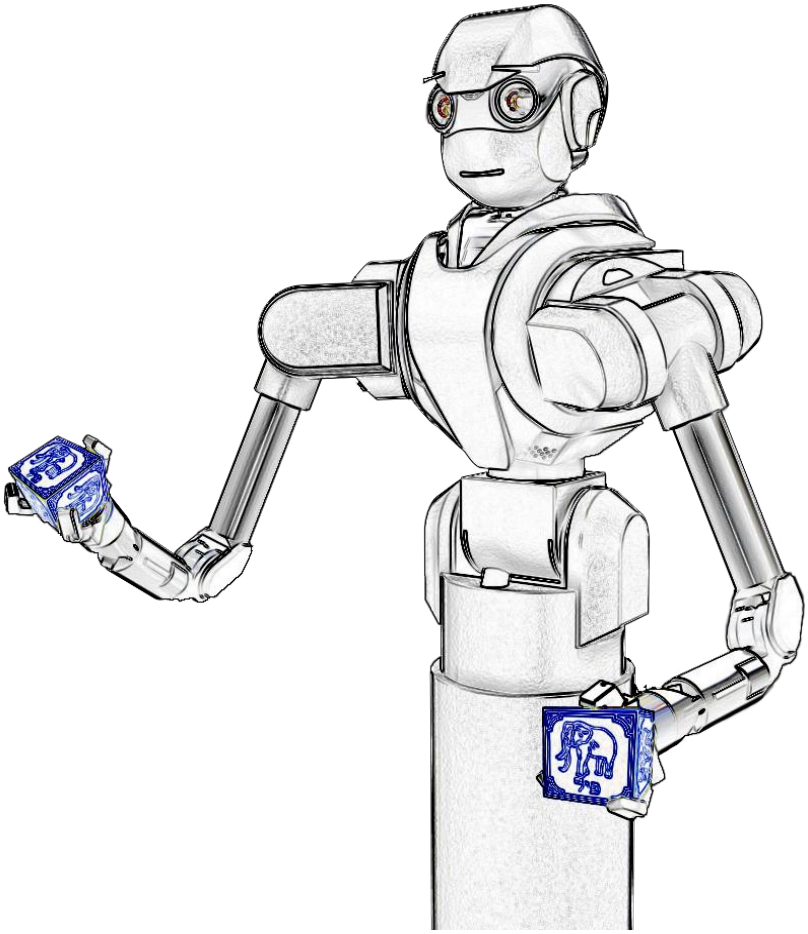
Developmentally Learning the Support Affordance of a Platform



Karl Deakyne – Yehoshua Meyter – Brian Russell

CpreE 585x

TABLE of CONTENTS

List of Figures

List of Tables

Abstract

This paper describes an approach to teach a robot the support affordance of a platform. This is an important concept for robots to learn and one that humans gain knowledge of at an early age. For this goal, a robot performed a sequence of exploratory behaviors focusing on the edge of a platform. Four of these behaviors included pushing an object past the boundary of a platform edge. From these behaviors, the robot's arm was tracked along with the object it was pushing. Using this tracking, it became possible to find the point at which the robot's arm disassociated from the object it was pushing. Machine learning algorithms were then used to classify objects as supported or not supported based on this point of disassociation. Results are drawn from this classification and the concept of self-detection is explored as another avenue for this research.

Introduction

Support is a fundamental concept that mankind relies on to complete a variety of rudimentary tasks, such as judging how and where to place objects on a platform. In most cases, humans can solve this task with minimal effort because they've learned the fundamental concepts of support at a young age. At six and half months, an age where most children are not even capable of crawling on their bellies, infants have already developed a sense of whether or not an object should fall when it is placed on the edge of a platform [1]. Though this basic concept is understood by the simplest of human minds, it escapes the "minds" of complex machines. Modern robots possess sophisticated hardware, but their lack of sophisticated software prevents them from autonomously learning the notion of support.

This research works towards teaching a robot to solve support problems by having it build its own infantile intuition of support. The robot generates this intuition by exploring its surroundings, a methodology borrowed from developmental psychology. This field provides evidence that children learn using self-generated rules to build models of their environment [2]. Validation or violation of these rules will change a child's exploratory behaviors, and lead to new rules for an improved understanding of the world [3]. By having the robot push an object around an edge, it can develop an expectation of where the object is supported. Repeated trials with a variety of objects and edges have developed these expectations and given the robot a prediction of support.

An understanding of support is important for any intelligent being, whether it is a human or a machine. It can keep towers upright, determine if a potentially dangerous object will fall over, describe if an unattended object will remain where it was last seen, and even identify if an object can support an intelligent being. In the following sections of this paper we will describe more applications of this project, look at related work in the fields of artificial intelligence and developmental psychology, and further explain the approach taken to solve this problem.

Proposed Applications

A robot with the ability to explore an object and determine whether or not it can support another object can then use this knowledge to begin exploring the support of a tower of objects. To keep the tower upright, each piece needs to be supported not only by the object directly below it, but also every other object in the tower. For each new object, the robot would develop new boundaries of support to predict where to place the next object. In this manner, the robot could also develop an intuition of what orientations would produce a stable configuration.

There are a large variety of common tasks that involve the notion of support, including placing objects on tables, carrying objects, and walking. Preprogramming a robot to do every one of these tasks is infeasible, as the program would have to take into account all the tasks, all the possible objects the robot would interact with during the tasks and every environment that the task could be performed in. Preprogramming a robot to do some of the tasks in a specific environment with a collection of objects is also undesirable, as it limits the usability of the robot. In order to perform each one of these common human tasks, a robot needs the ability to learn the relevant support affordances for each problem.

Related Work

For this research, a platform is considered an object capable of supporting another object. As infants, humans learn about platforms from watching their parents place things on tables, counters, and desks. As they age, infants are presented with toys and are allowed to explore placing these toys in a variety of places. During this process we learn to avoid placing things on the edge of a support platform. This is an important idea to understand as it helps infants avoid crawling off a cliff such as a staircase. In studies conducted by Walk and Gibson, infants were presented with a visual cliff, an apparatus that looks like a cliff's edge but is actually a traversable platform. In these experiments, most children were able to detect and avoid the visual cliff [4]. However, in some further studies around 66% of infants would cross to the other side of the transparent platform [5].

Some animals don't have the same issues in detecting and avoiding the "visual cliff". Gibson conducted studies with the young of many species to determine if visual cliff avoidance is innately present or if it is learned [6]. The study found that animals determine the presence of a visual cliff by gravitating toward the shallow side of a visual cliff apparatus, showing an innate ability to detect depth. In fact, 100% of goats tested within a few hours of their birth avoided the visual cliff altogether. The differences between the goats and human trials suggested that humans are not born with an ability to detect depth; instead they have to learn the meaning of it through locomotor experience [5]. In order to gain this experience, infants use models of vicarious learning and then apply it towards their own experiential play with toys. Through this interaction they learn which objects can and cannot support something as well as what parts of the object will provide support.

Exploration is a concept that is fundamental to human learning. By examining the surrounding world, an infant begins to develop assumptions about both object and self-support. At a certain age, infants begin to discover the point at which they can support themselves [7]. However, before supporting themselves, infants begin to develop some ideas about object support [1]. Young children explore the surface of a table to determine how far a block can be pushed until it falls. This is a type of balance test that children generate to build a model of rules about surfaces. When a toy block falls off of the table it changes the child's beliefs about that surface.

This leads the child to change their exploratory behaviors [2]. While pushing a block towards the edge of a platform, a child constantly creates new rules for that platform that are based on the changes that occur during their interactions.

These rules that children create and redefine begin to develop their support affordances for objects and surfaces. Affordances are relationships between one thing and another, and in particular they are what one thing can provide another [9]. E. Gibson expanded on this idea and linked exploratory actions to learning affordances [10]. The children described above are beginning to develop their knowledge basis of support through exploration and this is the same method the robot used in this project to develop it's own affordances.

Affordances for robotics have been explored before in various applications. The understanding of the relationship between objects, actions, and reactions has been used to perform "task interpretation and planning capabilities" and perform simple imitation games [11]. Also, a robotic vehicle at the Middle East Technical University has used affordances to distinguish between objects that can be moved by the vehicle and those that cannot. The only objects avoided are those that are considered non-traversable, thus a more intelligent obstacle avoidance model is created through the use of affordances [8]. Support is a novel application for robotic affordances and this is the first paper to explore this application.

Problems associated with support, however, have been explored many times before. In one such case known as Blocks World Planning, AI researches attempt to minimize the number of steps necessary to rearrange a stack of blocks [12][13][14]. There are many solutions to this problem, but in all of the proposed solutions one important detail is left out. Not one of these papers explains how robots will perform the stacking action itself.

In order for a robot to determine that an object is capable of being supported, it has to learn what part of a platform can offer support. This paper explores affordances as a novel approach to this support problem. In a study dealing with the robot being capable of detecting itself, it has been shown that with an efferent-afferent delay system it is possible for a robot to determine that an object in its field of view is part of self, as long as it follows the same movement patterns. While moving an object, the robot can determine that the object is a part of self [15]. If the object is no longer moving in tandem with the robot, it can then be classified as not part of self. Using techniques that allow the robot to record where it lost control of the object, our project has created a model for predicting regions of support.

Experimental Setup

Robot Platform

The robotic platform in the Developmental Robotics Lab at Iowa State University was used for this research. It consists of two Barrett Whole Arm Manipulators and two different Barrett Hands. One hand is equipped with tactile pressure sensors and the other is capable of vibrotactile sensory feedback with an external sensor. The robot is also equipped with two Logitech QuickCam Pro 4000 webcams for vision and two Audio-Technica U853AW cardioids microphones for auditory input. For this particular research, a Zcam 3D camera was used in order to gain the perception of depth. Because a change in depth is a defining characteristic of a platform, this camera adds an import sensing modality to this platform.

Data Collection

For these experiments, data was collected for 390 trials and was processed offline. The modalities of vision, proprioception, and tactition were recorded with the following parameters. The color vision used the native webcam resolution of 640x480 pixels and with a lowered frame rate of 10 fps. The Zcam's images were recorded in 320x240 pixel grayscale at roughly 21 FPS. Proprioceptive data was sampled directly from the Barrett WAM's motor controller at a rate of 500 Hz. Tactile data was acquired at a rate of 10 Hz.

Objects

The objects used in this project have been constructed from pieces of wood and are composed of three distinct shapes: a circle, a square and a triangle. These objects were chosen because they present variation in shape while still maintaining a manageable data set. Unlike many household objects, these constructed blocks have a definite form and a uniform material density. While this may appear as oversimplification, these objects are not unlike various children's toys, including building blocks and shape sorters.

In order to track these objects easily, they were colored bright green. Such coloring made these objects easily separable from the surrounding area. Because the focus of the project was centered on platform edge exploration, this was deemed to be an acceptable modification of the objects.

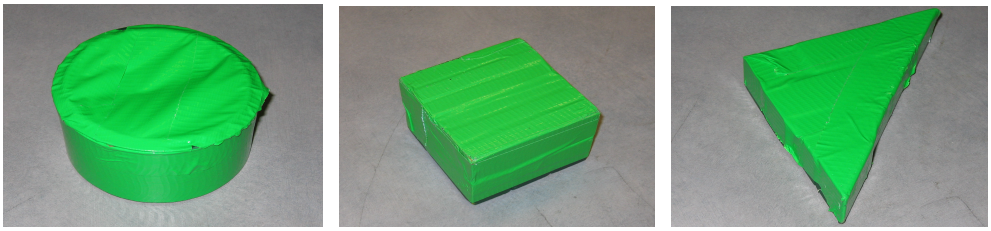


Figure 1: Objects used for Experiment

Constructed Platform

Initially, the proposed experiments required the construction of an independent platform. However, this quickly proved to be infeasible due to the limitations of the robot's end-effector space. Additionally, a substantial slope and platform length was necessary in order to track the objects in vision. Therefore, it was decided that an additional platform would not be constructed and that a different setup would be created instead.

The modified setup consisted of a standard office table and a removable wooden board. The board was hinged so that it could be adjusted to fit a variety of configurations. Furthermore, the board chosen was pre-finished with a low coefficient of friction. Such a surface proved to be ideal for tracking sliding objects. Its ease of adjustment also made it extremely versatile through which a variety of experiments could be performed.

In total, the removable board was used in three different configurations and completely removed for the remaining two configurations. Each configuration gave the robot a different perception of the platform itself. In configurations where the platform ramp was completely removed, the robot was essentially exploring a cliff-like environment. This sensation was explored by the robot's fingertip and using an object. Both explorations are discussed in the methodology section.

In configurations where the board was attached to the table, the robot explored three unique edge properties. In the first position, the platform was extended horizontally and the robot pushed objects onto it. Because the objects did not separate from the hand, this gave the robot an expectation that it could control the object it was pushing. The expectation was subsequently violated when the board was put in an inclined position. In this position, the ramp had an approximate slope of -1 and formed an interior angle with the table of approximately 135 degrees. In this position, objects did separate from the hand and thus violated the robot's expectation that it had complete control of an object.

The final configuration of the board was unique in that the edge was perceived vertically. Clamped vertically to the table, the platform served as a bounding wall for the platform. With this type of edge, objects can be pushed to the limits of the supporting platform but cannot be pushed past those limits. The intent of this configuration was to explore and determine the edges of the platform using the joint torque values of the robot's arm.

Basic Manipulation Strategy

In each trial of the data set, the robot moved its hand horizontally across the table until it reached a predetermined stop position or encountered a torque limit. The data set included ten trials for each object, encountering each surface, in three different starting positions along the edge of the table, creating 360 trials for the object training. Likewise, the robot performed an additional 30 trials where it used only its fingertip to explore the edge of a platform. In the following section, these behaviors are explained in greater detail.

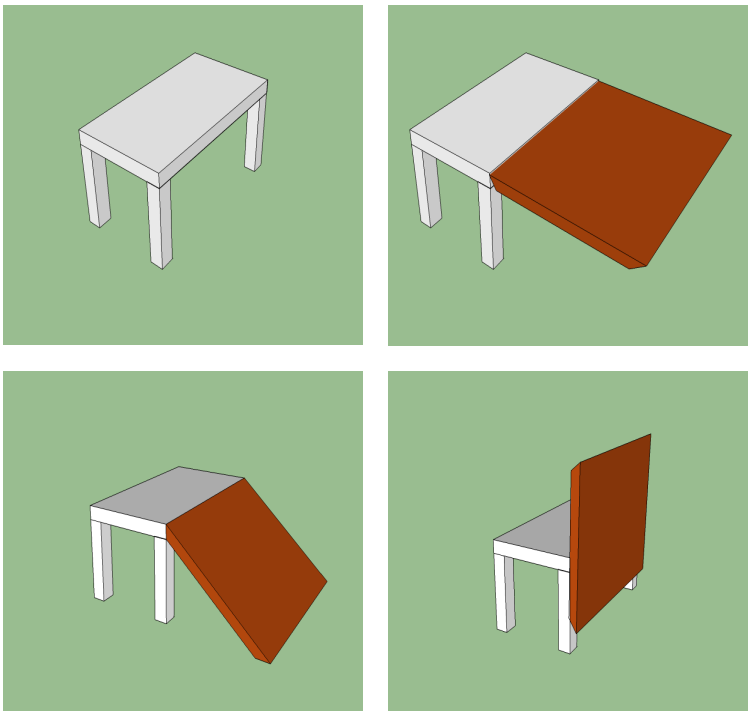


Figure 2: Four orientations of the platform and ramp. From the top left, no ramp, horizontal ramp, slanted ramp, and upright ramp.

Methodology

In this project, the edge of a platform was explored through several different manipulation techniques and sensory modalities. The boundaries of a platform can occur in several different forms. For instance, the edge of a platform could be a vertical drop off, a slanted incline, a vertical wall, or unchanging in height. For that reason, these exploration behaviors were created in order to explore these different properties of platform edges. These behaviors are detailed in the following section.

Exploratory Sliding with Self

This behavior allowed the robot to determine the location of the edge through embodied exploration. In order to accomplish this task, the robot performed a sliding motion in which its finger maintained contact with the table. This slide moved from one of three initial starting locations to a corresponding destination point. The destination point in each trial was located beyond the boundaries of the platform. Therefore, it was anticipated that the vibrotactile sensor data could be used to detect the edge of the platform. For this sliding behavior, ten trials were collected for each set of terminal coordinates. In total, thirty trials were collected for this behavior.

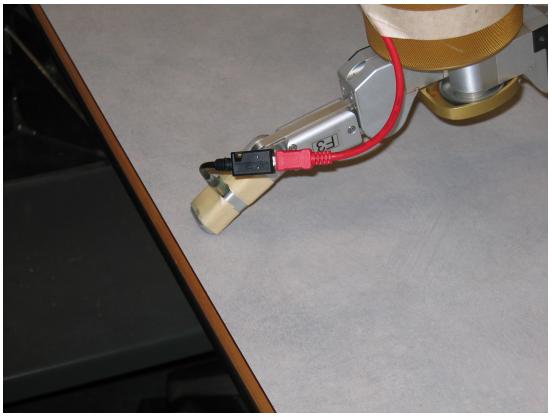


Figure 3: Robot finger with vibrotactile sensor exploring the table edge.

Exploratory Sliding with Objects

As previously discussed, humans understand the affordances of an edge through learning with the aid of objects. In a similar manner, the robot was programmed to use objects in order to explore the test platform's edge. For the first set of experiments, the platform ramp was oriented horizontally and served as an extension of the platform itself. The reasoning for this was that the robot would begin to build an assumption that objects are under its control during a pushing behavior. With this orientation, the robot explored the entire width of the ramp by using three beginning and ending positions that covered the entire board.

For the second experiment, the platform ramp was setup to form an incline against the table. With this configuration, the robot pushed objects towards the table's edge and onto the ramp. Once on the ramp, the object began to disassociate from the hand as it's speed increased. Such a behavior is intended to mimic a person knocking objects off of a table or platform. Once again, the entire board was explored using the same terminal positions tested in the previously described ramp configuration. Modalities of depth and vision were captured for both of these experiments in addition to proprioceptive and tactile data. Each object was pushed for ten trials at each position in order to generate a significant number of trials to build a learning model.

The third experimental setup required the board to be oriented vertically. Instead of falling past the limits of the platform, the objects' movements were instead constrained by the edge. Thus the joint torques became the primary modality of interest for this behavior. For the trials, the arm was given an initial starting position and a goal position. In order to press against the upright ramp, the goal position given was approximately three inches past the edge of the table. Therefore, the goal was impossible to meet but allowed for the arm to press firmly against the upright board. As objects were pushed against the upright ramp, they met resistance until a joint torque limit was met. Once met, the trial ended and the arm no longer exerted force on the board.

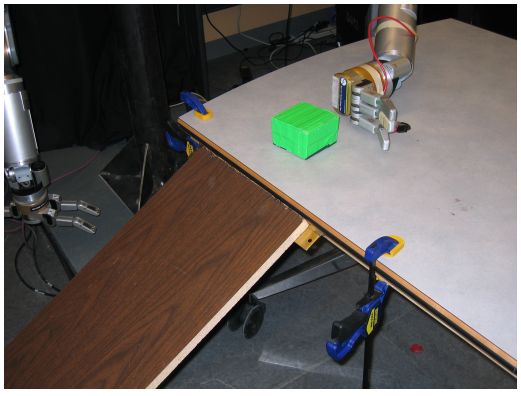


Figure 4: Platorm with slanted ramp configuration.

The fourth experimental setup did not use the platform ramp and instead performed experiments which an object falling to the ground. Similar to the previous three procedures, the robot's arm pushed objects from an origin position to a goal position. Since the goal position was set past the edge of the platform, the hand continued to move while the objects fell to the floor. The behavior used terminal points at locations where the removable ramp had been placed. Therefore, this pushing behavior is identical to the previous pushing behavior with the exception of the platform ramp being removed.

Processing and Analyzing the Collected Data

The project's goal of discovering various properties and affordances of a platform edge necessitated the use of object tracking. As previously stated, the three objects used in these experiments were colored before data collection began in order to track them using computer vision software.

As stated in the project proposal, the OpenCV computer vision library was used as a basis for developing a color tracker. Although OpenCV does not have a standard color tracker, it provides a strong foundation on which to build a custom color tracker. For this project, the color tracked objects were distinct from the surrounding environment. Therefore, the color tracker was tuned to recognize specific RGB values as follows.

The first stage of the color tracker separates the image into three RGB channel images. Each channel image was then filtered to show pixels within a specific RGB range. As these pixels were filtered, they were set to the maximum intensity value if they were found within a given range. With each color channel containing only pixels within a specific range, the three channels could be merged back into a single image. The newly combined image was constructed from pixels present at a location in all three RGB channels. Therefore, only objects within a specific color range appear in the final recombined image.

With this filtered image, OpenCV has built in functions to detect contours in an image. Since the filtered, recombined image contained only the object of interest, this function needed only to detect a single, adequately sized contour. With this contour detected, a rectangular bounding box was drawn over the resulting contour and superimposed onto the original image for visual verification. In this manner it was possible to track both the robot's gold colored cuff and the associated object that it was pushing.

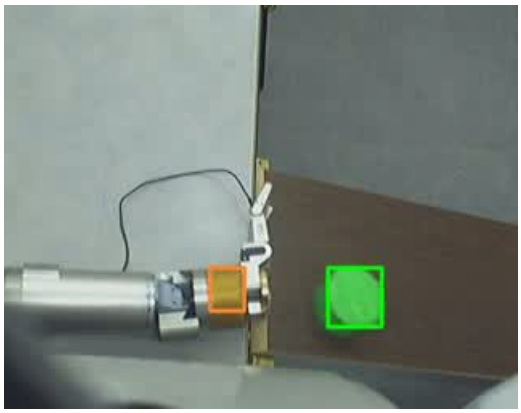


Figure 5: Color tracking of robot arm and cylinder object.

Tracking Self-Exploration

In order to detect the edge of the platform using only the fingertip, both vision and vibrotactile sensing data were analyzed. Because the robot's finger experienced a jolt as it fell off of the platform, the accelerometer data values could be analyzed for rapid changes. When viewing a scatter plot of the accelerometer data over time, this rapid change is values was quite apparent. A sample scatter plot of this data is shown below as an illustration of this behavior.

Once it became apparent that the vibrotactile sensor gave a strong indication of the edge, the readings from the vibrotactile sensor were analyzed to find points where the x,y,z coordinate values substantially deviated from their normal values. Once these values were detected, that particular timestamp was recorded for future mapping back to visual space. In order to map the observed vibrotactile events back into visual space, the arm was visually tracked. Therefore, the cuff of the arm was tracked and its coordinates were recorded for each time step.

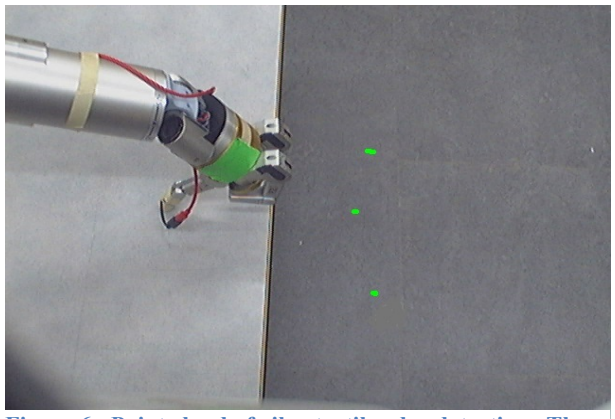


Figure 6: Point cloud of vibrotactile edge detection. The points indicate the location of the robot's wrist when the edge was detected. There are 30 points in this image but they are clustered tightly together.

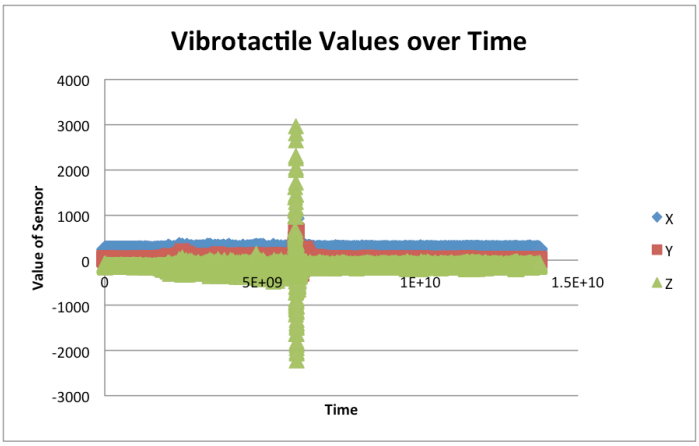


Figure 7: Scatterplot showing vibrotactile measurements over time (for a particular trial). Note the large value change where the edge occurs.

To show the effective of vibrotactile changes visually, a point cloud was generated from the processed vibrotactile data. Points from the point cloud were generated using the following method. If the accelerometer values substantially deviated from their normal values, a point was generated and placed at the location of the robot's wrist for that time. The point cloud below shows how the robot consistently experienced sharp changes in its vibrotactile sense when arriving at the edge of the platform. One will notice; however, that the points are shifted away from the platform. This occurs because the points represent the location of the wrist at that time and not the finger experiencing the vibrotactile event. While tracking the finger would have been a preferable alternative, this research aimed to maintain consistency when tracking the arm. Because the wrist was tracked (and not the finger) for all other behaviors, it was decided that the wrist would be tracked for this behavior as well.

Tracking the Object

Once visually tracked, the objects' and arm's movements needed to be numerically extracted in order to perform an analysis of their behavior. Since bounding boxes were created for the tracked object and the arm, it was possible to simply track the center of each bounding box as it changed position over time. Therefore, files of the object and hand coordinates for a given trial were created. In each coordinate file, the object's position in visual space and the hand's position in visual space were recorded. With this information, it became possible to determine each object's speed for a given frame. Using this speed, a decision could be made whether the hand and object were moving at differing rates. Subsequently, it could be inferred that differing speeds indicate a loss of control over an object.

With the assumption that differing speeds indicate a loss of control over the object, it became necessary to visualize the results of this implication. For that reason, point clouds were generated to show the first occurrence of disassociation between the object and robot's arm. Points were placed at the threshold where the object moved faster than the hand pushing it. These points mapped to visual space cluster around the edge at each of the three pushing trajectories. The figures below show this mapping.

Similar to the slanted ramp setup, the object and hand were tracked visually when the platform ramp was oriented horizontally. After this tracking, the object and hand coordinates were extracted and processed to look for a disassociation. Predictably, no disassociation was found because the object never moved faster than the hand in these trials. Still, this data was used for classification of supported object positions

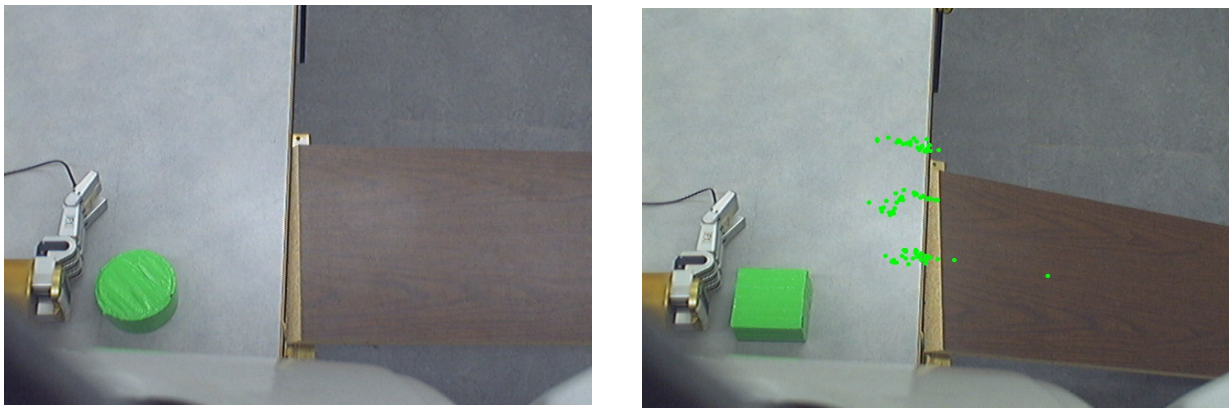


Figure 8: Point clouds for horizontal and sloped configurations. Note that points are generated when the object moves faster than the hand. Because this only occurs for the sloped configuration, points do not occur in the horizontal configuration.

Detecting the Edge of a Constrained Platform

While vision was the modality of interest for the horizontal plane and slanted ramp configurations, it was not of great interest to detect the edge in this particular configuration. Instead, the modality of proprioception was used to detect the point where the platform ended and the upright board began. To use proprioception, it was necessary process the joint torques to find instances where they exceeded their standard values. When viewing the proprioceptive data as a scatter plot over time, it is obvious that certain joints experience a sharp torque increase when the bounding board is met. A plot of proprioceptive data over time is shown to the left.

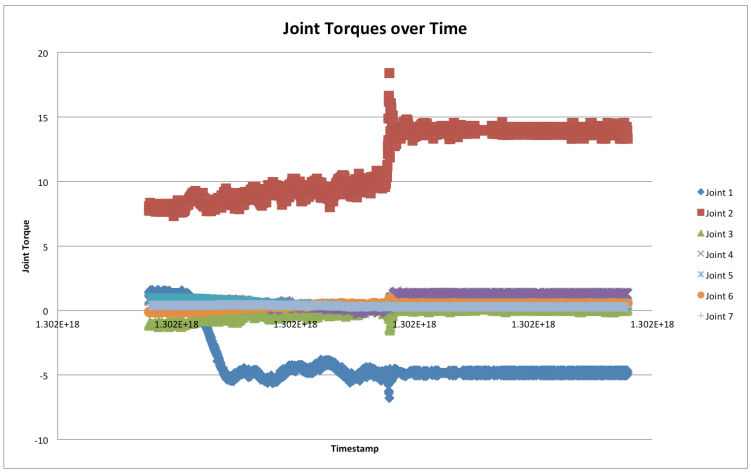


Figure 9: Chart showing the joint torques when pressing against the upright ramp. Note that joint 2 is particularly affected.

After tracking the joint values over time, it became necessary to map them to visual space in order to

verify the accuracy of this detection method. In order to accomplish that task, a series of steps needed to take place. First, OpenCV was employed to visually track the objects as they were pushed against the board. Therefore, the joint and hand positions were recorded for each time step in a trial. With these coordinates, it became possible to associate the location of the object with a particular timestamp. Since the joint tracker determined the time at which torque thresholds were exceeded, it was possible to determine the location of the pushed object at the time where the joint torque threshold was exceeded. Given this information, a point cloud was generated that showed the center of a pushed objected when a joint threshold was exceeded. An image of the created point cloud is shown below.

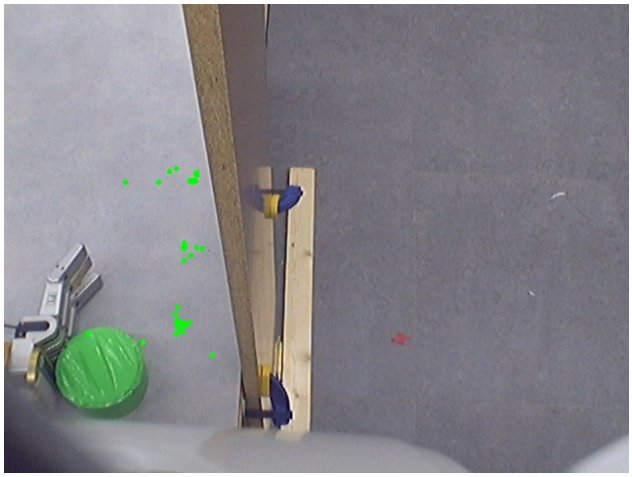


Figure 10: Point cloud generated from the moment where joint torques meet predefined thresholds.

Detecting the Edge of a Cliff

Although an attempt was made to track the object falling off of the platform, it soon proved to be a difficult problem. The object could be tracked as long as it remained on the table but after it was pushed over the edge, it fell to the floor and was no longer in visual space. This was an anticipated problem from this project's inception and was the motivating reason behind constructing the platform ramp. Therefore, this portion of the dataset is being retained for future work.

Processing Depth

The change of height is a key property of a platform's edge. Therefore, the Zcam 3D camera was used in order to detect this change in height. Although the Zcam captures the depth in a scene, this depth image still requires processing in order to be used. For this project, OpenCV was again used to process the images. Although several methods of processing the image were considered, it was ultimately decided that all of the pixel intensities could simply be summed to find overall depth changes in the image.

For each trial, only the first frame of depth data was analyzed to find a change in depth. At the recording of the first frame, the arm, object, table, and platform ramp were all in approximately the same positions regardless of trial. Therefore, the only changes in the image when varying the platform were in the intensity values of the platform itself. For that reason, the method of pixel summation was implemented as a simple means of detecting depth changes within an image.

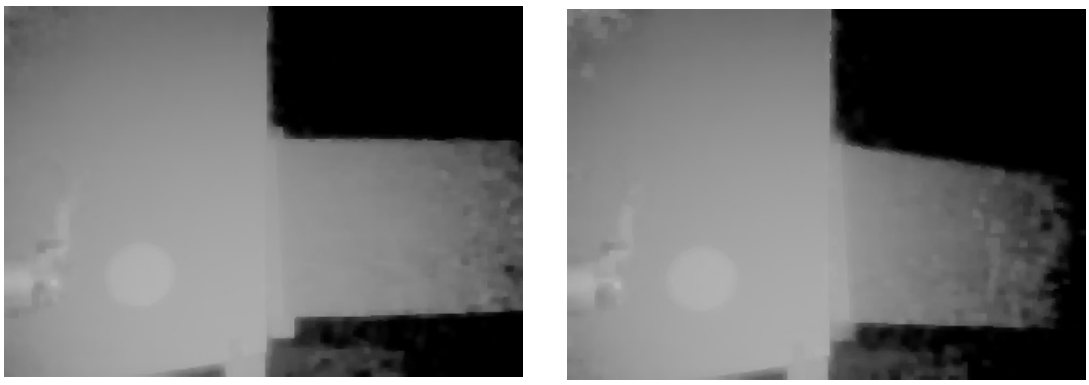


Figure 11: Side by side comparison of horizontal and slanted ramp depth images. The horizontal platform ramp is shown on the left and the sloped ramp is shown on the right.

Results of Experiments

Initial results for this project showed the effectiveness of the edge detection through the various modalities. These results, shown as point clouds in the preceding section, show the effectiveness of the platform sensing visually, but not quantifiably. Therefore, the question became whether or not a robot could determine if an object was still under its control given a coordinate position and a depth image. While similar to the initial idea proposed or this research, the experiment changed as follows.

In the proposal, it was stated that a regression would be fit for the object coordinate positions in order to determine the distance each could be pushed. Basically, for a given object and table position, the classifier would determine the distance that an object could be pushed before reaching the edge. This prediction would be output as a numerical value. However, the slow frame rate of the data captured made this an infeasible task. Whenever the object neared the platform's edge, there were only around three frames in the video stream that captured this moment. Therefore, the experiment changed to include other platform variants and a modified classifier.

With this change in analysis, the learning method becomes the following. Given a depth $d \in D$ where $D = \Sigma$ depth pixel intensities, $t \in T$ where T = object type, and a hand position $h \in H$ where H = {horizontal center of hand, vertical center of hand}, classify $c \in C$ where C = {0 – Supported, or 1 - Unsupported}. In this way the classifier is predicting the notion of support based on the training data.

The training data for this classifier was selected from a subset of the entire dataset. Trials where the platform ramp was oriented horizontally and trials where the ramp was slanted were used for the training and test data. Using these two configurations, it was possible to use depth as an input to the classifier. The horizontal ramp orientation appeared with a greater intensity compared to the slanted ramp. Therefore, all of the trials containing the horizontal platform ramp contained significantly different depth values than the slanted ramp.

After collecting the data for this project, it became clear that learning the affordances of a platform's edge was more difficult to quantify that initially assumed. For this first stage in the overall goal, the focus of this project centered around two platform ramp positions. The positions where the platform was oriented horizontally and placed slanted against the platform became the dataset from which results were drawn. However, there is proposed future work that uses all collected data for analysis.

Algorithms

As the name implies, the J48 tree builds a decision tree to determine if the robot has control of the object or not. Beginning at the top of the tree, each node splits the data into subsets based on one attribute. Selecting the attribute for the each node is based on the information gained, with larger gains placed higher on the tree. The bottom nodes of the tree are the decision values, which in our case are "control" or no control". In this project Weka was used to create the information statistic for each attribute and to generate the decision tree used for classification [16][17].

The k nearest algorithm uses the four attributes to map the data set into R4. Each point in this space also has the classification of controlled or uncontrolled. This method differs from the other two because instead of creating a predictive model, it stores all of the data from the learned set and uses it to find a classification for new data points. The control of a new point is found based on the known points surrounding it. The k closest known points vote for the classification of the new point, with the majority votes determining control. For this experiment, k=5 was selected [18].

The naïve Bayesian classifier assumes that all the attributes are conditionally independent of each other for a given classification of "control" or "no control". Using the training dataset, this algorithm generates probability distributions for each classification P(C), and the conditional probabilities of the attributes given a classification P(Xi|C), where Xi represents hand position, object type and depth measurement. The probability distribution of control given a novel data point is given as

$$P(C|H = h, T = t, D = d) =$$

$$\frac{P(H = h|C) * P(T = t|C) * P(D = d|C) * P(C)}{\sum_C P(H = h|C) * P(T = t|C) * P(D = d|C) * P(C)}$$

Where the hand position, object type and depth value are given by H, T, and D respectively and the current data point has values of h, t, and d. This algorithm will use the probability distribution of the training data set to predict control for the points in the testing data set [18].

Each of these algorithms was tested using a five fold cross-validation method. This method first splits the training data into five different subsets. Then four of these subsets are used as training data, with the fifth used as testing data. This process is then repeated four more times until every subset has been used as testing data.

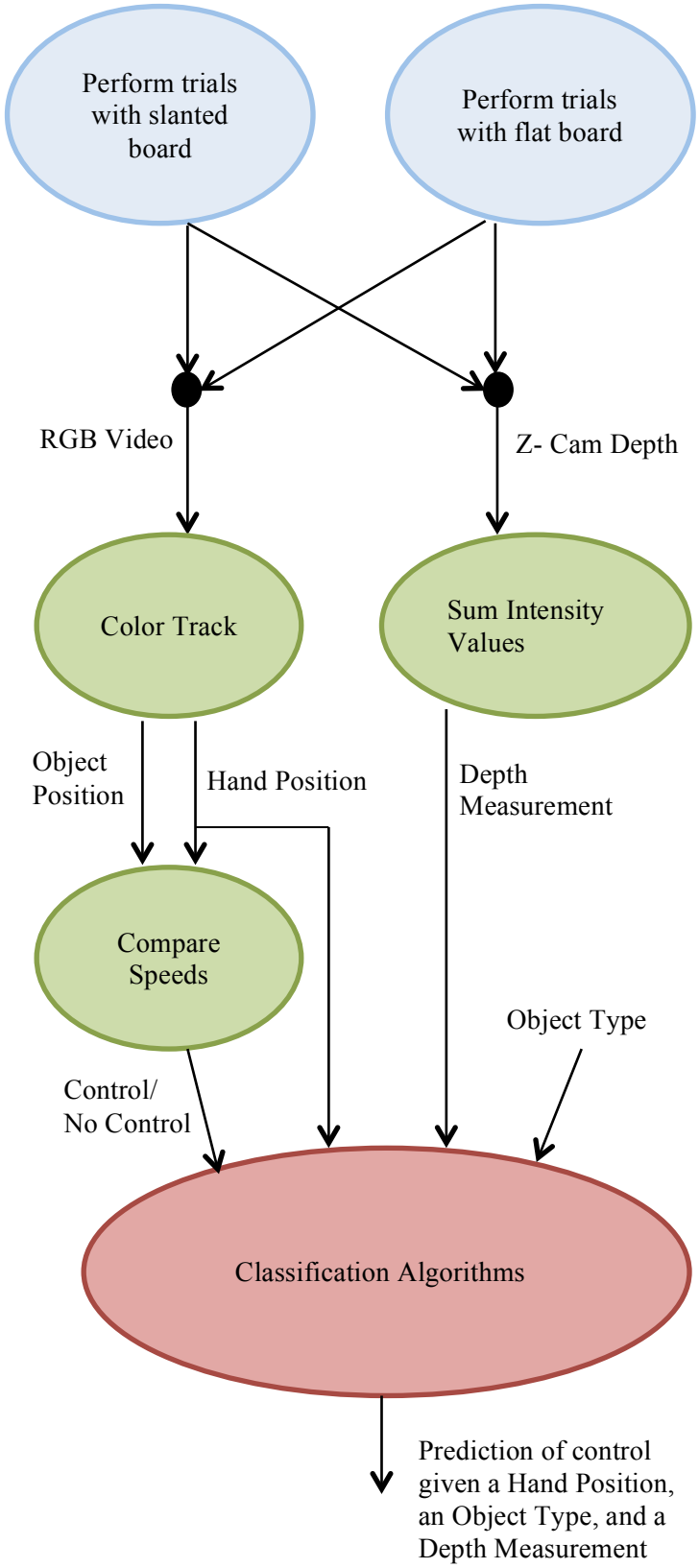


Figure 12: Diagram showing the sequence of the project.

When this process is finished, the results from each of the five iterations can are averaged or combined to create a single model for each algorithm.

Results of Classification

The data used for training and testing was comprised of 3 objects pushed for 10 trials in two board configurations for a total of 3x10x2 = 60 trials. However, each trial lasted a varying amount of time and thus yielded 9772 specific instances for testing and training data. For these 9772 data points, 5475 were instances of supported objects and the remaining 4297 were unsupported objects. Thus, classifying by chance P(C) = 0.56.

Each classification algorithm classified significantly better than chance. The results are summarized in the table below. Overall, the decision tree, J48, slightly outperforms the k-NN in correct classification percentage. Upon examination of the generated decision tree structure, it was found that the tree first looked at the hand's x coordinate in the frame. Below a certain x-coordinate, all y-coordinates for the first two objects are irrelevant. Only the triangle needs additional information before a classification can be made. Depth begins to factor into the decision as the object begins to near the edge of the platform. Given the few number of attributes to classify over and the strong structure of the data, it is fitting that the decision tree has a strong performance. On this dataset, this classifier has a correct classification rate of 98.75%.

Like the J48 decision tree, the k-NN with k=5 also shows strong performance in classification. With each of the attributes providing meaningful information, the k-NN predictably performs quite well. As seen in the visual point clouds, the objects clustered around the edge of the table. Since k-NN looks for similar data points in order to perform a classification, this dataset is classified correctly at a rate of 97.88%.

For this particular dataset, the Naïve Bayes classifier displayed the worst performance among the three algorithms. Because the Naïve Bayes approach considers all inputs separately, this dataset became more difficult to classify. Depth and the x coordinate of the hand are both needed to provide a correct classification. By considering these attributes separately, the correct classification rate falls short of the levels achieved by the other two classifiers. Ultimately, this classifier still performs significantly better than chance with a correct classification rate of 85.24%.

Classifier	Percent of Correct Classification
Naïve Bayes	85.24%
kNN (k=5)	97.88%
J48 (C4.5)	98.74%

Table 1: Results of classifier accuracy for predicting object support.

An Alternative Control Classification

As a parallel to the discussion above, self-detection was explored as an alternative way to classify control. Using this method, the object is controlled only if the robot considers it a part of its own body. The tracking data from the RGB video was processed offline and input into a program to process the entropy measurement of the hand and an associated object. Screenshots of this program are found on the right and show the progression of the program's execution. At the top of each image are two statistical measurements, where each frame considers the information from all previous frames to find the newest value on the far right. The p-value is a measure of the probability that the current observation would be made, assuming that the object is a part of the robot's hand. A larger p value indicates greater confidence that the two objects are dependent, and is a good indication of where the robot has control of the object.

The other measurement is the mutual information of the object and the hand, and indicates the dependence of the two variables on each other. This measurement sometimes provides good information on whether the hand is in control of the object, but it is a less reliable indicator than the p value. For instance, this value drops with the p value as the object begins sliding down the ramp, but rises again towards the end of the video when both the object and the hand come to rest.

If the data used to calculate these statistical measurements is altered to include only the previous three frames instead of every previous frame, they produce a much clearer result. The two graphs below are taken from one trial and show a clear drop off confidence and mutual dependence at around 1.3 seconds. This self-detection classification of control has produced data consistent with the previous classification and is deemed an acceptable alternative.

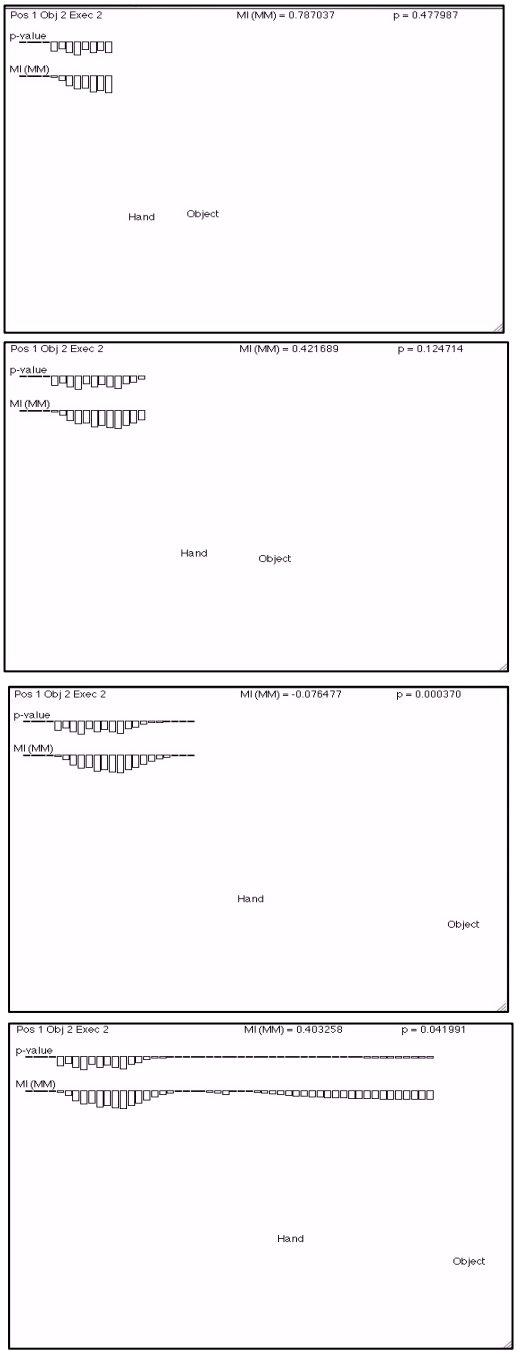


Figure 13: Visualizations of entropy calculation over time.

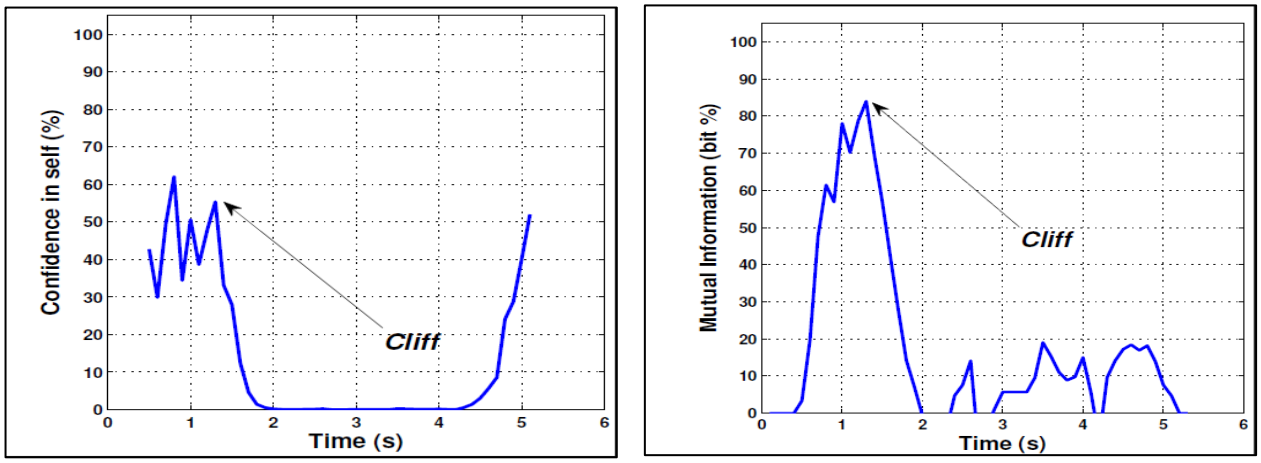


Figure 14: Graph of entropy change over time. Note the cliff when the object separates from the hand. This drop indicates a loss in confidence that the arm has control of the object.

Future Work

In this project, data was collected for four different orientations of the table and the board, but different modalities required different types of measurements. This differing data made it difficult to create a standard attribute list that could be used to classify the data across all orientations. Currently only the two most similar orientations are used by the machine learning algorithms to predict the control of new data, though future work will include all the gathered data.

As shown before, the training data currently provides the machine learning algorithm with the hand position, a depth value from the first frame of the trial, and the object type from each frame of the video. Each frame is also labeled as having "control" or "no control" based on whether the object is moving faster than the hand. This classification works well for the flat plane and the ramp, though it needs to be reconsidered for the other two cases. For instance, in the case of the edge of the table, the object leaves the field of view, making calculating velocity impossible. Also, the current attributes would not be able to distinguish the case of no board from the case of the vertical board because the depth value is not much different between these two cases. A new classification that accounted for

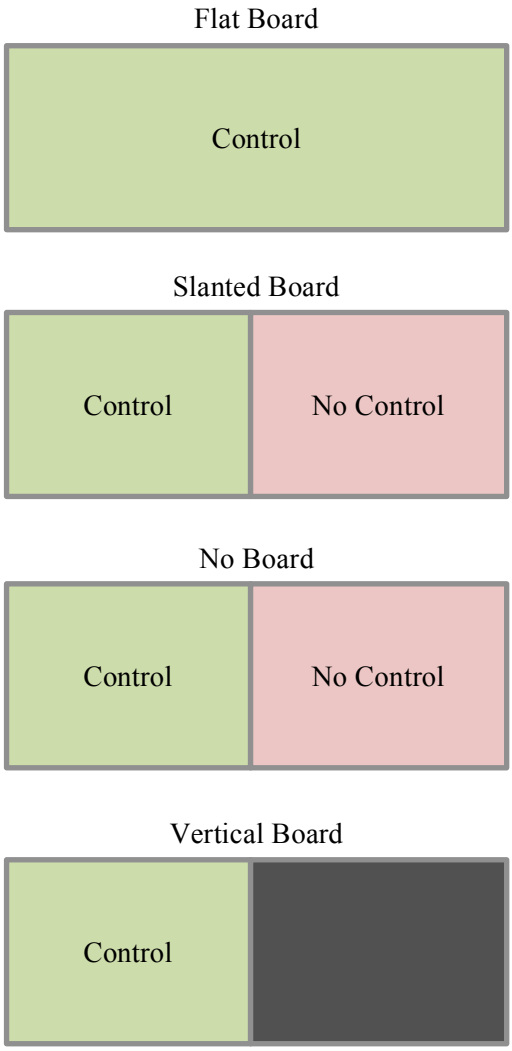


Figure 15: Illustration of regions of control.

these problems was identified, but only after finalizing the current results, making it impossible to incorporate within this project's planned timeline.

The new classification is as follows. First, in any frame that the block could not be seen; the block velocity is given a null value. Second, in addition to the attributes already given to the machine learning algorithm, the new model will provide the maximum sum of the absolute values of the joint torques for each trial. Like the depth measurement, each trial will have one reading of this torque value and it will be shared among all of the frames in the trial. Finally, the requirement for control is changed slightly and will show loss of control if the object's velocity is faster than the hand's velocity or if the object velocity is a null value. Naturally, if the robot can't sense the object, it can't have control over the object.

The expected result is pictured on the previous page, illustrating the regions of control and no control for a given object. The first two are the orientations considered in this project, where the robot has control in every position on the flat board, but loses control on the slanted board at a certain point.

By adding a null object velocity as a condition for no control, the orientation of no board can be added to the data set and should produce control boundaries similar to the slanted board. The robot will maintain control of the object as it moves across the table, but loses control when the object falls off the table and out of view. Here the object has a null velocity and the robot loses control of the object.

The fourth picture illustrates the final case, where the board is placed vertically against the table. Here the robot will have control of the object everywhere that it can move, but will be physically constrained by the board. The black region of the diagram is a region where control cannot be determined because the region is never explored by the robot.

With this model, each frame of the RGB video will have a hand position, an object type, a depth value for the trial, a torque value for the trial, and a control classification. This data can be given to the same machine learning algorithms used for this project to find a more robust prediction of control that incorporates all of the data gathered in this project.

Conclusions

The purpose of this project was to explore the properties and affordances of a platform edge. In order to accomplish this task, the robot performed two experimental behaviors using three different setup configurations. By using the pushing behavior and running tests with a horizontal and slanted platform, machine learning classifiers were able to determine whether an object was supported on a platform with a correct classification rate significantly better than chance. Classification was accomplished using the Naïve Bayes, k-NN, and J48 (C4.5) classifiers.

While the focus of this research was directed toward support classification, the aspect of self-detection also arose as a finding. As the robot moved an object with its hand along a platform, it built the assumption that the object was part of itself. However, this hypothesis was rejected once the object separated from the hand at the edge of the platform. Thus the robot could begin to differentiate the object from its own hand using measurements of entropy.

Future work for this research has been outlined and should focus on analyzing the remainder of the dataset. The robot can truly begin to build its own models of a platform edge once it has

explored the edge using a variety of modalities and learning techniques. Various implementations for this future work have been investigated but the most promising avenue appears in the preceding section of this report.

Acknowledgements

This group would like to thank Jivko Sinapov and Vladimir Sukhoy for their help on this project. Jivko provided a program used to control the robot and record depth data. This program was modified to perform all of the data collection for this project. Vlad used his entropy program to process our data for the alternative control classification.

Work Cited:

[1] R. Baillargeon et al., "The Development of Young Infant's Intuition About Support." Early Development and Parenting Vol 1 (2), 69-78, 1992
[2] B. R. J. Jansen, H. L. J. van der Maas, " The Development of Children's Rule Use on the Balance Scale Task." Journal of Experimental Psychology 81, 383-416, 2002
[3] E. B. Bonawitz, S. Lim, & L. E. Schultz, "Weighing the evidence: Children's naive theories of balance effect their exploratory play." Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology. Cambridge, MA
[4] Walk, R., & Gibson, E. A comparative and analytical study of visual depth perception, Psychological Monographs, 1961, 75(15, Whole No. 519)
[5] Campos, J. J., Hiatt, S., Ramsay, D., Henderson, C., & Svejda, M. The emergence of fear on the visual cliff. In M. Lewis & L. Rosenblum (Eds.), The origins of affect. New York: Plenum Press, 1978
[6] Gibson, E. J., & Walk, R. The "visual cliff." Scientific American, 1960, 202, 64-71
[7] S. Berger, C. Theuring, K. E. Adolph, "How and when infants learn to climb stairs." Infant Behavior & Development 30, 36-49, 2007
[8] Ugur, E.; Dogar, M.R.; Cakmak, M.; Sahin, E.; , "The learning and use of traversability affordance using range images on a mobile robot," Robotics and Automation, 2007 IEEE International Conference on , vol., no., pp.1721-1726, 10-14 April 2007
[9] Gisbon, J.J. 1966. "The Senses Considered as Perceptual Systems." Boston: Houghton Mifflin
[10] Gibson, E. J. (1988). "Exploratory behavior in the development of perceiving, acting, and the acquiring of knowledge". Annual Review of Psychology, 39, 1-41.
[11] Montesano, L.; Lopes, M.; Bernardino, A.; Santos-Victor, J.; , "Affordances, development and imitation," Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on , vol., no., pp.270-275, 11-13 July 2007

[12] Winograd, Terry. 1971. Procedures as a representation for data in a computer program for understanding natural language. Technical report AI TR-17, MIT Artificial Intelligence Laboratory
[13] P. H. Winston, Learning Structural Descriptions from Examples. In P. H. Winston (Ed.), The Psychology of Computer Vision. New York: McGraw-Hill, 1975. pp. 157-209
[14] John Slaney, Sylvie Thiebaux, Blocks World revisited, Artificial Intelligence, Volume 125, Issues 1-2, January 2001, Pages 119-153
[15] Stoytchev, A., "Behavior-Grounded Representation of Tool Affordances," In Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 3071-3076, Barcelona, Spain, April 18-22, 2005

[16] I. H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005
[17] Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993
[18] Pool, David L. and Alan K. Mackworth. Artificial Intelligence: Foundations of Computational Agents. New York: Cambridge University Press, 2010. Print.