software. The three cameras were used in the vision portion of the algorithm. Two of the cameras were used together as a stereo pair placed above the unknown pile orthogonal to the table. The stereo cameras were used to gather depth information of the unknown pile of objects. The third camera was used in the classification process. The third camera was placed to the side of the arm, near the area of the unknown pile, above the isolated object orthogonal to the table. The main computer that the software ran on was connected to the three cameras through a USB connection and also connected serially to the computer that operated the PUMA 500.

Figure 4.1 illustrates the setup of connecting the main computer to the robotic computer as well as connecting the three webcams through a USB port. Figure 4.2 shows the actual setup within the lab. Figure 4.3 shows the gripper used in these experiments. The gripper associated with the PUMA 500 contained a small opening to grasp objects. The gripper was rearranged to allow for a pair of salad tongs to be placed within it. The tongs were used as an extension to provide a wider opening at the end of the gripper for more robust grasping. The tongs were fastened to the gripper with the use of duct tape.

The entire process of extraction and classification is described in the following steps. The software pseudocode of the vision system and the robotic system can be found in Appendices A and B. The software for the vision system was written in Visual Studio C++ 6.0 on a Windows XP OS and the software for the robotic system was written in C on a QNX OS. The format in which the serial communication is being illustrated in Figure 4.4.

- Initialize serial port on vision and robot computer
- Use cameras from vision computer to find target region along with depth information
- Determine grasp point for target region
- Convert grasp point into (X,Y,Z) coordinates
- Send (X,Y,Z) coordinates over serial port to robot computer
- Robotic arm moves to position and extracts object using gripper
- Robotic computer then replies to vision computer of completion via serial port
- Vision computer informs robotic computer to place object in classification area

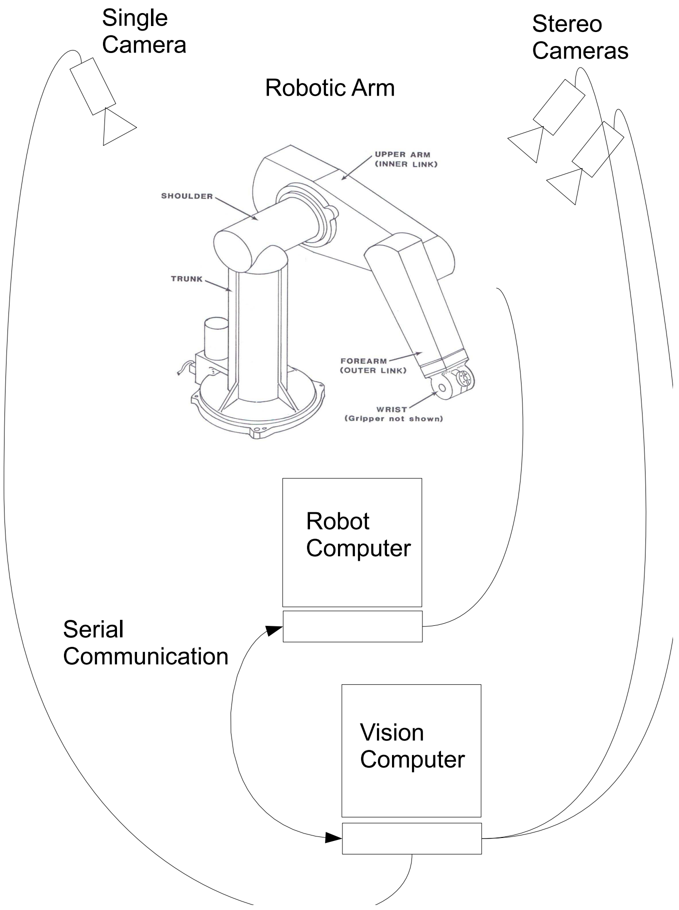


Figure 4.1: The setup and connection of the robot computer, main computer, and three webcams.

- Vision computer provides coordinates of classification area
- Robotic arm moves to classification area, releases object, and moves out of view of camera
- Robotic computer then replies to vision computer of completion via serial port
- Vision computer finds binary mask and computes skeleton

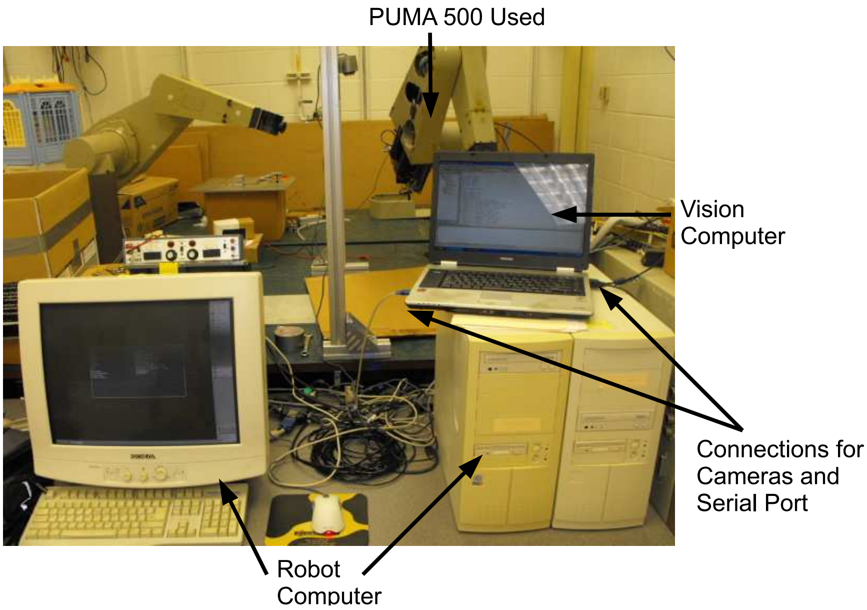


Figure 4.2: The actual setup within our lab.

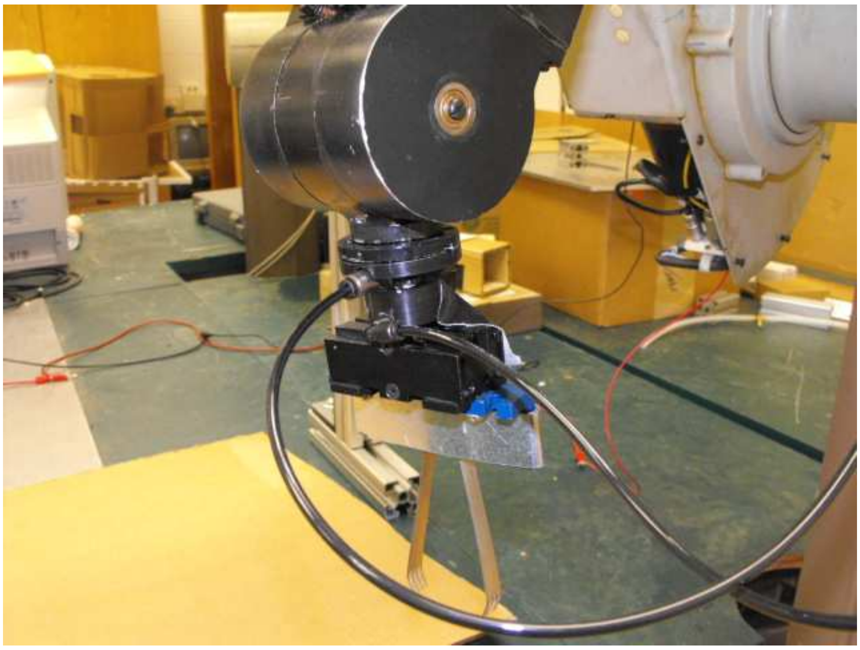


Figure 4.3: The revised gripper used in our experiments.

- Vision computer also locates feature points on objects and tracks feature points throughout all interactions
- Vision computer sends (X,Y,Z) coordinates of first interaction location via serial port
- Robotic arm interacts with object at the (X,Y,Z) location
- Vision computer continues sending each coordinate of interaction locations
- Robotic arm interacts at each location one at a time
- Vision computer monitors interactions to produce a final skeleton
- Vision computer informs robotic computer to extract object from classification area
- Robotic computer extracts object and places it at a specified location that is out of view of any camera
- The entire process starts again until no more objects are left in pile

The results of the system are displayed below at the different steps of the algorithm to allow the user to view what the robot is currently viewing, which object was selected for extraction, and the results of interacting with the object. With each step the algorithm finds a selected object to first extract. The PUMA arm is then sent to the grasp point to take the object out of the picture and isolate it. The arm then labels the object based on its color. Next, the arm interacts with the object at calculated areas based on the skeleton of the object. Finally, a 2D model of the object is created after monitoring the feature points on the object.

This approach simplifies the perception of an environment by using manipulation to interact with it. The idea of using vision alone to understand about a scene can provide a limited amount of information. The process of interaction allows the robot to become involved with the environment and also become a part of it. Also, being involved with the environment gives

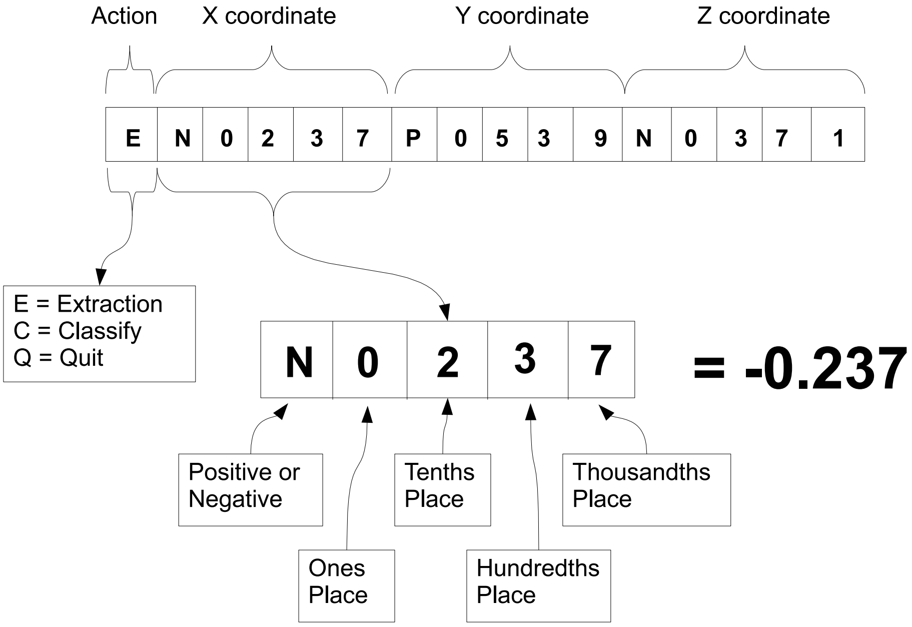


Figure 4.4: The serial communication format.

the robot a sense of what is in the environment along with how it moves and reacts to other objects.

When determining if no more objects were found, a value of 400 pixels was used for b_{min}. The entire system, from image input to extraction to classification to manipulation, is automatic. The items used for the initial experiments were soft stuffed animals — a good example of non-rigid objects that can move freely in unpredictable directions. In later experiments, metal and plastic objects were used to simulate a recycling bin. Socks and shoes were also used to simulate a pile of clothes in a hamper.

4.2 Small group experiment

Figures 4.5, 4.6, and 4.7 show the individual steps resulting from an experiment involving four objects. The image is segmented, the algorithm finds an object and determines a grasp point, then the robot grasps the object and sets it aside to a predetermined location to the left of the pile. After a color histogram model of the object has been constructed, a skeleton of the object is created to locate possible points of interaction and revolute joints. The robot then pokes the object at calculated locations and monitors the movement of the object in response to the robot's interaction. After a final skeleton is determined, the object is removed from the field of view. The process is then repeated as the system finds, extracts, and examines each object in turn, until there are no more objects remaining in the pile.

4.3 Classification experiment

We repeated the experiment with a larger set of eight unknown objects to demonstrate the classification process and the possible uses of labeling individual objects for further learning. Each time an object was extracted, the system captured an image of the isolated object, along with its binary mask and final skeleton. Figure 4.8 and 4.9 shows the eight images that were gathered automatically by the system as the objects were removed from the pile. The mask shows which pixels within the image were used for constructing the color histogram. The skeleton shows the 2D outline model of the object along with the revolute joints.

After the database of histograms was built, the objects were randomly rearranged in a new pile to test the classification performance of the system. As the objects were extracted again from the new pile, the color histogram of each object was compared against the database to determine the most likely match. If more than one color histogram was larger than Th_{min}, then the skeleton was used as a second form of classification to further identify the object (we set Th_{min} to 70%). Information calculated from the skeleton that could be used to further separate unlike objects are the number of extremities, or the number of revolute/non-revolute joints. During the experiments reported here, only the number of revolute joints on the object was used. Figures 4.10 and 4.11 shows the images gathered in the second run along with the

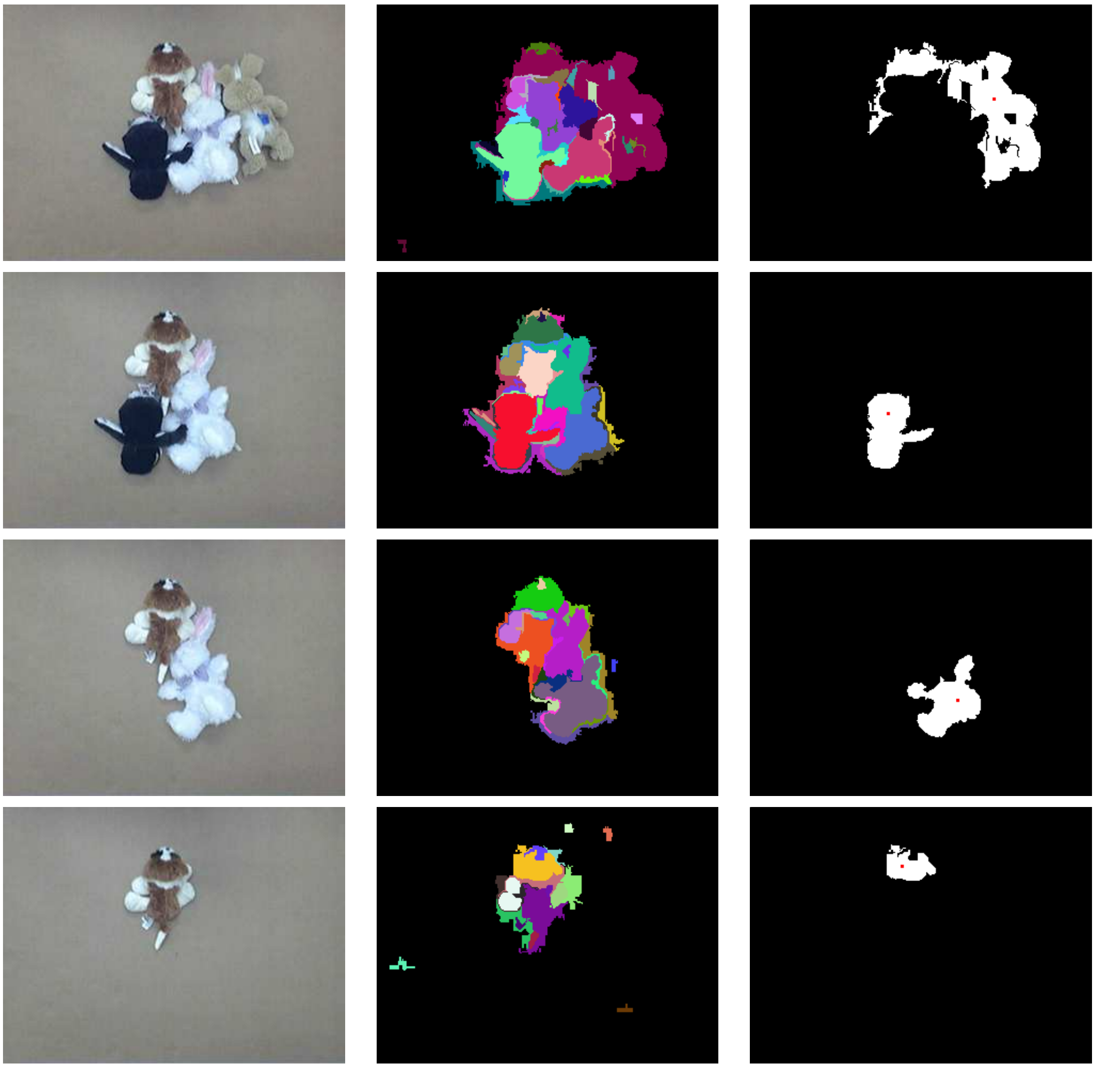


Figure 4.5: An experiment involving a pile of four unknown objects. From left to right: The image taken by the left camera, the result of graph-based segmentation, and the object found along with its grasp point(red dot). Time flows from top to bottom, showing the progress as each individual object is located and extracted.

best matching image from the first run. These results demonstrate that the color histogram and skeleton are fairly robust to orientation and non-rigid deformations of the objects.

The confusion matrix is shown in Table 4.1, indicating the probability (according to the color histogram intersection) of each query image matching each database image. The higher

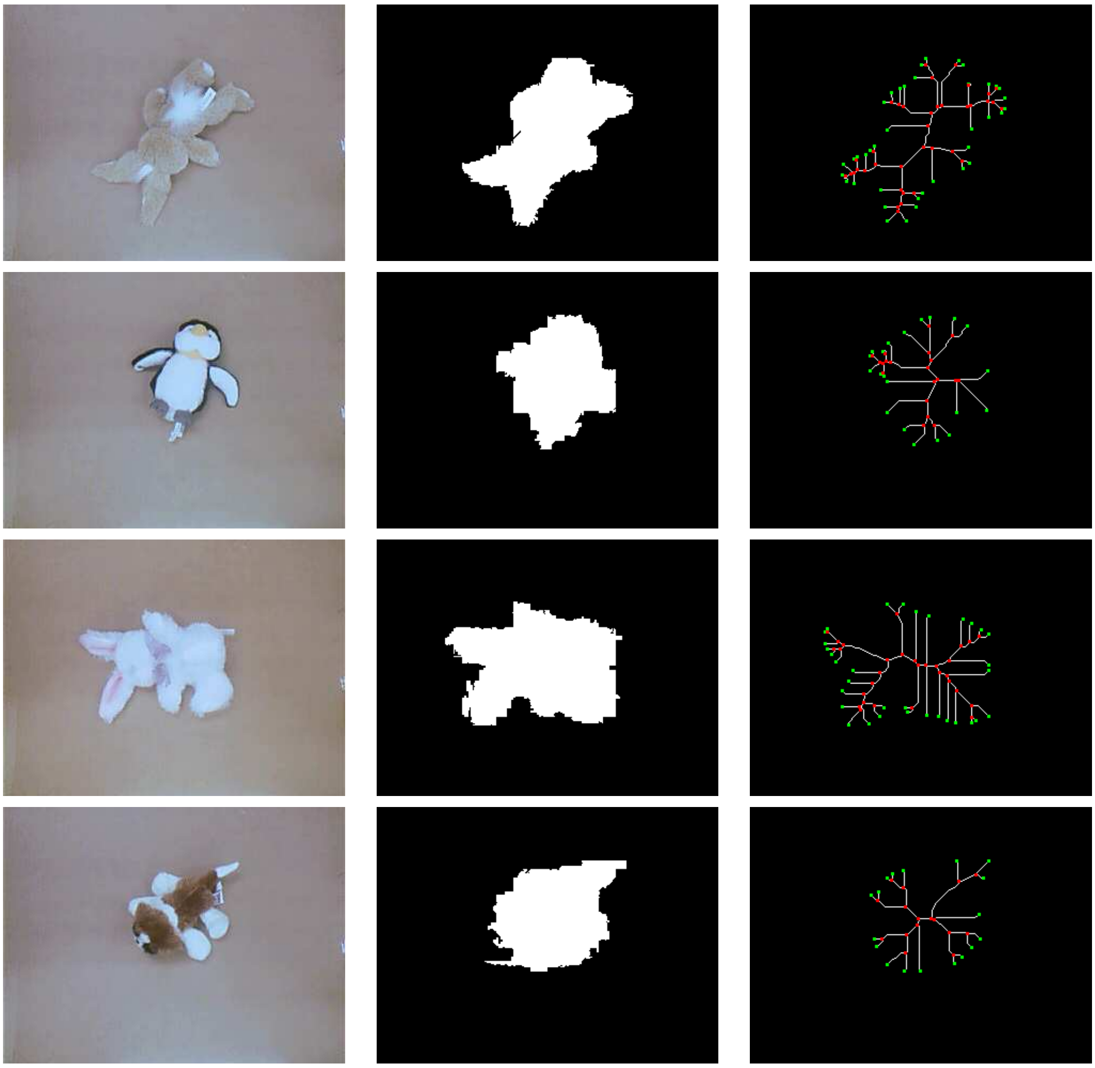
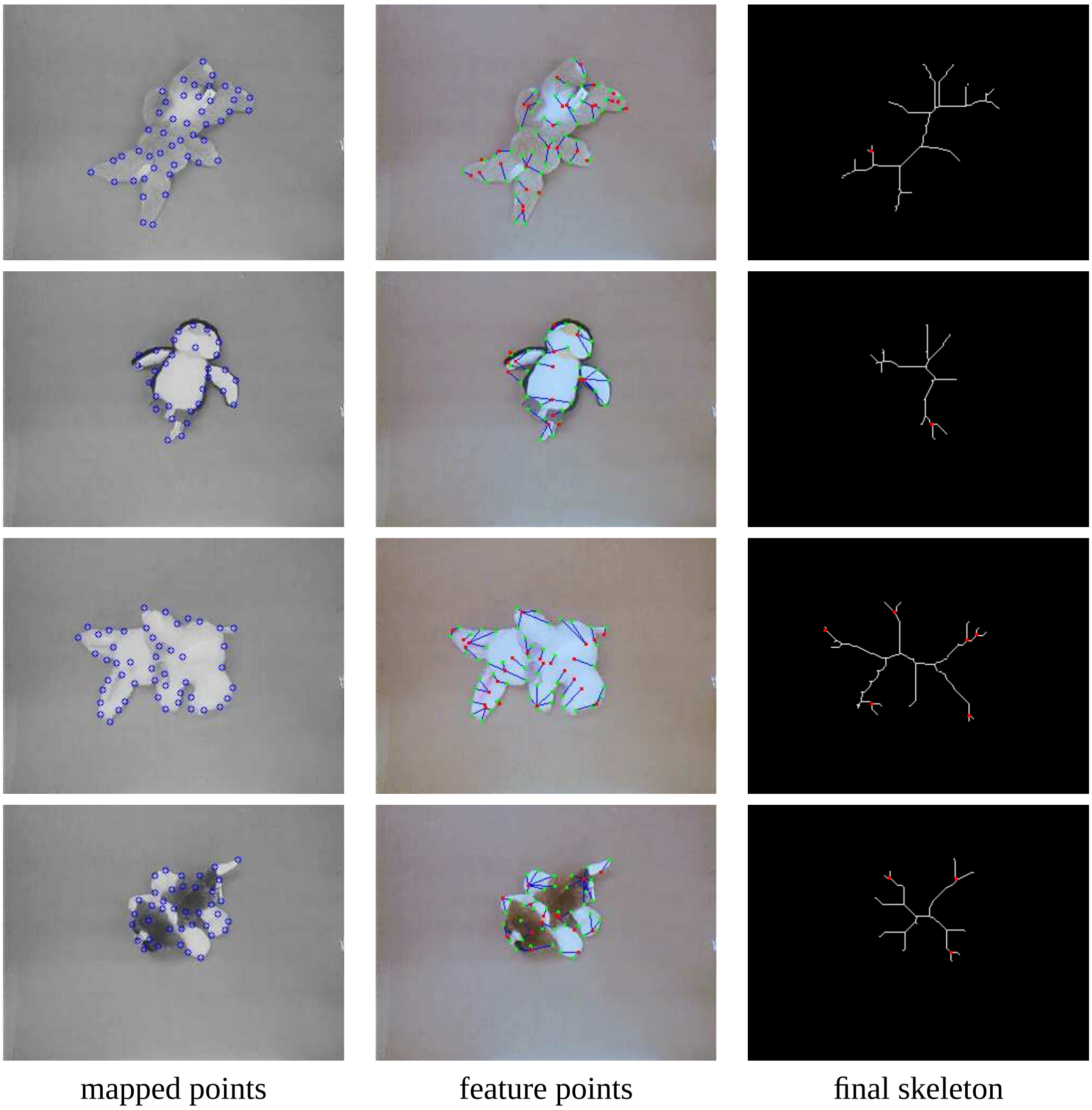


Figure 4.6: The same experiment involving a pile of four unknown objects. From left to right: The image, taken by 3rd camera, after the object has been picked up and separated, the binary mask of the isolated object, and the skeleton with the intersection points and end points labeled. Time flows from top to bottom, showing the progress as each individual object is examined.

the value, the more likely the two images match. Bold is used to indicate, for each query image, the database image that exceeds the Th_{min} threshold. In addition, recalling the classification threshold of Th_{min} described in section 3.2, each of the matches is considered valid


mapped points

feature points

final skeleton

Figure 4.7: Continuing the same experiment from Figure 4.6. From left to right: The feature points gathered from the isolated object, the image after mapping the feature points to the intersections points, and the final skeleton with revolute joints labeled. Time flows from top to bottom, showing the progress as each individual object is examined.

except for #6, which is 0.01% below the threshold. As a result, this item would be incorrectly labeled as one that had not previously been seen.

In the cases of #1 and #7, the query image matched to two separate database images. In order to decide which query image was correctly related to a database image, the number

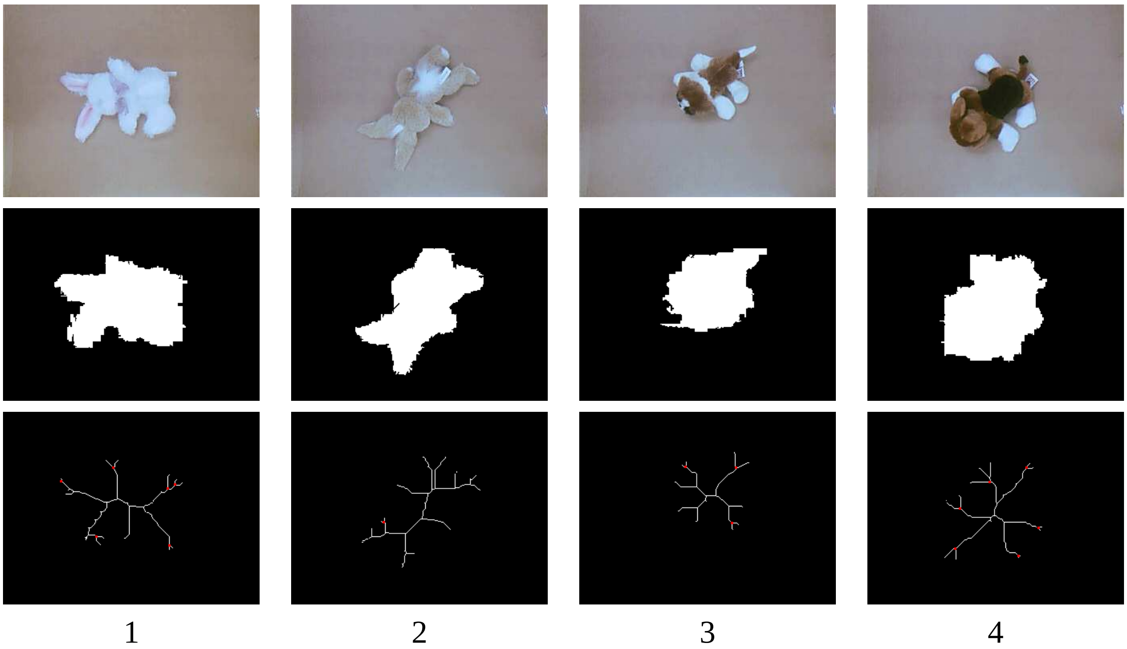


Figure 4.8: Top: Images of the individual objects gathered automatically by the system for the purpose of creating a database of objects previously encountered. Middle: The binary masks used for building the color histograms of the objects. Bottom: The final skeletons with revolute joints labeled.

Table 4.1: Evaluating Probabilities of stuffed animals: The rows represent query images and the columns represent database images.

#	1	2	3	4	5	6	7	8
1	**0.84**	0.27	0.25	0.22	0.15	0.27	**0.92**	0.18
2	0.33	**0.81**	0.35	0.39	0.26	0.38	0.46	0.33
3	0.54	0.50	**0.70**	0.67	0.45	0.40	0.56	0.36
4	0.41	0.45	0.60	**0.88**	0.56	0.41	0.39	0.48
5	0.19	0.19	0.25	0.41	**0.90**	0.20	0.15	0.42
6	0.39	0.51	0.32	0.33	0.27	**0.69**	0.47	0.35
7	**0.78**	0.36	0.28	0.24	0.16	0.33	**0.97**	0.25
8	0.29	0.33	0.37	0.61	0.51	0.33	0.24	**0.83**

of revolute joints were calculated. For each query image, the corresponding database image contained a higher number of revolute joints than the other database image, which clearly decided the correct objects to be matched. If the probability values were the only form of

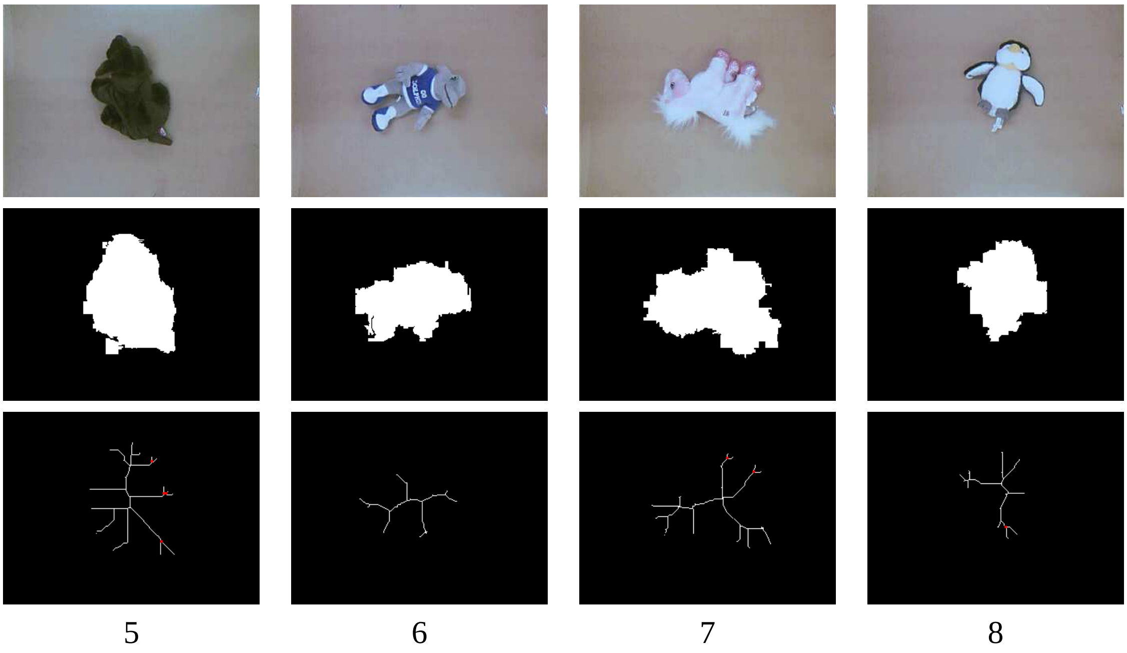


Figure 4.9: TOP: Images of the individual objects gathered automatically by the system for the purpose of creating a database of objects previously encountered. MIDDLE: The binary masks used for building the color histograms of the objects. BOTTOM: The final skeletons with revolute joints labeled.

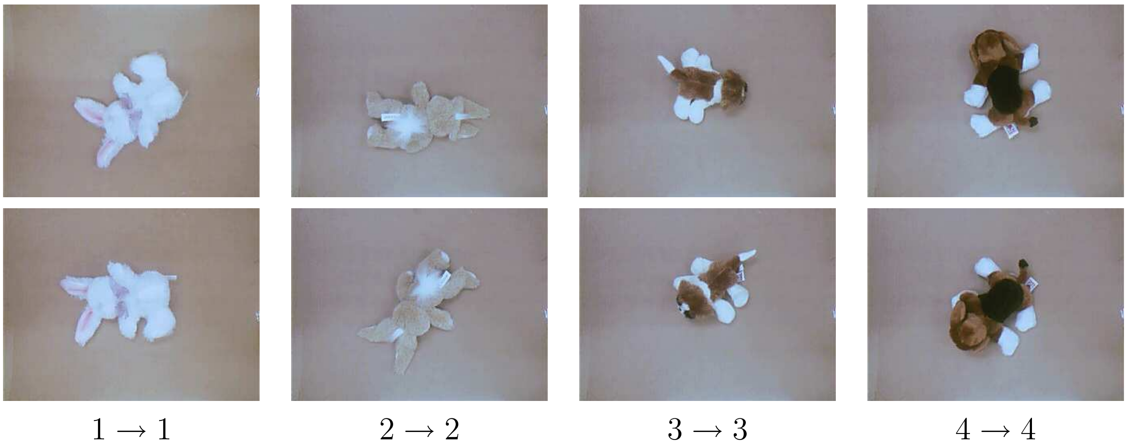


Figure 4.10: Results from matching query images obtained during a second run of the system (top) with database images gathered during the first run (bottom). The numbers indicate the ground truth identity of the object and the matched identity.

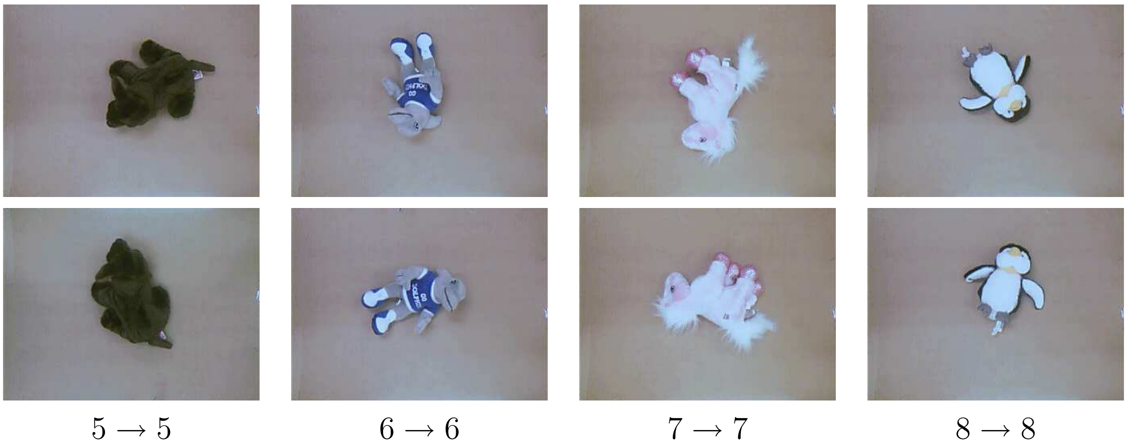


Figure 4.11: Results from matching query images obtained during a second run of the system (top) with database images gathered during the first run (bottom). The numbers indicate the ground truth identity of the object and the matched identity.

classification, then query image #1 would be misclassified. In this case, the skeleton was needed to properly classify the object.

4.4 Recycling using Metal and Plastic

A practical example where the proposed approach could be particularly useful is that of separating items to be recycled from a pile of metal and/or plastic objects which are often thrown into a container without any organization. This experiment tests the ability of the system to locate small objects within a pile and be able to extract them for sorting, as in a recycling plant. We used bottles and cans which are representative types of the objects that may be found within a recycling container. The algorithm was able to distinguish between each item for extraction until all of the objects had been removed.

Each of the items were treated equally, instead of (for example) plastic objects having a priority over metal objects. After the item was extracted and classified, a sorting algorithm could be used to decide in which bin the object should be placed. Note that after the objects have

Table 4.2: Evaluating Probabilities of metal and plastic: The rows represent query images and the columns represent database images.

#	1	2	3	4	5
1	**0.87**	0.46	0.56	0.40	0.42
2	0.43	**0.94**	0.59	**0.76**	0.53
3	0.39	0.56	**0.96**	0.39	**0.77**
4	0.41	**0.80**	0.62	**0.79**	0.46
5	0.33	0.57	**0.84**	0.39	**0.95**

been set aside for individual examination, it is much easier to determine their characteristics for such a sorting procedure than when they are cluttered in the entire group. The results of the experiment are shown in Figures 4.12, 4.13, 4.14, and 4.15.

Due to the limitations of our current gripper, whenever the algorithm computed a grasp location, a human manually grasped the object at that location using an “EZ Reacher”, which is an aluminum pole with a handle that, when squeezed, causes two rubber cups at the other end to close, enabling extended grasping. While the human was grasping and moving the object, the algorithm continued to run as if a robot were in the loop. As a result, no modification to the algorithm was made.

The confusion matrix regarding the recycling experiment is in Table 4.2, indicating the probability of each query image matching each database image. All of the cases, except for query image #1, were involved with having 2 query images match containing a probability value higher than Th_{min}. The deciding factor for grouping the correct images together was finding the revolute joints for each object. For each query image, the corresponding database image contained a higher number of revolute joints than the other database image, which clearly decided the correct objects to be matched, as in the previous experiment.

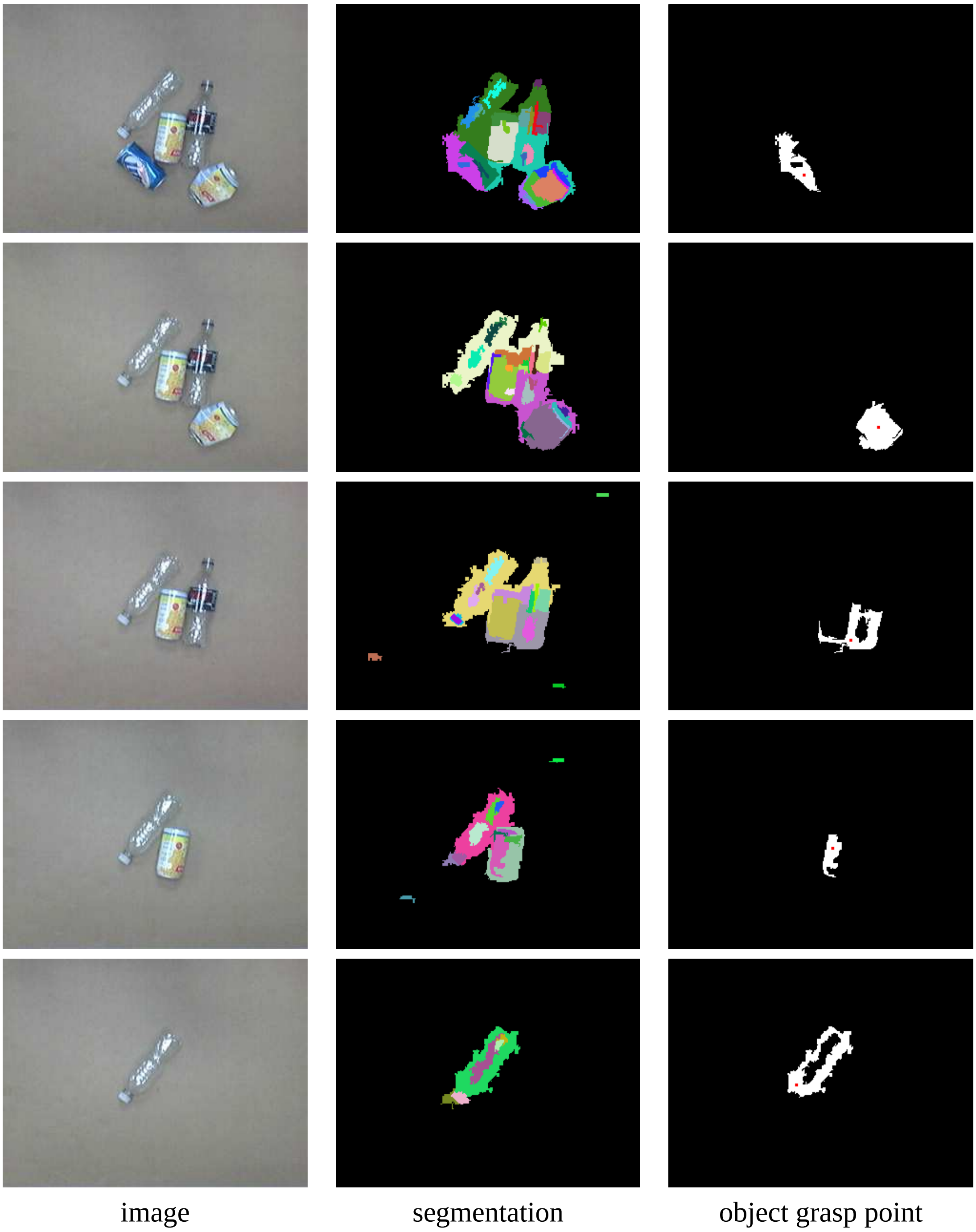
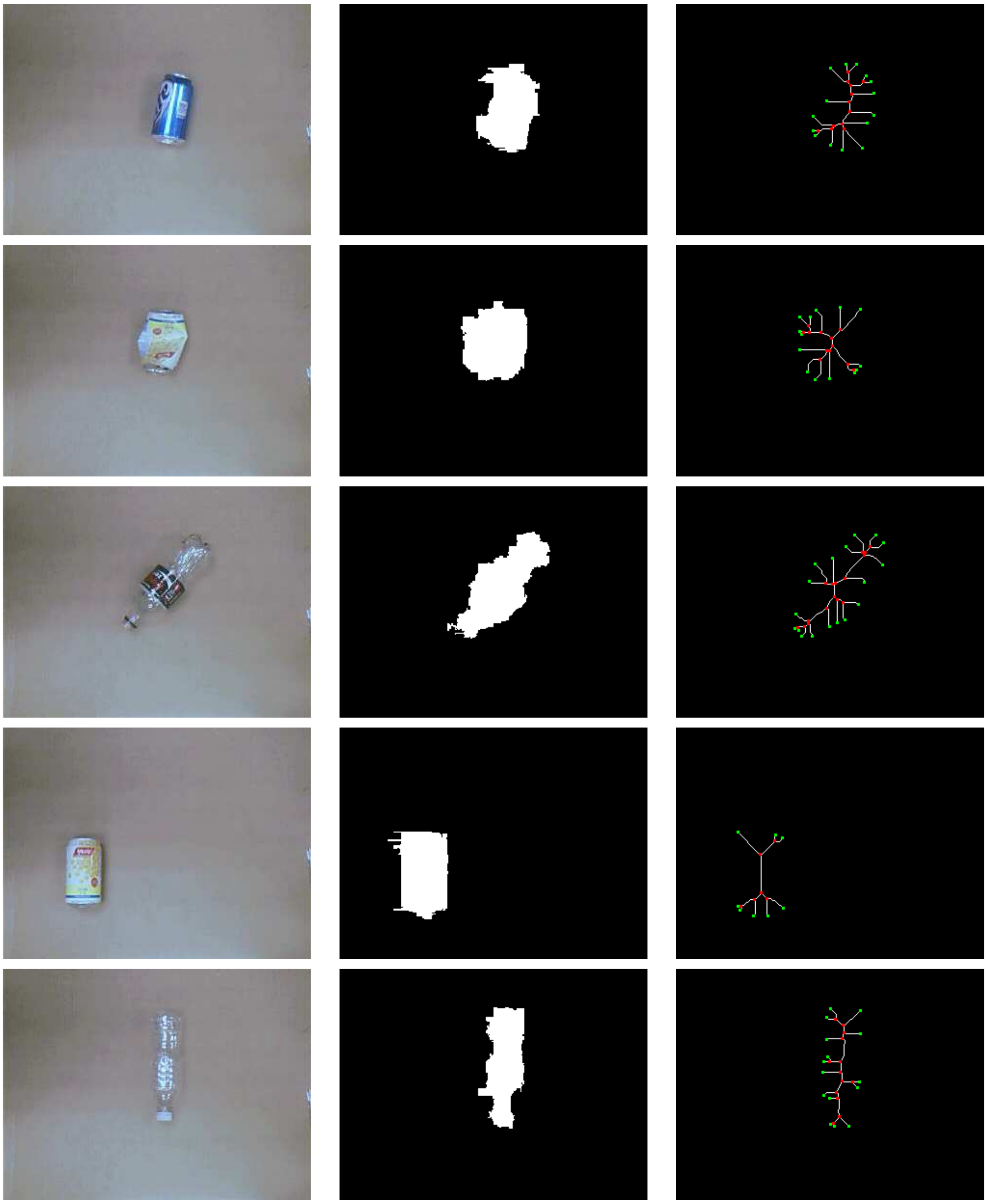
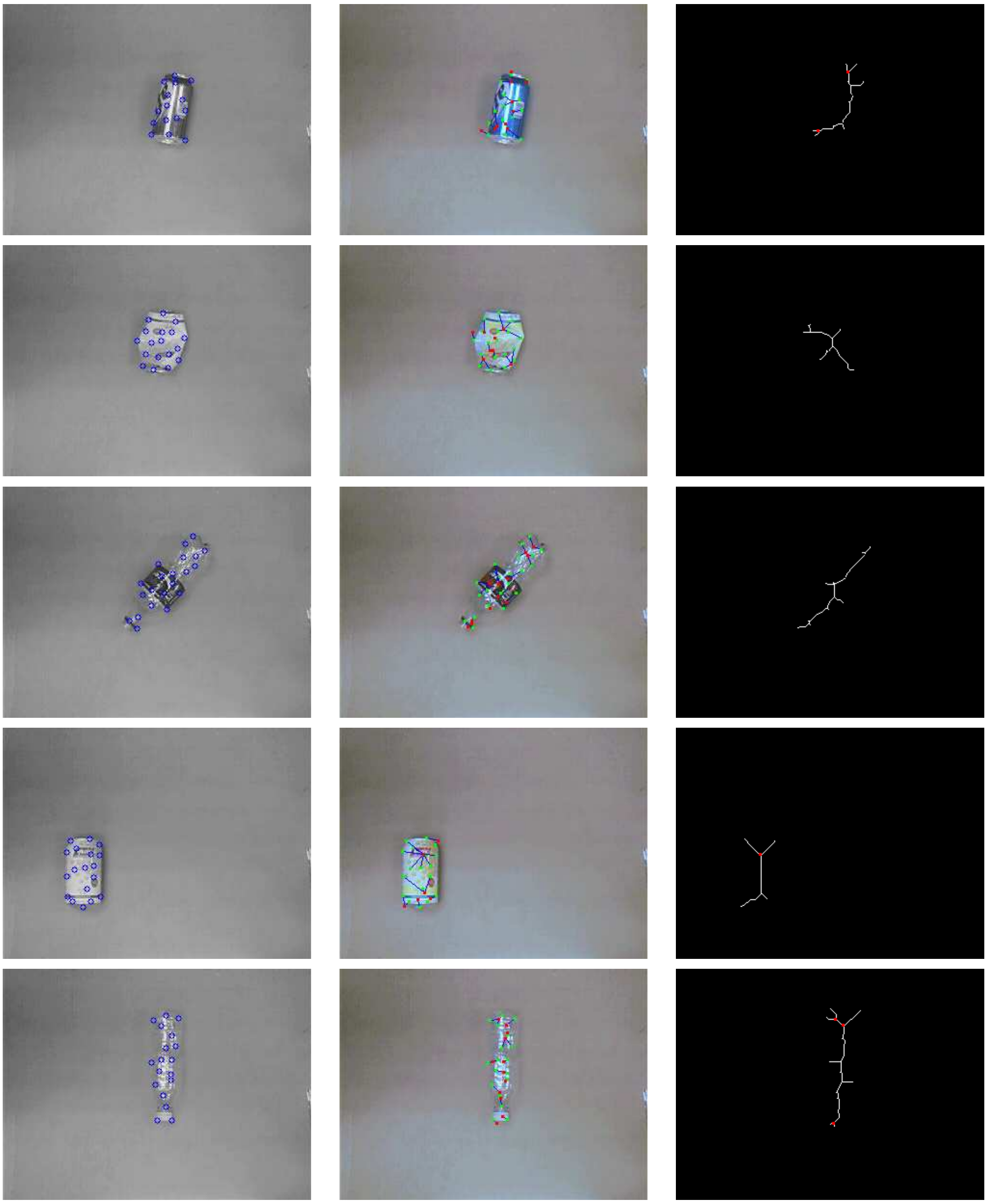


Figure 4.12: Example of a recycling experiment containing a pile of five plastic and metal objects that were individually separated and examined.



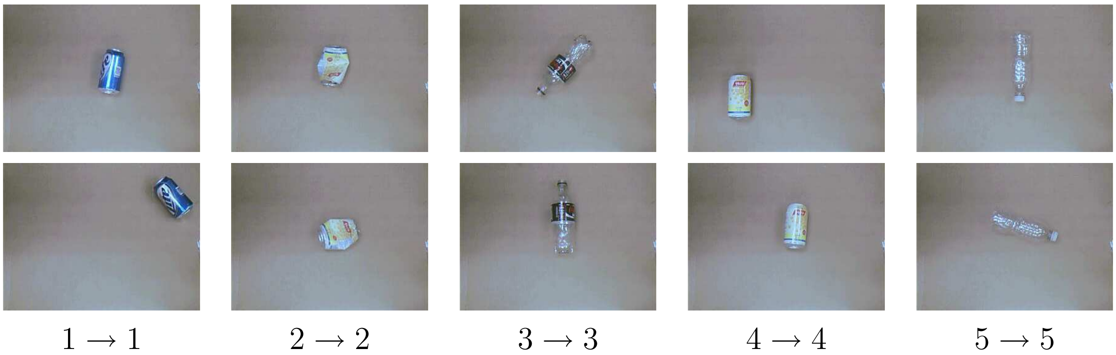
object isolated	object mask	skeleton with points

Figure 4.13: Example of a recycling experiment containing a pile of five plastic and metal objects that were individually separated and examined. From left to right: The image taken after the object has been picked up and separated, the binary mask of the isolated object, and the skeleton with the intersection points and end points labeled. Time flows from top to bottom, showing the progress as each individual object is examined.



mapped points feature points final skeleton

Figure 4.14: Continuing the same experiment from Figure 4.13. From left to right: The feature points gathered from the isolated object, the image after mapping the feature points to the intersections points, and the final skeleton with revolute joints labeled. Time flows from top to bottom, showing the progress as each individual object is examined.



1 → 1 2 → 2 3 → 3 4 → 4 5 → 5

Figure 4.15: Results from matching query images obtained during a second run of the system (top) with database images gathered during the first run (bottom) for the recycling experiment.

4.5 Sorting using Socks and Shoes

Another practical example where the proposed approach would be particularly useful is that of sorting socks in a pile of laundry or organizing your shoes by grouping them with the corresponding pair. This experiment also tests the ability of the system to locate objects within a pile and be able to extract them for sorting like in the previous experiment. We used socks and shoes of different color and size to represent what you may see in a pile of laundry or shoes lying on the floor. The algorithm was able to distinguish between each item for extraction until all of the objects had been removed.

Each of the items were treated equally like in the previous experiment. After the item was extracted and classified, a sorting algorithm could be used to decide whether the other half of the sock or shoe has been located or not. If the other half of the sock or shoe has not been examined previously, then that object is set aside until the other half has been extracted. Note that after the objects have been set aside for individual examination, it is much easier to determine their characteristics for such a matching procedure than when they are cluttered in the entire group. The results of the experiment are shown in Figures 4.16, 4.17, 4.18, and 4.19.

Table 4.3: Evaluating Probabilities of socks and shoes: The rows represent query images and the columns represent database images.

#	1	2	3	4	5
1	**0.87**	0.69	0.26	0.29	0.16
2	0.62	**0.90**	0.24	0.38	0.18
3	0.29	0.25	**0.86**	0.20	0.12
4	0.25	0.26	0.17	**0.93**	0.38
5	0.26	0.24	0.12	**0.99**	0.56

Due to the limitations of our current gripper, the same steps were taken as in the previous experiment where the human manually grasped the object at a location using an “EZ Reacher”. While the human was grasping and moving the object, the algorithm continued to run as if a robot were in the loop. As a result, no modification to the algorithm was made.

The confusion matrix regarding the socks and shoes experiment is in Table 4.3, indicating the probability of each query image matching each database image. All of the cases, except for query image #5, were correctly matched with the corresponding database image. In case #5, the white shoe and white sock were paired to be a match and resulted in an incorrect pairing. If the probability of query image #5 and database image #5 was higher than Th_{min}, then the white sock would have been correctly paired with the other white sock due to the number of revolute joints found through interaction. This experiment is an example of how two different objects can be mistakenly paired together through vision only.

4.6 Comparison with Related Work

In [12], revolute and prismatic joints on a rigid object were categorized by using a similar technique of calculating feature points within a video sequence. To demonstrate that our approach can calculate the same information on a rigid object as well as a non-rigid object,

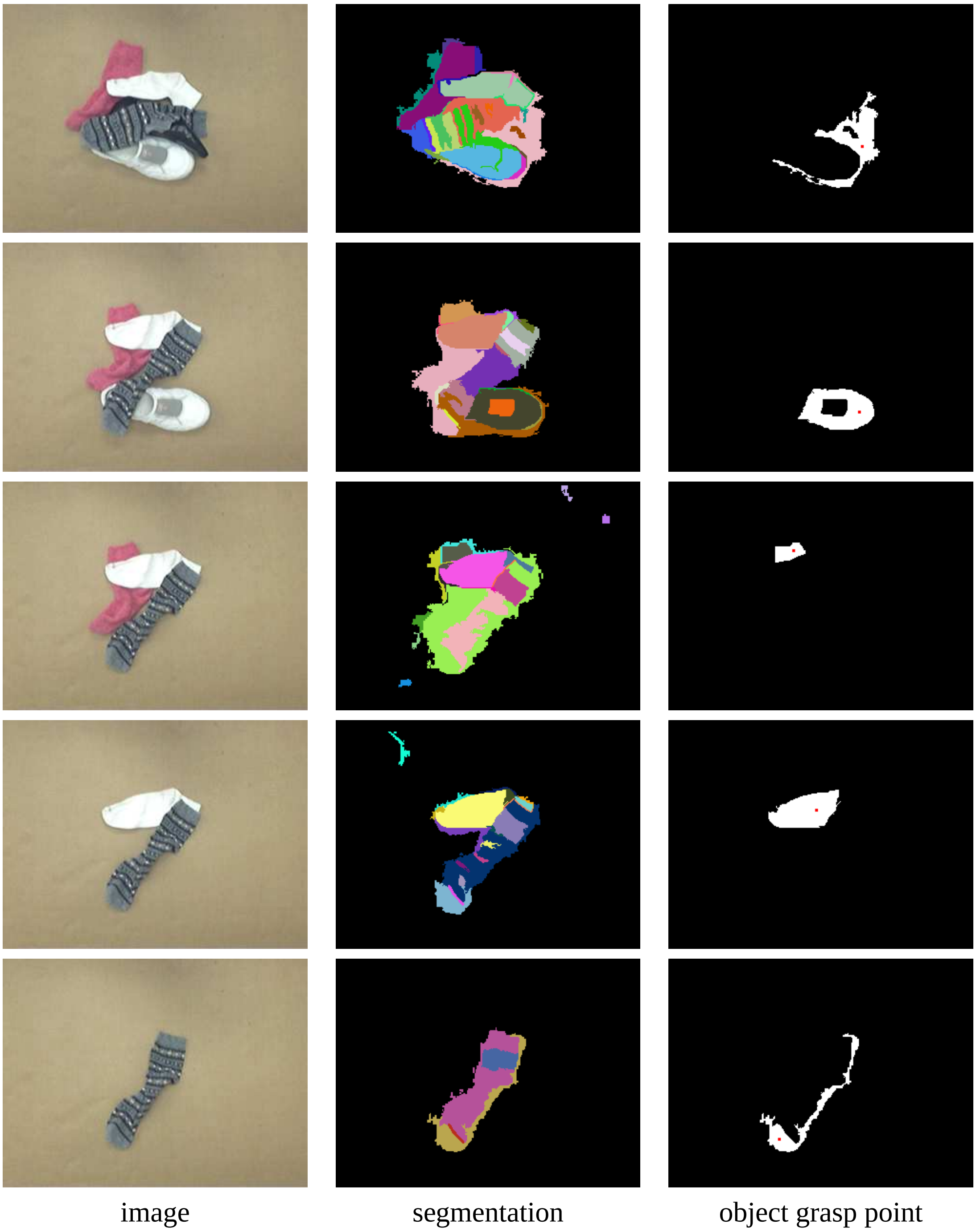


Figure 4.16: Example of a service robot experiment containing a pile of socks and shoes that were individually separated and examined.

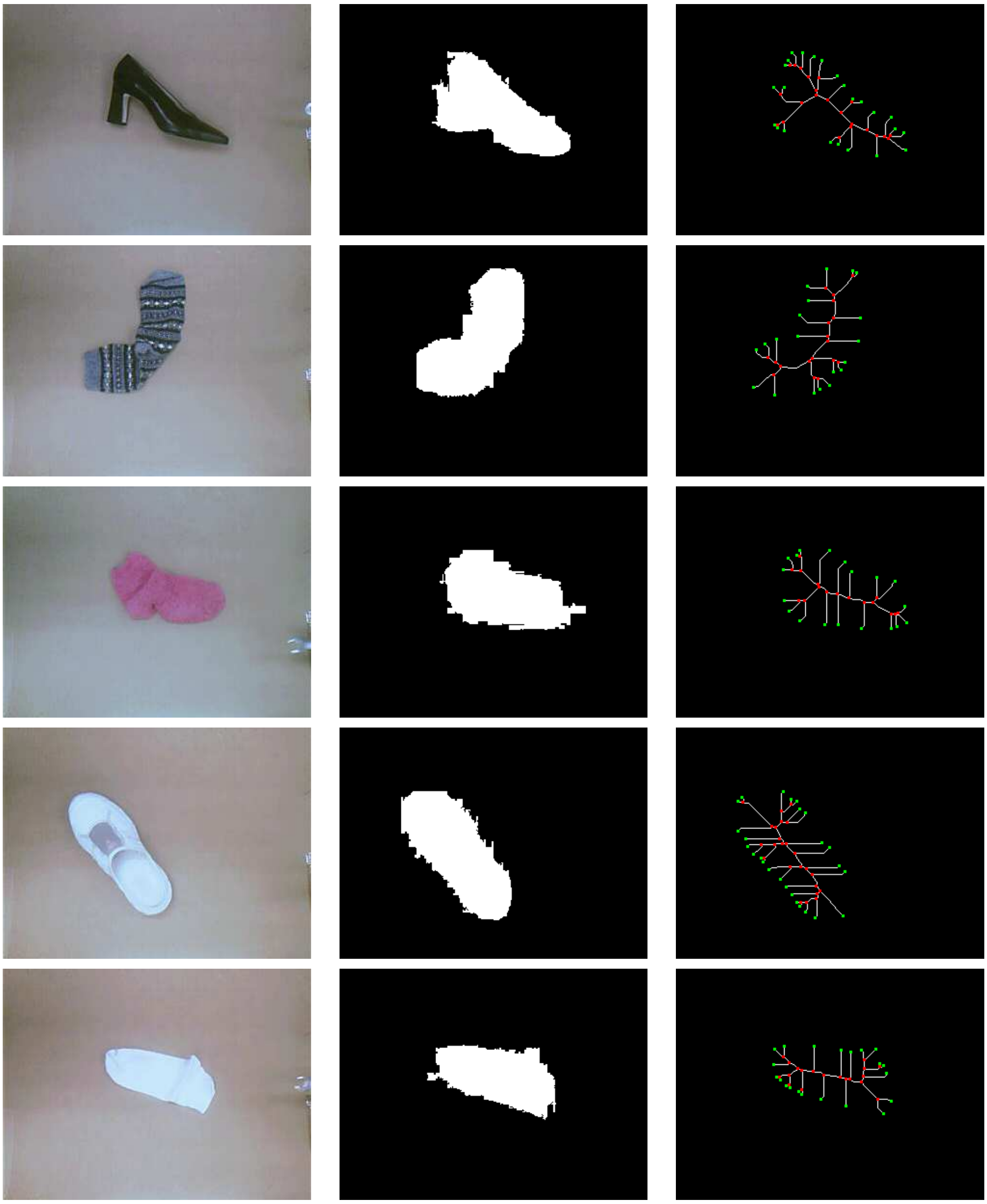
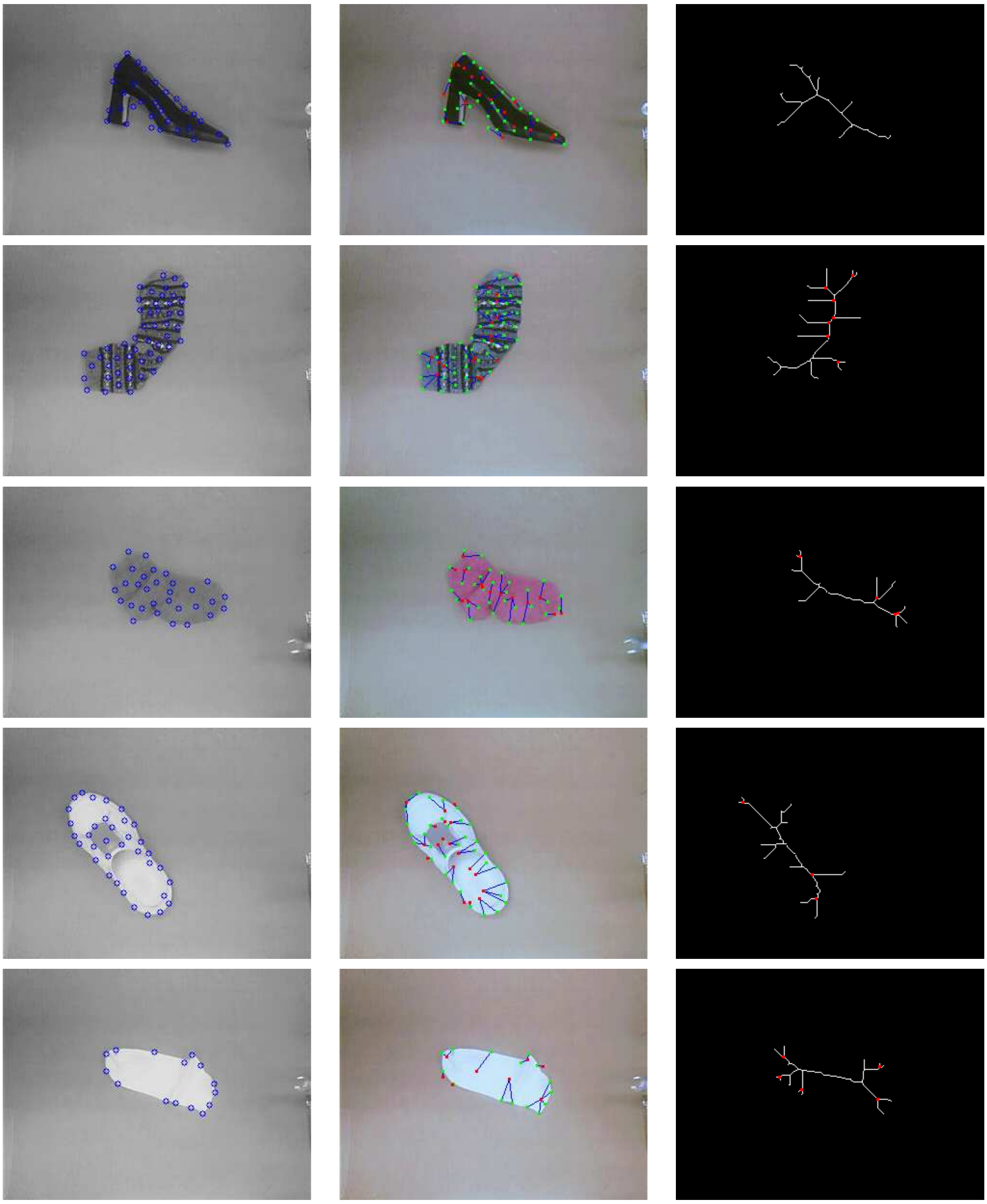
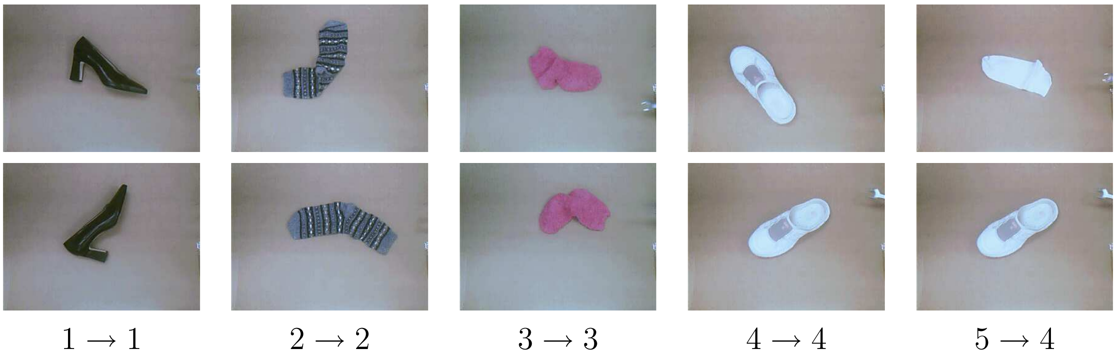


Figure 4.17: Example of a service robot experiment containing a pile of socks and shoes that were individually separated and examined. From left to right: The image taken after the object has been picked up and separated, the binary mask of the isolated object, and the skeleton with the intersection points and end points labeled. Time flows from top to bottom, showing the progress as each individual object is examined.



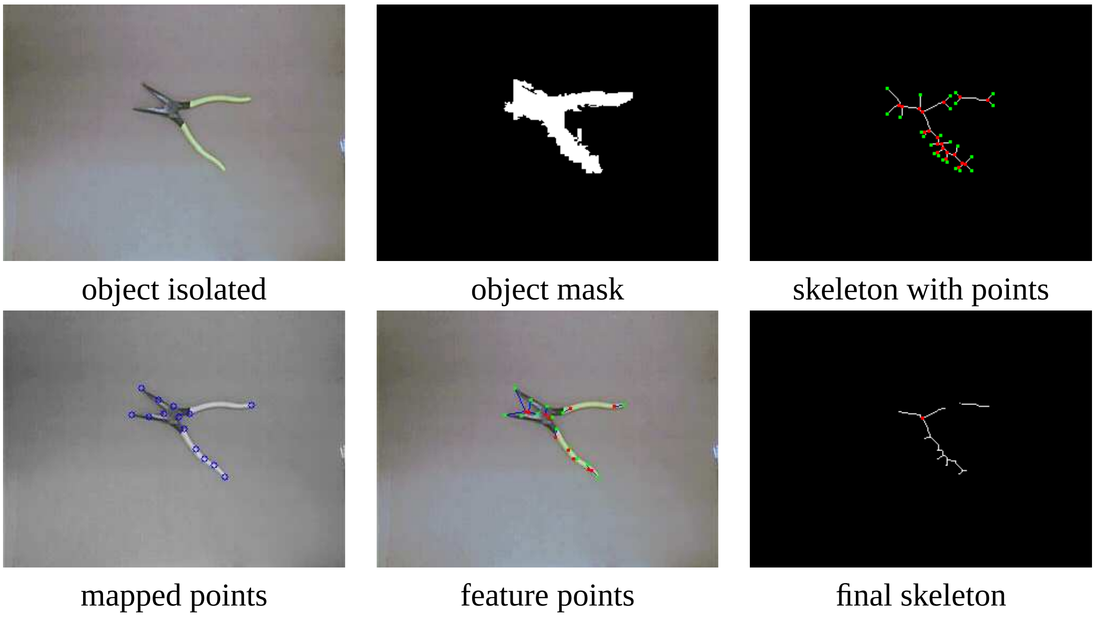
mapped points feature points final skeleton

Figure 4.18: Continuing the same experiment from Figure 4.17. From left to right: The feature points gathered from the isolated object, the image after mapping the feature points to the intersections points, and the final skeleton with revolute joints labeled. Time flows from top to bottom, showing the progress as each individual object is examined.



1 → 1 2 → 2 3 → 3 4 → 4 5 → 4

Figure 4.19: Results from matching query images obtained during a second run of the system (top) with database images gathered during the first run (bottom) for the recycling experiment.



object isolated object mask skeleton with points

mapped points feature points final skeleton

Figure 4.20: Example of comparing our approach to that of related work in [12]. From top left to bottom right: The image taken after the object has been picked up and separated, the binary mask of the isolated object, the skeleton with the intersection points and end points labeled, the feature points gathered from the isolated object, the image after mapping the feature points to the intersections points, and the final skeleton with revolute joints labeled.

Figure 4.20 gives the original and final image along with the steps in achieving the end results for a pair of pliers.

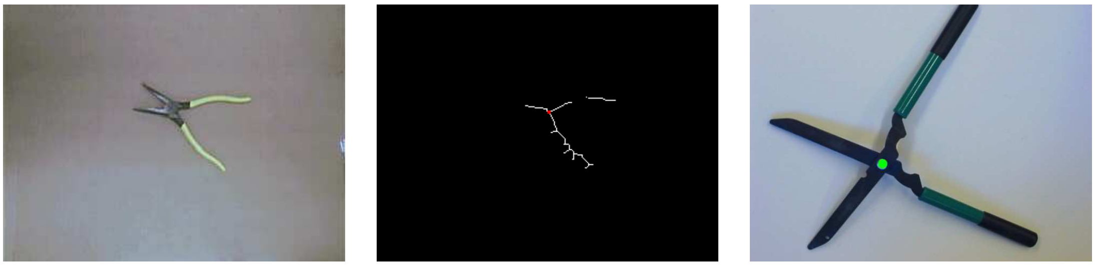


Figure 4.21: Example comparing our approach to that of related work in [12]. LEFT: Original image from our approach. MIDDLE:Results from our approach with the red dot representing the revolute joint. RIGHT: Results from [12] with green dot representing the revolute joint.

Figure 4.21 illustrates an image from [12] and the resulting image from our algorithm to verify that the same amount of information was gathered using both strategies. Our approach extends the work in [12] by also including the ability to handle non-rigid objects.

Chapter 5

Conclusion

We have proposed an approach to interactive perception in which a pile of unknown objects is sifted by an autonomous robot system in order to classify and label each item. The proposed approach has been found to be effective over a wide range of environmental conditions. The algorithm is shown empirically to provide a way to extract items out of a cluttered area one at a time with minimal disturbance to the other objects. This system uses a stereo camera system to identify the closest item in the scene. The information gathered from a single camera is sufficient to segment the various items within the scene and calculate a location for the robotic arm to grasp the next object for extraction. The stereo system is only used to calculate depth.

Monitoring the interaction of the object builds upon the approach in [12] to group different feature points together that share the same characteristics. The work in [12] also determined locations of revolute joints for planar rigid objects. Extending their work, the items that were monitored in our experiments were both rigid and non-rigid. This system demonstrates promising results for extraction and classification for cluttered environments.

The proposed approach only begins to address the challenging problem of interactive perception in cluttered environments. Other avenues can be explored in regards to improving the classification algorithm and learning strategy. When looking for a target item, one must consider the orientation of the object along with the angle from which it is viewed. Additional interaction and labeling techniques could be used to improve the ability of the system to determine which characteristics of an object make it distinguishable from other objects.

Currently, the system is setup to only allow interactions from two directions. Using the third camera as a mode of reference, the robot is able to interact the top part and the left part of the object in the classification images. The right side and bottom part of the object is out of reach from the robotic arm and would occlude the object from the camera's view if it tried to interact the other parts of the object. A solution to this problem would be to have the isolated objects placed on a turntable so that the robot would be able to interact with all directions of the object without occluding any part of the camera's viewing area.

The learning process is considered to be the most important part of any robot system. Software can be created for a robot to look for a single item, but what makes the system better is to learn about the other items for which it is not looking. Instead of the operator having to change or rewrite the software to look for another item, the robot system can learn about every item that it encounters and the operator only needs to reference a single item by one word or phrase to look for another item. The system can learn about similar features for items and group them into specified categories based on how they react to the robot system.

Each item that the system encounters for classification needs to give a lot of information to allow the algorithm to learn about what it does instead of what it looks like, just like using affordance cues [15] [16] [21]. In the case of a shoe, the system can model how it looks, inside and out, and how it reacts to moving around on the table. We can learn that each shoe has a general shape and that it contains an empty area in the center of it. The robot system could go as far as knowing what goes inside of the shoe and know how to classify it even further as something to wear. We know that you do not put your hands in your shoes to wear, but the robot would learn that either a foot or hand could fit inside of this item and group it in the same category as clothing.

Another improvement of the modeling of the object would be to incorporate a 3D model instead of a 2D model. The 3D model would provide a more accurate representation of how each revolute joints moves and give a more detailed skeleton that describes the overall shape of the object. In the case of giving the system a round single colored ball, after viewing and interacting with the ball, the cameras would only see a circle that doesn't roll, in a 2D world. The system would disregard information vital to discovering the dynamics of each object if the object did something in the 3D world and looks like another in the 2D world, just like the ball scenario. We believe these are fruitful areas for future research.

In the initial discussions for this approach, the idea of designing a robotic system to autonomously navigate though a cave or dark enclosed area was mentioned. The main purpose would be to create a 3D map or model of the area and everything within it. But to accomplish this task, we would need to know how to learn about what was inside of the cave or area and how to move it if it was obstructing the robot's view or in the robot's path. This idea guided us to the work of learning about the environment and how to interact with it, whether the objects are in a cave, a recycling bin, a clothes hamper, or on the floor in your house.

Appendices

Appendix A

Software Pseudocode for Vision System

```
Initialize serial port Capture IMG1 and IMG2 from stereo camera

While (TRUE)

{

      Copy IMG1 to tempIMG

      Calculate depth from IMG1 and IMG2

      Segment IMG1 into different regions using graph-based segmentation

      If (No regions) %No more objects in pile

        Quit;

      Calculate average depth within each region

      Choose region with largest depth to be object on top

      Calculate centroid of chosen region to be grasp point

      Send (X,Y,Z) coordinates to robot for extraction

      Use tempIMG and current image from camera to determine if object was extracted

      While (Object not extracted)

      {

            Set ”e” to a positive value
      Send (X,Y,Z + e) coordinates to robot for another attempt
      If ((Z + e) $\geq$ (Table top coordinates))
        Quit;
}
If (Object extracted) %Classify object
{
  Send coordinates of classification area to robot
  Capture current image of object from 3rd camera
  Calculate binary mask of object
  Compute color histogram of object
  Use binary mask to make skeleton of object
  Locate intersection points and end points on skeleton
  Find feature points on object
  Send coordinates of first end point to robot
  While (Tracking feature points)
  {
        Send coordinates of next end point to robot
        Monitor feature points as robot interacts with object
        If (# of frames is a multiple of $f_{length}$)
        {
          Find groups of feature points that moved within the last $f_{length}$ frames
          Calculate ellipse surrounding each group
          Determine end points of major axis of each ellipse to be either
               end or joint of group
          Locate intersection point closest to joint of each group and
```

```
                label it as revolute
            }
        }
        Classify object with color histogram and # of revolute joints
    }
}
```

Appendix B

Software Pseudocode for Robot System

```
Initialize robotic arm to home position
Initialize serial port
While (TRUE)
{
      Read data off of serial port
      If (data == ”Quit”)
        Quit;
      Convert data into (X,Y,Z) coordinates
      Compute inverse kinematics of coordinates
      Determine if coordinates are for extraction or classification
      If (extraction)
      {
        Move robotic arm to hover over position
        Open gripper
        Lower robotic arm to actual position
        Close gripper
```

```
            Move robotic arm to home position
            Write data to serial port of completion
        }
        Else if(classification)
        {
            Move robotic arm to position
            Move robotic arm forward 2 inches
            Move robotic arm back 2 inches
            Write data to serial port of completion
        }
}
```

Bibliography

[1] G. Bertrand. A parallel thinning algorithm for medial surfaces. *Pattern Recognition Letters*, 16(9):979–986, 1995.

[2] A. Bicchi. Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *IEEE Transactions on Robotics and Automation*, 16(6):652–662, 2000.

[3] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986.

[4] F. Chaumette and S. Hutchinson. Visual servoing and visual tracking. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 563–584. Springer, 2008.

[5] T. Christensen and M. N ø rgaard. Learning the dynamic properties of an unknown object through interactions with an autonomous mobile robot. In *Institute of Electronic Systems, Aalborg University, Denmark*, 2000.

[6] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[7] P. Fitzpatrick. First contact: an active vision approach to segmentation. In *International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[8] P. Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 1292–1298, 1991.

[9] P. Gibbons, P. Culverhouse, and G. Bugmann. Visual identification of grasp locations on clothing for a personal robot. In *Towards Autonomous Robotic Systems (TAROS)*, pages 78–81, August 2009.

[10] B. Hannaford and A. Okamura. Haptics. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 719–740. Springer, 2008.

[11] I. Jolliffe. *Principal Component Analysis*. 1986.

[12] D. Katz and O. Brock. Manipulating articulated objects with interactive perception. In *Proceedings of the International Conference on Robotics and Automation*, pages 272–277, May 2008.

[13] J. Kenney, T. Buckley, and O. Brock. Interactive segmentation for manipulation in unstructured environments. In *International Conference on Robotics and Automation (ICRA)*, pages 1377–1382, 2009.

[14] D. Kragic, M. Björkman, H. Christensen, and J. Eklundh. Vision for robotic object manipulation in domestic settings. *Robotics and Autonomous Systems*, 52(1):85–100, July 2005.

[15] V. Krunic, G. Salvi, A. Bernardino, L. Montesano, and J. Santos-Victor. Affordance based word-to-meaning association. In *International Conference on Robotics and Automation (ICRA)*, pages 4138–4143, 2009.

[16] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2007.

[17] F. Osawa, H. Seki, and Y. Kamiya. Clothes folding task by tool-using robot. *Journal of Robotics and Mechatronics*, 18(5):618–625, 2006.

[18] K. Salleh, H. Seki, Y. Kamiya, and M. Hikizu. Inchworm robot grippers for clothes manipulation. *Artificial Life and Robotics*, 12(1–2):142–147, 2008.

[19] A. Saxena, J. Driemeyer, and A. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27:157–173, February 2008.

[20] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002.

[21] D. Skočaj, G. Berginc, B. Ridge, A. Štimec, M. Jogan, O. Vanek, A. Leonardis, M. Hutter, and N. Hawes. A system for continuous learning of visual concepts. In *International Conference on Computer Vision Systems*, 2007.

[22] M. Stark, P. Lies, M. Zillich, J.Wyatt, and B. Schiele. Functional object class detection based on learned affordance cues. In *International Conference on Computer Vision Systems (ICVS)*, pages 435–444, May 2008.

[23] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[24] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[25] I. Walker. A successful multifingered hand design — The case of the raccoon. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 186–193, August 1995.

[26] B. Willimon, S. Birchfield, and I. Walker. Interactive perception for cluttered environments (under review). In *International Conference on Robotics and Automation (ICRA)*, 2010.