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

Group: `grasp1` — Jördis Hollander (s2956543)

Abstract— **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 to the this process:**

1) **Detecting the position of the box object: This was facilitated through provided code and leveraged the ROS (Robot Operating System) ecosystem.**
2) **Classification of the angle of the box object: This was performed by creating a data-set of images of boxes in the real-world environment. A classifier was then trained on that data. Various preprocessing techniques and features were compared and contrasted in the development of the final classifier. The Python machine-learning framework `scikit-learn` was used to create the classifier based on the Support Vector Machine (SVM) learning model.**
3) **Grasping the object: The grasping data determined via the position detection and the angle classification is used to prepare a grasp which is then communicated to the MICO robot arm through the *MoveIt* framework.**

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.

The preprocessing steps which 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.**

The feature extraction steps which were tested are as follows:

- **Leaving the preprocessed image as is.**
- **Extracting HOG features.**
- **Extracting SIFT features.**

The classification pipelines were tested for the product of the set of preprocessing steps and the set of feature extraction steps, i.e. 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 grayscaled image with HOG feature extraction. On the simple testing data-set (consisting of single boxes at various angles) this pipeline operated at 88% 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 to the arm of this information so a grasp attempt can be made.

The MICO arm is mounted to a table with a fixed viewpoint camera providing context as to what is on the table. Images from this camera are 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. 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 (0°, 45°, 90°, 135°). Support Vector Machines (SVM) are well suited to creating this sort of multi-class classifiers [2]. 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.

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.

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 [1]. These features were contrasted against simply using the preprocessed image as is.

Thus the tested feature extraction steps are as follows:

- Leaving the preprocessed image as is.
- Extracting HOG features.
- Extracting SIFT features.

With the classifier in-place, the next stage is in implementing the grasping through message communication via the *MoveIt* framework. With this in the place the robot is ready to grasp a box from the environment. A description of the process is 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 *Alice* object 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 the information is relayed to the MICO arm.

In the remainder of the report, the method for angle classification and object communication are elaborated upon. The experimental setup and methodology is described, both of the simulation and of the real-world composition. The experiments that were undertaken are detailed, the results described and the final conclusion formed. 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 parts. 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°.This configuration for the data-set was maintained in the `JSON` data format.

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. To begin with the data-set was partitioned into a training set and a testing set. The underlying model for the classifier is a SVM. The training process leveraged the `scikit-learn` machine learning framework.

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 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.

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 of these resulted in a variety of pipelines. The most successful of these pipelines had to be determined and was done so by testing each pipeline on the testing data-set. The most accurate pipeline was included in the final application. The results of these tests are included in section IV.

C. *Grasping*

The process for grasping requires the position for each object on the table and the angle for each object. 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 simulation of the environment. Grasping then simply consisted of setting the starting state to the starting position with the fingers closed, moving the arm to the position above the object, and orienting the arm to match the angle of the object. The object may then be grasped, with the arm then moving to a predetermined final position. If all of 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.

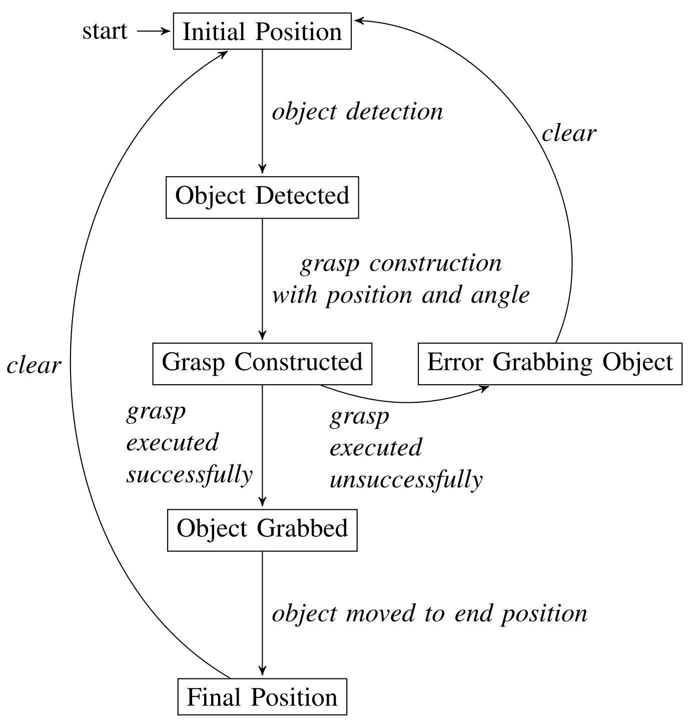


Fig. 1: State Machine for Overall System

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 via the *Alice* object. The object 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 the bin. An image of the environment may be seen in figure 2.

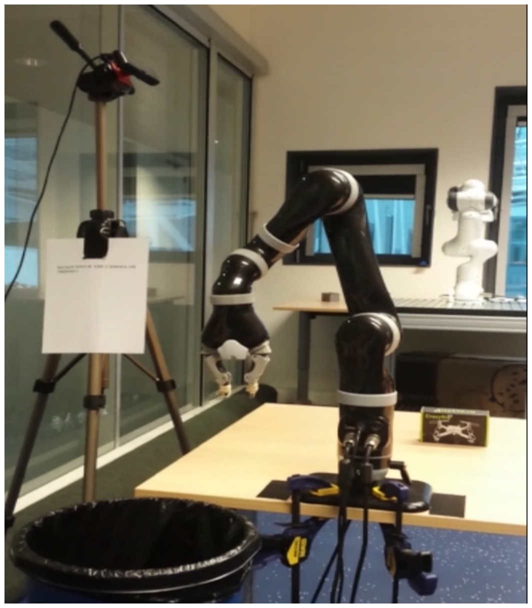


Fig. 2: Real-world Execution Environment

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 simulation also consisted of an arm and a fixed viewpoint camera. This camera provides a snapshot of the scene containing the target boxes.

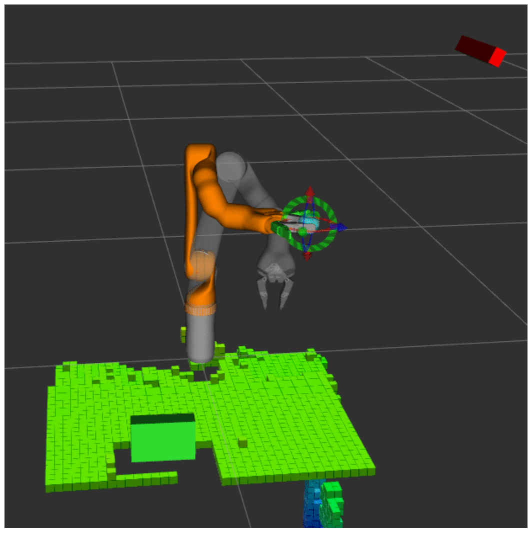


Fig. 3: Simulated Execution Environment

The success criterion for the simulation is also a successful grasp of the boxes in the simulated 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 in a variety of orientations. 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. In the real environment the outcome was a more consistent than the virtual environment. For the real environment the runtime was approximately 44 seconds consistently 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 and the power of the hardware available to the simulation.

As these measurements were not of main focus for the project, the timings were not included as a result and testing the timing was discarded.

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°. Each trial was successful.

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 result, 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
- Limits of object detection (minimum size and transparency)
- Limits of grasping (object thickness, object height)
- Limits of classification (various object dimensions)
- Limits of classification with multiple independent objects
- Limits of classification with stacked objects

The specific tests and their outcomes are detailed in section IV.

IV. Results

A. *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 × 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 an 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.

B. 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 for the simple cases in the real-world the system was successful in every case. This may 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, 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° degrees 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. As a result, it is now important to consider the real-world performance for more complex scenarios.

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

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

1) Limits of possible locations to grasp on the table: There are two obvious limiting factors. First, the box must be within the bounds of both the camera and 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. When testing with a box that is both easily detectable, classifiable, and graspable these limits were found.

2) 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, 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): 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 attempting to grasp. As a result, there is a minimum thickness required for the object in order to be grabbed. The minimum thickness was approximately 1.3 cm.

If the arm is oriented correctly, it is able to pick up objects only 1 cm high. This was tested on a Ricola lozenge box. 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 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: The images the classifier was trained on consisted of single boxes. When multiple boxes are present in a scene it is possible that the image has the boxes overlapping. Though the object detection may detect these as multiple objects, the classifier is likely to misclassify the image, as it considers the overlapping boxes as one large box. However, it may be lucky and determine the correct angle.

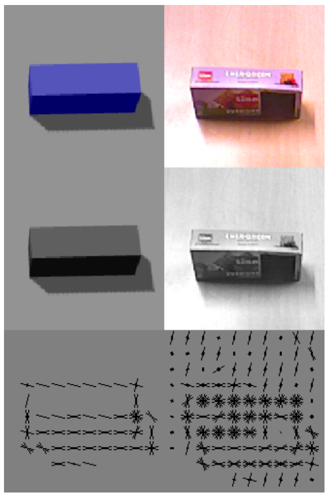
6) Limits of classification with stacked objects: As with the previous limitation, stacked objects pose a problem. Object which are stacked with the same orientation, tend to be classified correctly (though it is likely that they are classified as one large object). Each object in the stack would then 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 box objects from a table. To this end, the three major components of object detection, angle classification, and grasping were considered. 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 determined, that grayscaled images when run through HOG feature extraction lead to the best results, resulting in 88.5% accuracy on the test data-set. The classifier however, has difficulty in handling tall, thin objects and has great difficulty classifying correctly in the simulation. The primary reason for this is apparent when considering figure 4.


(a) 0°

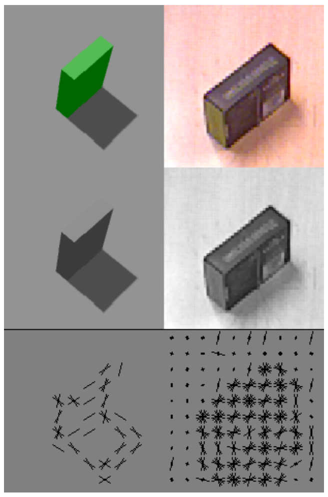

(b) 135°

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, given that the classifier is trained on these real-world noisy features it is able to 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 is almost characteristic of that angle. However, when considering the simulated box for figure 4b, the classifier is unable to classify correctly. 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 handled 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] 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.

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

Appendix

Additional Images

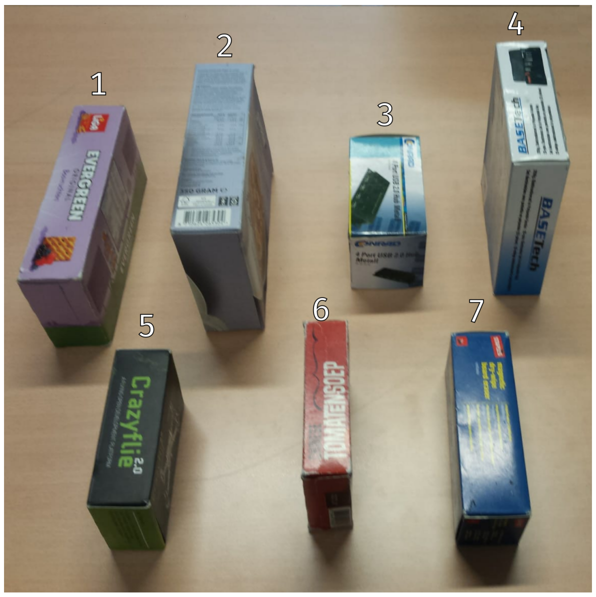


Fig. 5: Boxes 1–7 Used for Testing