Autonomous Systems Practical Extension – WBAI14001.2018-2019.2B Robotics Grasping

Group: `grasp1` — Jördis Hollander (s2956543)

In this paper, automation via robotic arms is explored. A pipeline was developed to allow the Kinova MICO robot arm to grasp box objects from a table. There are several steps in this process.

First, the position of the box object was detected. This was facilitated through provided code and leveraged the Robot Operating System ecosystem. Next, the angle of the box object was classified. The classifier was based on the Support Vector Machine learning model and was trained on a data-set of images of boxes in the real-world environment. Various preprocessing techniques and features were compared and contrasted in the development of the final classifier. Finally, the object was grasped. This was done using the data determined via the position detection and the angle classification. This data was then used to prepare a grasp which was then communicated to the MICO robot arm.

The primary point of research is the comparison of the accuracy of the various classification methods. The pipelines for each classifier consisted of a preprocessing step and a feature extraction step. Various preprocessing steps and feature extraction steps were considered. The classification pipelines were tested for the product of the set of preprocessing steps and the set of feature extraction steps, in other words each preprocessing step was tested in conjunct with each feature extraction step. Each classifier was trialed in the simulation and tested on the real-world image data-set.

The most-accurate classification pipeline was the pipeline which consisted of grayscaling the image as the preprocessing step and extracting the Histogram of Oriented Gradients as the feature extraction step. On the testing data-set (consisting of single boxes at various angles) this pipeline operated at 88.5% accuracy.

I. Introduction

Automation is a fundamental goal for AI and robotics. Through automation we reduce unpleasant workloads for people, thereby freeing human capital to explore other ventures, while simultaneously improving the accuracy and consistency of repetitive tasks. This project explores automation by developing a pipeline that allows a robot arm to grasp a box object from a table. This requires identification of the position of the box, determination of the angle that the box rests at (relative to the head of the arm), and communication of this information to the arm so that a grasp attempt can be made.

The MICO arm is mounted to a table. A camera with a depth sensor is pointed at the table with a fixed viewpoint providing context as to what is on the table. Information from this camera is used to determine the positions of the objects on the table as well as their angular orientation.

The positions of the boxes were identified through provided code that interacted with ROS (Robot Operating System). This means that the focus of this project was on classifying the angle of the box and on communicating this information to the MICO arm. Classification entailed the creation of a classifier. To this end data was collected in the form of images of various boxes at various angles. These images were categorized manually into *buckets* of different angles . Support Vector Machines (SVM) are well suited to creating this sort of multi-class classifiers [1]. However, building the classifier requires extracting features from the images to train the statistical model. To determine a robust solution a variety of prepocessing steps and extraction techniques were attempted on the image data-set.

With the classifier in-place, the next stage is to implement the grasp through message communication. With this the robot is ready to grasp a box from the environment.

A description of the steps that must occur within the system are as follows:

1) The arm should return to its (predetermined) starting position.
2) The information on the positions of the objects that are visible to the camera are retrieved by querying the provided interface to the object detection server.
3) For each detected object, the image of the scene is fed into the classifier to determine the angle of the object.
4) Using the position of the object and the angle, the grasp is constructed and then relayed to the MICO arm.
5) For each object:
    a) The arm moves above the object.
    b) The arm performs the grasp on the object.
    c) If the grasp is successful the arm moves the object to the (predetermined) ending position and releases the object.

In section II the method for angle classification and object communication are elaborated upon. In section III the experimental setup and methodology is described, both of the simulation and of the real-world composition. Section IV details the experiments that were undertaken and lists their results. Section V discusses the results and forms a final conclusion. In this final section, issues that arose throughout the development of the pipeline are also discussed.

II. Method

The approach for developing the pipeline to allow the MICO arm to grasp a box from a table consists of three main sections. A data-set of box images, a classifier that determines the angles of the boxes, and a grasping function that picks up and removes boxes from the table.

A. Data-set Creation

The intention of the classifier is to determine the correct angle of orientation of the box with respect to the arm. A data-set is needed to train this angle classifier. As the context for the objects in the environment come from a consistent fixed viewpoint camera, acquiring the requisite data is simple.

The provided object detection script captured the images from the environment. These images were saved to disk and formed the basis of the data-set. 12 boxes of a variety of shapes, sizes, and colors were used to create the data-set. Each box was placed in up to 13 orientations and tagged with the angle their orientation most closely corresponded to. The angles were one of 0°. 45°, 90°, and 135°.

Previously, fewer types of boxes and a greater variety of possible angle values were considered. However, this lead to inaccuracies in the resulting classifier. These inaccuracies were resolved in two ways. By increasing the variety of boxes in the data-set and by reducing the number of angles considered. This is reasonable for this use-case as it was determined that the arm could grasp a majority of the boxes in their various orientations with an angle of one of 0°, 45°, 90°, or 135°.

B. Constructing the Classifier

With the data-set in place, the classifier could be constructed. The underlying model for the classifier is a SVM. First, the data-set was partitioned into a training set and a testing set.

Each image in the data-set underwent a form of preprocessing and feature extraction prior to training (or indeed is also required before actual classification). There were a variety of possible preprocessing steps and feature extraction possibilities.

The preprocessing steps that were tested are as follows:

- Doing nothing to the image.
- Converting the image to grayscale.
- Blurring the image.
- Blurring the grayscale version of the image.
- Finding the contours of the image.
- Finding the contours of the grayscale image.

Image blurring and contour detection are performed via image filters (convolution kernels) and were executed using `ImageFilter.BLUR` and `ImageFilter.CONTOUR` which are built-in to the Python `Pillow` library. For reference, the convolution kernel for blurring may be seen in equation (1) and the convolution kernel for contour detection may be seen in equation (2).

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (1) \qquad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (2)$$

Next, the feature extraction steps must be considered. According to Dong et al., features based on a Histogram of Oriented Gradients (HOG) and Scale Invariant Feature Transform (SIFT) are two widely-used techniques with good performance for object recognition [4].

The HOG features are computed through several steps [2]. First, the input image undergoes gamma normalization, whereby square-root gamma compression is applied across each color channel. This reduces the influence of illumination in the final feature vector. Next, the gradient image is computed along the x and y of the image. For color images, this is computed across each channel, with the resulting gradient vector corresponding to the gradient vector of the largest norm for that pixel across each channel. The next step is orientation binning. The image is split into cells, each of which consist of 16×16 pixels. Each cell accumulates the information of the orientation of the gradients contained within it. This information is then used to divide the gradient angle into bins spaced over 0° – 180°. These HOG descriptors are then combined into the final feature vector.

As described by Lowe, the computation of SIFT features consists of four primary steps [3]. The first step is scale-space extrema detection which uses a difference-of-Gaussian function to determine points that are invariant to scale and orientation. This is used to compute the difference between the input image and a copy of the image that has been scaled. Each point in the resulting difference is compared against neighboring points at the same scale and neighboring points up and down the scale. If the point is a maximum or minimum against these points then it is an extrema. The second step is keypoints localization and involves identifying keypoints among the identified extrema. A model is fit to determine location and scale and filters candidate keypoints based on stability. The third step is orientation assignment and assigns orientations to each keypoint based on local image gradient directions. This provides invariance to image rotation. The magnitudes and orientations of the local image gradients are sampled around each keypoint location. This gradient information is rotated to align with the keypoint and weighted by a Gaussian with variance proportional to the keypoint scale. This is then used to create the keypoint descriptors is the form of histograms over a window in a keypoint. This final step provides orientation invariance and resistance to local shape distortion and changes in illumination.

However, the resulting keypoint descriptors are not suitable as a feature to train the SVM due to the differing number of keypoints that may be extracted from each image. This is resolved by computing a Visual Bag of Words representation by using KMeans clustering on the extracted SIFT descriptors. The descriptor of the image is then computed by assigning each SIFT of the image to one of the K-clusters. The resulting histogram is then normalized with L1 normalization and is now of consistent length. This is then used as the basis for the SVM.

These feature extraction techniques were contrasted against simply using the entire preprocessed image as the feature. Thus, the feature extraction steps are as follows:

- Leaving the preprocessed image as is (for comparison purposes).
- Extracting HOG features.
- Extracting SIFT features.

The combination of each preprocessing step and each feature extraction step resulted in a variety of pipelines. The most accurate of these pipelines was included in the final application.

C. Grasping

Grasping requires the position and angle for each object on the table. The position is determined through provided code and the angle may be determined via the classifier. These attributes for the object are then added to the internal representation of the environment known as the octomap. With the objects in the octomap, the provided code and the *ROS* ecosystem can handle the motion planning. As a result, grasping then consisted of setting the starting state, moving the arm to the position above the object, and orienting the arm to match the angle of the object. The object is then grasped, with the arm then moving to a predetermined final position. If all the potential grasps fail, the arm then returns to the starting position. The messages for communicating with the arm were constructed using the *MoveIt* framework with the path planning and inverse kinematics integrated into the *ROS* ecosystem facilitating the actual motion computation.

D. Behavior/Architecture

To describe the overall behavior of the system it is useful to consult a state machine describing the basic states of the system during execution. See figure 1.

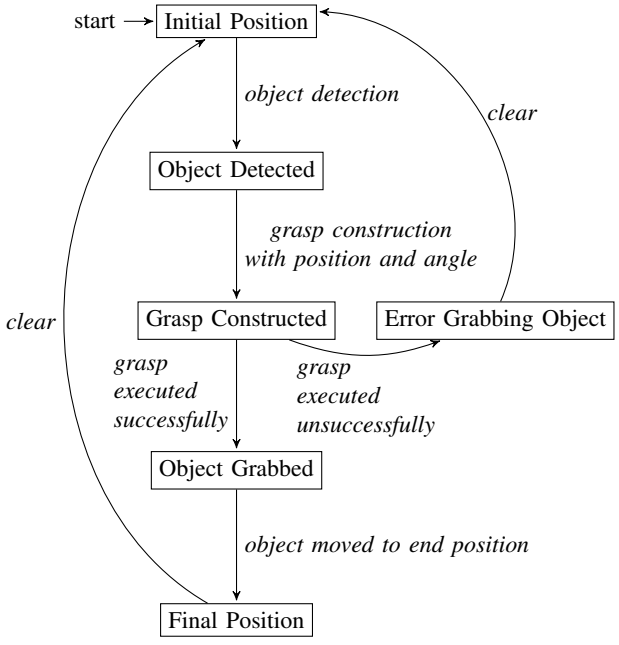


Fig. 1: State Machine for Overall System

The starting state for the system is with the arm being in its **Initial Position**. This entails the arm moving to a predefined start position with its fingers being in the open grasp position. Upon the detection of an object we shift to the **Object Detected** state. As the object has been detected its position may be extracted by communicating with the interface to the object detection server. This also provides an image which may be run through the angle classifier. This provides both the position and the angle which allows the transition to the next state with the **Grasp Constructed**. A grasp may fail or may be successful. On failure the system faced an **Error Grabbing Object** which is responded to by clearing the state and returning the arm to the **Initial Position**. If the grasp has executed successfully the arm has the **Object Grabbed**. In this state the arm moves to the **Final Position**, opening the fingers and releasing the object. Finally the arm then returns to the **Initial Position**. This cycles continues while there are objects to be detected.

III. Experiments

The experiments that could be undertaken fall into two categories. Experiments testing the functionality of the system as a whole and experiments testing the functionality of the sub-components of the system. Of the three primary sub-components: object detection, angle classification, and grasping, angle classification is of particular importance. As a result, the trials were focused on testing the entire functionality of the system and of the capabilities of the angle classifier specifically.

A. Experimental setup

The real-world setup consists of a table with the Kinova MICO arm mounted to it. There is a stationary camera providing a fixed viewpoint of the table. Testing the overall capabilities of the system consists of placing boxes onto the table and running the program. If successful the arm will grasp each box and remove them from the table, dropping them in a bin. An image of the environment may be seen in figure 2.

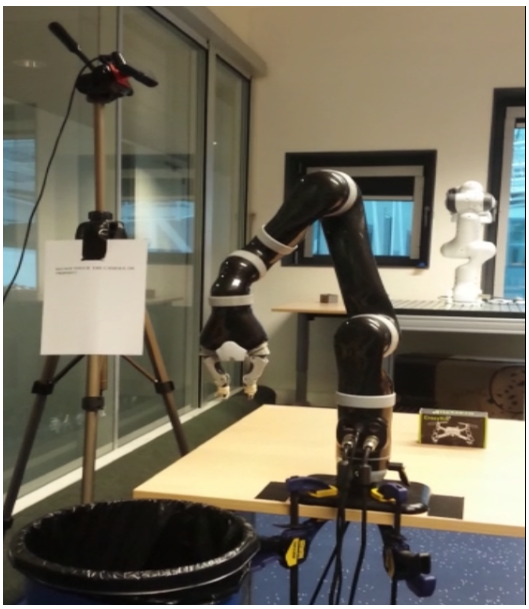


Fig. 2: Real-world Execution Environment

Aside from the real-world setup, the system was also developed and tested using a simulation. *RViz* and *Gazebo* were the tools used to interact and view this environment. As may be seen in figure 3, the virtual environment also consisted of an arm and a fixed viewpoint camera. This camera provides a snapshot of the scene containing the target boxes. The success criterion for the simulation is also a successful grasp of the boxes in the simulated environment.

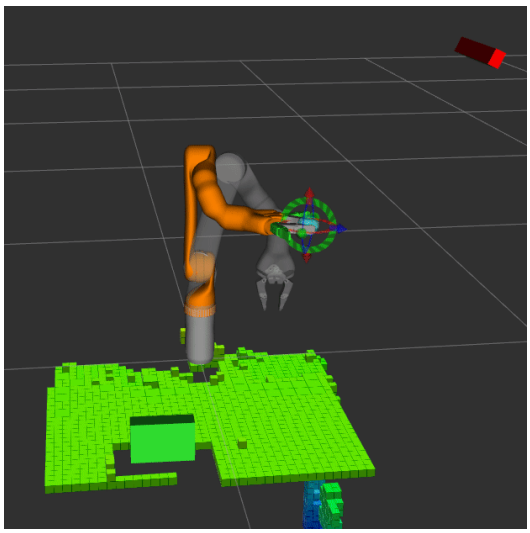


Fig. 3: Visualization of Virtual Environment

B. Experiments on the Classifiers

With respect to testing the classifiers, the setup consisted of constructing the various classification pipelines and executing them on the testing data-set. The value measured was on the accuracy of the classifier and was computed as follows:

$$\text{accuracy} = \frac{\text{number of images classified correctly}}{\text{total number of images}}$$

The testing data-set consisted of 85 images of boxes on the table positioned in angles from 0° to 180°. For each image the system should classify it as the angle it is closest to out of 0°, 45°, 90°, and 135°. The results for the various pipelines are presented in section IV. 85 images provides a reasonably large data-set for testing and exhausts the boxes of the right dimensions and mass that were reasonably accessible.

The most successful pipeline from these tests, i.e. grayscale preprocessing with HOG feature extraction, was used as the pipeline for the experiments on the overall system.

C. Experiments on the Overall System

For testing the overall system, three categories were candidates for measurement: the execution time, the accuracy of the system, and the robustness of the system.

1) Testing Timing: How long does it take to remove one box from a table? Measuring this consisted of setting up a trial box in the environment and executing the system. The time taken from the start of the execution to the end (with the arm returning to the start position) was measured. Though the execution time was not the priority for the project, it was important to ensure that the real world execution time was reasonable.

2) Testing Accuracy of Single Box Configurations: For the real-world environment, 7 different boxes were tested in 4 different configurations each. The boxes may be seen in the appendix. Each test for each box for each configuration was trialed twice. This resulted in a total of 56 trials. The four configurations were 0°, 45°, 90°, and 135°.

For the simulated environment, the same trials were conducted. Boxes of similar proportions were tested with the four configurations of 0°, 45°, 90°, and 135°. As may be seen in the results for these trials, the system was very inaccurate in the simulation. As a prerequisite for the robustness trails the system should perform well in the single box configuration test. Thus, the trials for robustness were not conducted for the simulation.

3) Testing Robustness: The robustness of the system was tested qualitatively. A variety of edge cases and multiple box configurations were tested. Each configuration was trialed twice. The results include a list of the various configurations and whether the system functioned correctly for that configuration. The limits that were tested are as follows:

- **Limits of possible locations to grasp on the table.** There are two obvious limiting factors. First, the box must be within the bounds of the octomap virtual planning environment. Second, the box must be reachable by the arm. Problem cases are when the box is too far away from the arm or too close for the joints of the arm to contort themselves. The limits are defined in terms of the distance in cm from the base of the arm. These limits will be tested with box 1 from figure 5 as it is easy to detect, classify, and grasp.
- **Limits of object detection (minimum size and transparency).** As may be seen in figure 1, `object detection` is required before any other step in the main execution pipeline. Obviously, the object must be within the visual field of the camera. However, there may be a lower limit on dimensions of the object where it stops being detected. The system was tested on boxes of different sizes to determine this lower limit. The system was also tested on transparent objects to determine if transparency would cause issues in object detection.
- **Limits of grasping (object thickness, object height):** When grasping an object the system is conservative in how far the fingers close. This is to prevent the arm from crushing the objects that it is grasping. Additionally, the arm may be unable to grasp an object which is too short. These limits were tested on boxes of different thicknesses and heights to determine possible limitations.
- **Limits of classification (various object dimensions).** Boxes were acquired outside of the initial testing data-set to test edge cases in object classification. This included boxes, that were tall and thin or short and fat.
- **Limits of classification with multiple independent objects.** The images the classifier was trained on consisted

of single boxes. Multiple boxes being present in a scene may cause issues in angle classification. This was tested by placing multiple boxes in the scene simultaneously to determine how the classifier would react. Some tests involved placing boxes behind each other and in other tests the boxes were separated with space between each other.

- **Limits of classification with stacked objects.** As with the previous limitation, stacked objects may pose a problem. The configurations that were considered were boxes that were stacked with the same orientation and perpendicular to each other. Boxes of different sizes that were stacked were also considered in the same fashion.

The aim of these tests was to provide a sense of the operational boundaries of the system and to help guide future development. The specific tests and their outcomes are detailed in section IV.

IV. Results

A. Results for the Timing Experiments

For the real world environment the runtime was approximately 44 seconds with a variance of approximately 3 seconds based on the position of the box on the table. The results for these tests in the virtual environment were highly variable due to limitations of the simulation. It is of little real world significance and depends highly on the computational resources available. These experiments have confirmed that the real world execution time is reasonable.

B. Results for Classifier Experiments

The classifier pipelines consist of a preprocessing technique and a feature extraction technique. In table I the accuracy for each pipeline (preprocessing technique $\times$ feature extraction) is shown.

Preprocessing Technique	Feature Extraction		
	Raw Image	HOG	SIFT
Color (no preprocessing)	75.6%	73.5%	–
Contour extraction	62.9%	70.1%	–
Blur	75.6%	87.3%	–
Grayscale	73.4%	88.5%	76.6%
Grayscale contour extraction	62.9%	84.2%	73.4%
Grayscale blur	76.6%	88.4%	77.8%

TABLE I: Accuracy of Various Classification Pipelines

First, it is interesting to note how high the accuracy is for the pipeline constructed from unprocessed raw images. However, this is likely to hold only for the limited scope of the project. As more complex boxes or orientations enter the fray it is unlikely for this pipeline to maintain its accuracy.

In the case of HOG feature extraction it is notable that the pipelines with the preprocessing techniques involving grayscaling operate at a higher accuracy. The intuition for this is that the absence of color improves the consistency of HOG feature extraction. The same logic lead to the incorporation of blurring.

Blurring the image (either in color or in grayscale) seems to result in a relatively accurate classifier irrespective of the feature extraction technique employed.

However, simply grayscaling the image and performing HOG feature extraction operates the best at 88.5% accuracy.

C. Results for Overall System Accuracy Experiments

Table II show the results for the trials for the various orientations for boxes 1–7 in the real-world environment. A ✓ indicates if the trial was successful (the box was grasped and removed from the table), while a ✗ indicates if the trial was unsuccessful.

Box	Orientations							
	0°		45°		90°		135°	
	1	2	1	2	1	2	1	2
1	✓	✓	✓	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓	✓	✓

TABLE II: Trials for System in Real-world Environment

As may be seen, the system was successful in every case. This may be explained by the relative simplicity of these orientations. When running the classifier based on grayscaled HOG features on the subset of the testing data-set that correspond to these images, it operates at a 96% accuracy. However, as may be seen in table III this does not hold when considering the trials in the simulated environment.

Box	Orientations							
	0°		45°		90°		135°	
	1	2	1	2	1	2	1	2
1	✓	✓	✗	✗	✓	✓	✗	✗
2	✓	✓	✗	✗	✓	✓	✗	✗
3	✓	✓	✗	✗	✓	✓	✗	✗
4	✓	✓	✗	✗	✓	✓	✗	✗
5	✓	✓	✗	✗	✓	✓	✗	✗
6	✓	✓	✗	✗	✓	✓	✗	✗
7	✓	✓	✗	✗	✓	✓	✗	✗

TABLE III: Trials for System in Simulated Environment

For the simulated environment every box that is at 45° and 135° fails. As a result, the accuracy in the simulated environment is 50%. The only portion of the pipeline that could lead to this sort of failure for these simple cases is the classifier. When testing this assumption, it was noted that each image from the simulated environment corresponding with the angles of 45° and 135° were classified as 0° or 90° degrees. The reasons for this are elaborated on in section V.

Ultimately, irrespective of the issues in the simulation, the results for the real-world execution holds the most significance. Thus, it is now important to consider the real-world performance for more complex scenarios.

D. Trials for Robustness of Overall System in Real-world Environment

The trials for robustness are qualitative and used to determine the limits of the system. So, a variety of limits associated with different parts of the system were considered.

1) Limits of possible locations to grasp on the table: For the tested box which was 9.5 cm high, it could be grasped when it was a minimum of 10 cm from the base of the arm and a maximum at the very edge of the octomap at approximately 67 cm from the base of the arm. When it was placed closer or further away the arm was unable to perform a successful grasp.

2) Limits of object detection (minimum size and transparency): If the object is transparent, such as with the tested plastic bottle, the system fails to detect the presence of the object. Additionally, when the system was tested with a Rubix cube with each side being 1.5 cm, it was discovered that the system could only detect the object if it was sufficiently close to the camera. When the cube was placed more than 45 cm away from the edge of the table facing the camera, it ceased to be detected. Thus, it is clear that based on the size and proximity to the camera the ability of the system to detect objects varies. The precise relationship and the lower limits of this were not determined.

3) Limits of grasping (object thickness, object height): During the tests on the various boxes it was determined that the minimum thickness required for the object to be grabbed was approximately 1.3 cm. Any smaller and the grip was too weak to be able to grab the object.

If the arm is oriented correctly, it is able to pick up objects only 1 cm high. This was tested on a small box (1 cm x 4 cm x 6 cm). However, if the orientation results in any of the fingers scraping against the table, the arm responds with an internal error signaling the end of the grasp.

4) Limits of classification (various object dimensions): When testing additional boxes outside of the original testing data-set, it was determined that tall and thin boxes which stood high were categorized as having an angle of 0°, irrespective of its orientation. The effect of the depth plays a role in determining the angle, and the reduced depth leads to misclassification.

5) Limits of classification with multiple independent objects: Though the object detection detects the multiple objects as independent objects, the classifier seems to misclassify the image, as it considers the overlapping boxes as one large box. In certain circumstances (such as if the multiple boxes have the same angle) the system may still function correctly.

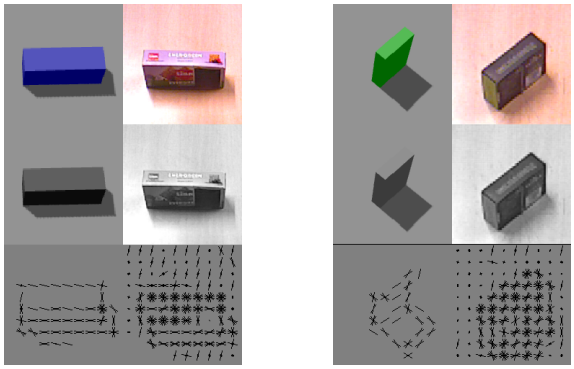
6) Limits of classification with stacked objects: Objects which are stacked with the same orientation, tend to be classified correctly (though it is likely that they are classified as one large object). Each object in the stack would then be removed sequentially.

Items of a similar size that are stacked perpendicular to each other, result in a classification that matches the orientation of either one object, the other object, or the average of the two. In any case, this configuration is unlikely to lead to a successful series of grasps.

Items where a larger box is involved in a stack with a smaller box is likely to be classified in terms of the orientation of the larger box.

V. Discussion & Conclusion

The aim was to develop a pipeline allowing the Kinova MICO robot arm to grasp boxes from a table. This required object detection, angle classification, and grasping. Object detection was provided out of the box and its limits were tested. Small objects that were far away and transparent objects pose problems for object detection. However, for a majority of the real-world scenarios tested, the object detection succeeds. Angle classification entailed the selection of the best combination between a prepocessing step and a feature extraction step. It was found that grayscaled images when run through HOG feature extraction had the best results, with 88.5% accuracy on the testing data-set. However, the classifier has difficulty in handling tall, thin objects and has great difficulty classifying correctly in the simulation. The reason for this is apparent when considering figure 4.



(a) 0° (b) 135°

Fig. 4: Processing Pipeline for Real-World vs. Environment for Several Angles

In both figure 4a and figure 4b, the HOG features are far noisier in the real-world environment than the simulated environment. However, as the classifier has been trained on these noisy features it can handle the classification correctly. Additionally, for the simulated box in figure 4a the classifier responds correctly. This is to be expected given that the shape of the resulting features are almost characteristic of that angle. However, when considering the simulated box for figure 4b, the classifier classifies incorrectly. This is evident as after grayscaling the shadow ends up as a very prominent feature. This distorts the extracted HOG features and causes the classification to fail. This is less relevant than the real-world performance but did impact the development process.

With classification handled, the final component was grasping the object. For the simple boxes that were tested, the grasping was successful in every case. However, there were limitations to the placement of the object (being too close or too far) and of the dimensions of the object (too thin). Ultimately, the resulting system is able to handle the detection, classification, and grasping of simple box objects robustly. However, more complex configurations involving multiple overlapping objects or stacked objects cause problems at the angle classification layer. Further research in segmentation of the objects, classification of these segments, and the determination of the ordering of events is required.

REFERENCES

[1] Chih-Wei Hsu and Chih-Jen Lin, A Comparison of Methods for Multi-class Support Vector Machines, National Taiwan University, 2015.

[2] Dalal, N. and Triggs, B., Histograms of Oriented Gradients for Human Detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, San Diego, CA, USA.

[3] David G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, University of British Columbia, 2004.

[4] Li Dong, Xinguo Yu, Liyuan Li, and Jerry Kah Eng Hoe, HOG Based Multi-Stage Object Detection and Pose Recognition for Service Robot, ICARCV, 2010, pp. 2495-2500.

APPENDIX

ADDITIONAL IMAGES

Fig. 5: Boxes 1–7 Used for Testing