# Enabling Block-Stacking Intuition in Robots

Geethika Hemkumar

***Abstract*—Building stable structures is important for a variety of tasks, such as construction and warehouse organization. Utilizing robots can improve the efficiency at which these tasks are completed. In order for this to become possible, robots must be taught to understand properties of stability and how to apply them towards building stable structures. This course project investigates utilizing block face surface area as an indicator of stability. A keypoint detector is trained to estimate the boundaries of each of the top faces of a collection of blocks within a scene. These detections are used to estimate their surface areas. Experiments evaluate the accuracy of the keypoint detector and further analyze the ability for these detections to estimate to determine an order, based on surface area, in which to stack the blocks.**

## I. Introduction

Building stable structures is important to a wide variety of applications, including compact storage in warehouses, organization of items in a refrigerator, and most notably construction. While stacking objects in domestic environments is relatively trivial for humans, stacking in warehouses and in construction for long periods of time can be strenuous for humans. Thus, developing the ability for robots to understand the properties of and build stable structures is appealing. A robot's ability to understand and apply concepts of structural stability has the potential to augment and increase the efficiency of human efforts.

Two sub-tasks, both related to stability, of the block stacking task are: understanding the order of which to stack a set of blocks, and ensuring that two subsequent blocks are placed in a stable manner relative to each other. When stacking objects, humans primarily rely on visual cues to ensure stability. For the former sub-task, ordering, size and shape are taken into account: typically, objects with flat faces that have relatively large surface areas are placed at the bottom, and more irregular/smaller/curved objects are placed closer to the top. For the latter sub-task, object alignment, the intersection between the area of the faces stacked on top of each other is maximized.

These and other intuitions about object stacking that humans have developed provide useful insights for the development of stacking capabilities in robots. In my course project, I will focus on investigating the former task: investigating the correlation between object face surface area and structure stability. More concretely, based on intuition and results from prior work, I hypothesize that the surface area of the largest face on each object can provide a useful comparator for object ordering.

The primary contribution of this project is a keypoint detection model based on the YOLOv8 model, where the keypoints correspond to the corners/points of interest on the face of a given 3D object, in this case cubes. These keypoints can be used to estimate the surface area of a face of the cube.

The model is able to predict the expected keypoints in an image observation containing several cubes reasonably well with a small amount of training data. In addition to analyzing the quality of the predictions given by the model, my experiments evaluate the ability to predict a cube's face surface area from the predicted keypoints and further generate a stable order of the cubes based on surface area. The experimental results show that while keypoint detection is reasonably accurate, the ability for these detections to accurately estimate the surface areas of cube faces and generate a stable stacking order based on this is not consistent. Further work is needed to understand how size can more accurately be perceived from an image.

## II. Literature Review

Prior work relevant to this project can be organized into two categories: **stability prediction/intuition** and **stacking policy formulation**.

### A. Stability Prediction/Intuition

[2] develop an algorithm for creating a stable vertical stack using 6 irregularly shaped stones. Their approach utilizes a physics simulator to determine the next best object to place on top of the existing stack, along with its optimal pose. This work formalizes intuitions about stability by noting the points of contact between two stones, the location of an object's center of mass, and the support polygon formed between two successive stones in the stack when considering favorable stacking poses. Similarly, in my project, I formalize intuition regarding to block face surface area as it relates to stability.

[3] contributes the ShapeStacks dataset, a large set of 3D objects with varying shapes and sizes, along with numerous images of stacks that are labeled to reflect their structure and stability. More specifically, two stability violations are annotated: *violation of the center of mass criterion*, which depends on how objects are positioned relative to others in the stack and notates when the overall center of mass of the stack becomes unstable, and *violation of the planar surface criterion*, which discourages the placement of blocks on top of a curved surface. These violations are used to train a stability predictor, which provides intuition regarding a stack's stability. Further, as shown in the paper, the insights gained from the stability prediction model allow for an understanding of the *stackability* of an object. This provides a metric by which a given set of blocks can be ordered, which aids in stacking.

Figure 6 in the ShapeStacks paper [3], referenced in Figure 1 above, shows a positive correlation between the *stackability*

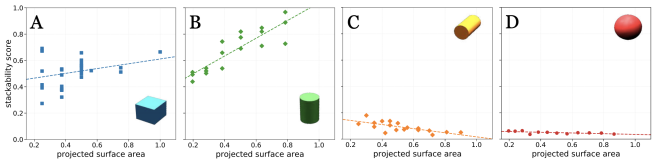


Fig. 1. Figure from ShapeStacks paper showing correlation between projected surface area on the x-y plane and the stackability of various objects

score of a block and the size of its projected surface area for objects with numerous stackable faces, such as cubes and cylinders, and negative or no correlation between these properties for objects with low *stackability* scores. This suggests the benefits of learning model to predict object face surface areas for the block stacking task.

The data collection and algorithm formulation of the aforementioned works serve as inspiration and a foundation for my own approach.

### B. Stacking Policy Formulation

Two works from DeepMind [4], [1] explore developing policies to enable robot arms to stack a set of objects.

The aim of [4] is to explore the robotic stacking task using a custom set of non-trivial block shapes. This is done in order to enable the robot to manipulate a variety of shapes and understand their relations in regards to stacking. Two tasks are explored: *skill mastery*, focused on enabling the robot to master 5 specific stacking tasks, and *skill generalization*, focused on creating a general model to enable the robot to stack a variety of object sets. An imitation learning-based reinforcement learning policy is trained to teach the manipulator block-stacking skills.

[1] is a more recent work from DeepMind that focuses on creating a single transformer model that is capable of performing a wide variety of tasks. The model is conditioned on images of the end state. RoboCat can learn new tasks/generalize to new robots with few demonstrations, and the experiments section shows its ability to perform stacking tasks, both in simulation and in the real-world.

My project is primarily focused on visual-based intuition about structure stability and formulating a block stacking order based on these intuitions, whereas the former papers focus more heavily on teaching stacking skills to a robot manipulator through reinforcement and demonstration.

## III. Task Description

In a given block stacking task, an agent will be given a set of $n$ blocks of varying shapes and sizes. The agent's goal is to build the tallest structure as possible without collapse. Like in prior work, only *single-stranded* [3] stacks will be considered.

Each stacking scenario contains 5 differently-sized cubes. The number and variety of blocks can be increased; the extension will be described in Section VIII.

A separate environment in robosuite based off of the existing stack environment is created for this task. This custom-designed robosuite environment allows an arbitrary number/variety of blocks to appear (as many as allowed given the constraints of the table size) and utilizes a custom placement initializer to allow for placing the blocks in various scenarios, mainly for data collection purposes.

At the beginning of each stacking task, the five blocks are placed in a line on the table. During intermediate states of the task, part of the blocks will be in their initial positions, and the rest will be on the block stack.

## IV. Data Collection

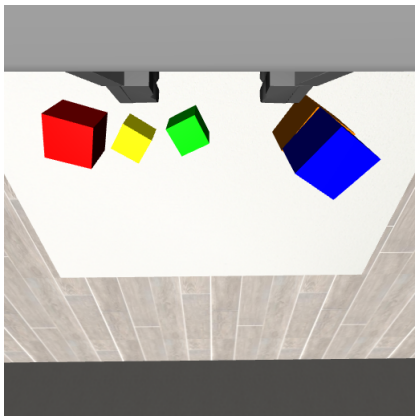

Fig. 2. Example of an intermediate state of the robosuite environment for this task

I collected data within the robosuite [6] simulation environment. To encourage the model to understand both initial and intermediate states, described in Section III, I collected image observations of both scenarios. An example of an initial state of the workspace is shown in Figure 3, and an intermediate state is shown in Figure 2.

To achieve this, I created a custom placement initializer within my robosuite environment, which chooses a random number of blocks out of the five to form a stack with and places the remainder of the objects on the table as they would appear in the initial state of the workspace. Given this random number of blocks to stack, n, the placement initializer will determine the n cubes with the largest faces. These cubes will be stacked in order of their face size, which is the desired order for the robot to stack them in.

After the scenario is generated, the robot arm will move to a position where the majority of the workspace state can be observed. Image observations from the *eye_in_hand* camera are collected.

I collected data of 96 different scenarios, each with a corresponding image observation. In order to use this data to train my keypoint detector, the observations are labeled with the desired keypoints. This is described further in Section V.

I considered using the ShapeStacks dataset for model training. However, the ShapeStacks project was focused on learning physical intuition about stack stability and did not utilize a robot. For teaching robots block-stacking intuition, observations from the robot's point of view would be of higher quality

than those from a global frame of view. An interesting point of future work would be to investigate how the observation point of view impacts the robot's understanding of objects in the scene and, further, its ability to stack these objects in a stable manner.

## V. Technical Approach

The overall algorithm is as follows:

1) Collect image observation of the workspace
2) Utilize keypoint detector to identify the corners of a face for each one of the cubes
3) Utilize keypoint predictions to estimate the area and center of each detected block face

### A. Keypoint Detector

Given an image observation of a set containing a number of cube-shaped blocks, a keypoint detector is used to estimate the corners of the face of the cube that is facing upward in relation to the table.

For this task, a pre-trained YOLOv8 pose model is fine-tuned on the dataset I collected within robosuite. For model training, each cube within an image observation is annotated with a bounding box to identify its class and a set of four keypoints to identify the four corners of the upward-facing face.

### B. Alternative Surface Area Prediction Model Approaches

Prior to using a keypoint detection model, I considered a few other approaches. The first was segmentation. I first considered utilizing the segmentation renderings provided by robosuite, but soon discovered that these would only be capable of providing cube segmentations per instance and each face of the cube may not be easily distinguishable from the others. Similarly, I considered training a model which would learn a segmentation mask of a cube face, which can then be used to estimate the face's area. The second approach I considered was training a model to directly estimate the areas/positions of cubes in an image. However, this method may require significant data collection in order to enable the model to generalize to a variety of cube sizes and rotations. Finally, I considered using CenterNet [5] to estimate the centers of each of the top-facing cube faces. CenterNet uses keypoint estimation to estimate the object's center. Using this keypoint, CenterNet can then estimate the size of the object. However, I hypothesized that using keypoint estimation to directly predict four corners of a cube face is better tailored to this surface-area based block ordering task.

### C. Estimating Surface Area from Keypoint Predictions

Using the inferred keypoints, the cube face's surface area can be estimated by determining the length of the edges formed by successive keypoints, then squaring this edge length. The four edge lengths of the polygon are averaged to give an estimation of the square's side length, and the surface area is estimated by squaring this value.

Figure 3 shows keypoint detections for an example scenario, numbered in the order in which they appear in the output array of keypoints. The keypoints are always labeled in a cyclical manner, so that edges can be formed from successive keypoints. Thus, in each scenario, the four edges are formed from the following pairs: {0, 1}, {1, 2}, {2, 3}, and {0, 3}.

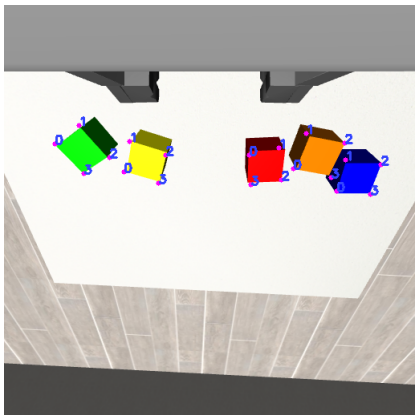

Fig. 3. Visualization of numbered keypoint predictions, used to form edges and estimate surface area of the cube face

Given the surface areas of each of the faces of the cubes in the workspace, the cubes are sorted in descending order by face surface area. Cubes with larger face surface areas are designated to be closer to the bottom of the stack.

## VI. Experiments

My experiments seek to answer the following questions:

1) How well does the keypoint predictor perform on test data?
2) How well does the keypoint predictor estimate a block stacking order based on face surface area?

Each experimental scenario contains five differently-colored cubes, each with a side length between 0.02 to 0.05 units (this dimension range seems to be the most reasonable given the size of the robot's gripper).

Table I shows the five categories of scenarios that are evaluated in the experiments.

| **Scenario Name** | **Cube Sizes** |
|---|---|
| *same-small* | All cubes are the same size with a 0.02 unit side length |
| *same-med* | All cubes are the same size with a 0.035 unit side length |
| *same-large* | All cubes are the same size with a 0.05 unit side length |
| *diff-small* | All cubes are differently sized, where the range of sizes is relatively small, between 0.02 and 0.035 units |
| *diff-large* | All cubes are differently sized, where the range of sizes is relatively large, between 0.02 and 0.05 units |

TABLE I
Details of Experimental Scenarios

### A. Keypoint Predictor Performance

To analyze the performance of the keypoint predictor, I considered all five scenarios described in Table I.

To evaluate the effectiveness of keypoint detection on each of these scenarios, the average Euclidean distance between the corresponding ground truth and predicted keypoints is computed over all keypoints in an image, where $\hat{p_x}/\hat{p_y}$ are the predicted keypoint coordinates and $p_x/p_y$ are the ground-truth keypoint coordinates:

$$E_{avg} = \frac{\sum \sqrt{(p_x - \hat{p_x})^2 + (p_y - \hat{p_y})^2}}{N_{keypoints}}$$

Table II shows the average Euclidean distance error for the keypoint detection scenarios tested.

| Scenario Name | Average Euclidean Distance Error |
|---|---|
| *same-small* | 1.4929 |
| *same-med* | 1.6705 |
| *same-large* | 2.3626 |
| *diff-small* | 1.8386 |
| *diff-large* | 5.8307 |

TABLE II
AVERAGE EUCLIDEAN DISTANCE ERROR FOR KEYPOINT DETECTION SCENARIOS FROM TABLE I

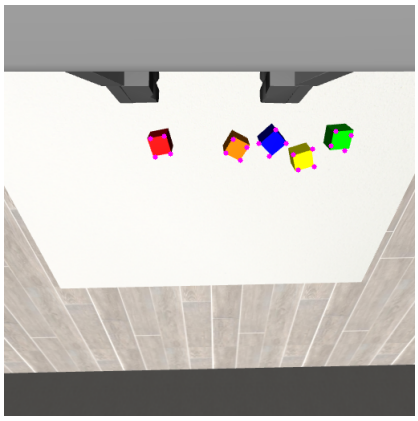

Fig. 4. Keypoint Predictions for *same-small* scenario

Figure 4 shows the keypoint predictions for the first scenario referenced in Table II. The predictions are close to the ground truth, as the predicted keypoints are close to the corners of the top-facing cube faces. The predictions for the green cube, however, seem to be slightly rotated. This may be due to the influence of other cube rotations seen in this this scenario (for example, the orange cube seems to be rotated in a similar manner to how the keypoints for the green cube are predicted) or the influence of green cube rotations in other scenarios.

Figure 5 shows the keypoint predictions for the second scenario referenced in Table II. Similar to the *same-small* scenario, the keypoint predictions are close to the ground truth. However, as evidenced by Table II, the error for this scenario is larger than that of the *same-small* scenario, and a visual comparison corroborates this observation.

Figure 6 shows the keypoint predictions for the third scenario listed in Table I. Compared to the *same-small* and *same-med* scenarios, the keypoints are further from the ground truth. The error for this scenario is the largest of the three scenarios with all cubes of the same size. This may be because the model did not see many cubes of this size during training and hence has not developed strong generalization capabilities to this cube size.

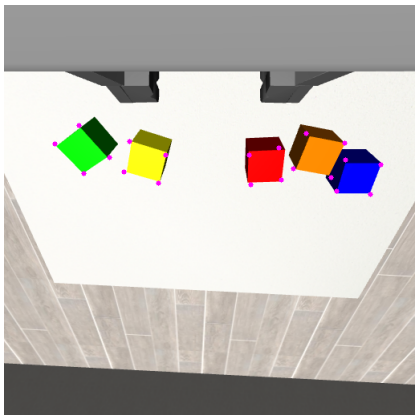

Fig. 5. Keypoint Predictions for *same-med* scenario

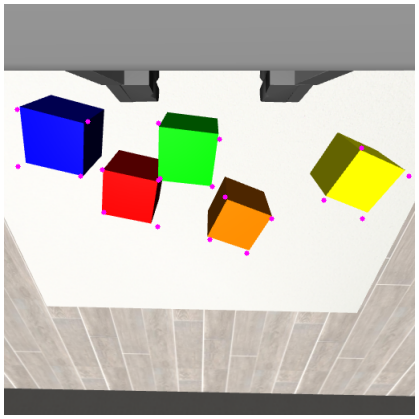

Fig. 6. Keypoint Predictions for *same-large* scenario

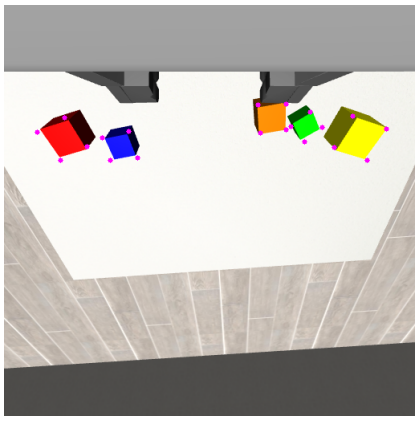

Fig. 7. Keypoint Predictions for *diff-small* scenario listed in Table I and scenario 1 listed in Table III

Figure 7 shows the keypoint predictions for the fourth scenario referenced in Table II. Compared to the prior scenarios, the predictions are further off from the ground truth. I hypothesize that this implies that the model may need more

training on image observations with a larger variety in cube sizes.

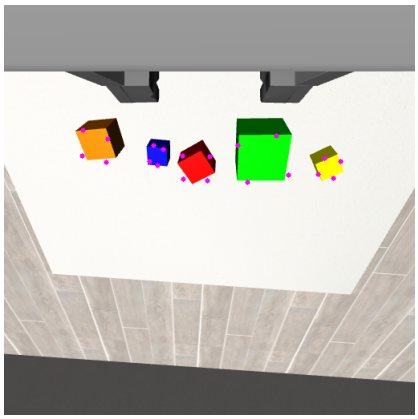

Fig. 8. Keypoint Predictions for *diff-large* scenario listed in Table I and scenario 1 listed in Table IV

Figure 8 shows the keypoint predictions for the fifth scenario referenced in Table II. Visually, the keypoint prediction performance in this scenario is the worst of the five scenarios presented, which is supported by the numerical error data. A combination of the large range in cube sizes and the presence of larger sized cubes, neither which may not have been presented often during training, may be the reasoning for this.

Overall, the keypoint detection model is able to identify the desired keypoints reasonably accurately, as shown visually and numerically. The main exceptions to this are keypoint detections that appear on the midpoints of cube edges or are slightly rotated in relation to the face corners (though the size of the square formed seems visually similar to that of the cube). This provides promising evidence that increasing the amount and variety of scenarios in training can improve the model's performance.

### *B. Stacking Order Estimation Performance*

To analyze the ability for the keypoint predictions to estimate a stacking order for the cubes based on face surface area, I considered the scenarios from Table I that involve differently sized blocks, which are *diff-small* and *diff-large*. Both the edit distance, computed using the NLTK library [1], between the ground truth and predicted sequences and the average Euclidean distance error, which is computed in the same manner as the errors in Table II, are reported for each scenario.

Table III shows the comparison between the ground truth and predicted stacking order for three variations of the *diff-small* scenario. The edit distance between the ground truth and predicted stack orderings increase as the average Euclidean distance error increases. This is expected, since the accuracy of the keypoint detection model will affect the quality of the cube face surface area estimations and hence the estimated stacking order. Figures 7, 9, and 10 show the keypoint predictions for scenarios 1, 2, and 3 in Table III respectively.

[1]https://www.nltk.org/

| **Scenario Number** | **Predicted Order** | **Ground-Truth Order** | **Edit Distance** | **Average Euclidean Distance Error** |
|---|---|---|---|---|
| 1 | Y, R, O, B, G | Y, R, O, B, G | 0 | 1.8386 |
| 2 | O, B, Y, R, G | O, Y, B, G, R | 2 | 3.4117 |
| 3 | O, B, G, R, Y | R, G, B, O, Y | 3 | 5.2306 |

TABLE III
PREDICTED STACK ORDERING BASED ON SURFACE AREA FOR *diff-small* SCENARIOS (R = RED CUBE, O = ORANGE CUBE, Y = YELLOW CUBE, G = GREEN CUBE, B = BLUE CUBE)

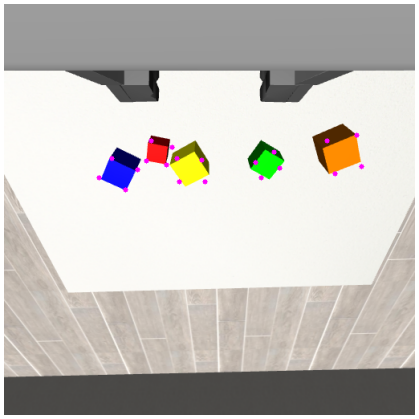

Fig. 9. Keypoint Predictions for *diff-small* scenario 2 listed in Table III

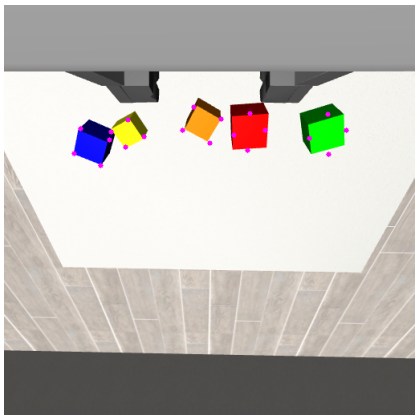

Fig. 10. Keypoint Predictions for *diff-small* scenario 3 listed in Table III

The predicted order for scenario 1 is correct. For scenario 2, both the yellow and blue and the red and green cube pairs are swapped in the ordering. Visually, the red and the green cube sizes are similar and the yellow and blue cube sizes are similar, which may explain these ordering errors. A similar reasoning as to why the blue and green cubes are swapped in the predicted ordering for scenario 3 applies.

Table IV shows the comparison between the ground truth and predicted stacking order for three variations of the *diff-large* scenario. Interestingly, unlike the trend in the results

| Scenario Number | Predicted Order | Ground-Truth Order | Edit Distance | Average Euclidean Distance Error |
|---|---|---|---|---|
| 1 | G, O, R, Y, B | G, O, R, Y, B | 0 | 5.8307 |
| 2 | O, Y, R, G, B | Y, O, R, G, B | 1 | 2.2881 |
| 3 | G, B, O, Y, R | G, Y, B, O, R | 2 | 1.8807 |

TABLE IV
PREDICTED STACK ORDERING BASED ON SURFACE AREA FOR *diff-large* SCENARIOS (R = RED CUBE, O = ORANGE CUBE, Y = YELLOW CUBE, G = GREEN CUBE, B = BLUE CUBE)

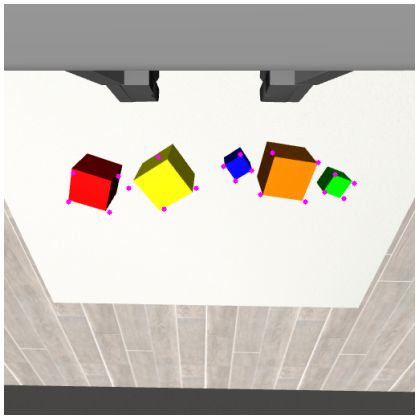

Fig. 11. Keypoint Predictions for *diff-large* scenario 2 listed in Table IV

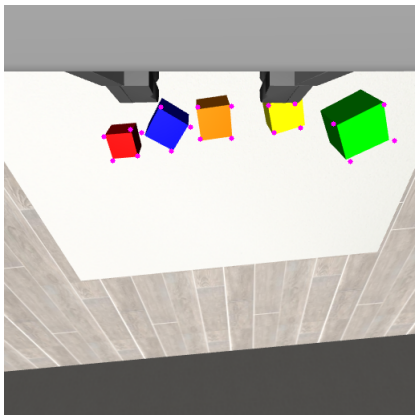

Fig. 12. Keypoint Predictions for *diff-large* scenario 3 listed in Table IV

from Table III, the average Euclidean error decreases as the edit distance increases. One possible explanation for this is that cube edge distances are not accurately represented by the detected keypoints, which affects the surface area estimations. Figures 8, 11, and 12 show the keypoint predictions for scenarios 1, 2, and 3 in Table IV respectively. Notably, a corner of the yellow cube in Figure 12 is occluded which affects the prediction of this keypoint.

The predicted order for scenario 1 is correct. For scenario 2, the yellow and orange cubes are swapped in the ordering. Visually, the yellow and orange cube sizes are similar.

These results convey that utilizing keypoint predictions to estimate cube face surface area provides inconsistent results. The performance of surface area prediction is highly dependent on the ability of keypoints to accurately estimate cube dimensions. Further, the perspective of the image is important, as this affects perceived cube sizes.

## VII. FUTURE WORK

There are a variety of avenues for future work. The most direct one would be to analyze ways to improve the performance of the current keypoint model, or to investigate other model architectures that may improve the performance of keypoint detection. Regarding the first question, I hypothesize that a wider number and variety of scenarios (i.e., scenarios with larger variety of block sizes and rotations) will improve the performance of the model.

Another area of future work would be to increase the variety of block types, as done in the ShapeStacks paper. This would require adding more classes/keypoint structures to the model to support different 3D object face shapes (circles, triangles, etc.).

Based on observations from the human perception system, we tend to perceive object sizes differently based on how close or far the object appears to be. Understanding how to more accurately estimate properties of objects that are further away would be an interesting point of future work.

Finally, to improve the sim-to-real transfer of this solution, investigating the impact of noise (i.e., in image observations and solutions to improve the robustness of the algorithm in the presence of noise) would be useful.

## VIII. CONCLUSION

This course project investigated utilizing a physical intuition about stable block stacking to form a stack ordering of a given set of cubes. To estimate this ordering, a keypoint detector model is trained in order to identify the corners of a face of each cube in the scene, then additional post-processing is done to estimate the surface areas based on these keypoints and determine an ordering based on this, where cubes with larger face surface areas are designated to be lower on the block stack. While the keypoint detection is reasonably accurate, the estimation of cube face area that is extracted from these detections is not a strong/consistent estimator of the true cube face size. Further work is needed to understand how size can more accurately be perceived from an image.

## IX. RESOURCES UTILIZED

1) I used the *Stack* environemnt from the robosuite GitHub repository https://github.com/ARISE-Initiative/robosuite/blob/master/robosuite/environments/manipulation/stack.py as a starting point for creating my own environment
2) Computer Vision Annotation Tool (CVAT) to annotate keypoint data
3) This script https://github.com/computervisioneng/pose-detection-keypoints-estimation-yolov8/blob/main/CVAT_to_cocoKeypoints.py as a starting point to

convert my annotations from CVAT to the format needed by the keypoint detection model in YOLO. I extended this script to take into account multiple object classes and potentially multiple objects in a single observation.

## References


[1] Anonymous. Robocat: A self-improving foundation agent for robotic manipulation. *Submitted to Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=vsCpILiWHu. Under review.

[2] Fadri Furrer, Martin Wermelinger, Hironori Yoshida, Fabio Gramazio, Matthias Kohler, Roland Siegwart, and Marco Hutter. Autonomous robotic stone stacking with online next best object target pose planning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2350–2356, 2017. doi: 10.1109/ICRA.2017.7989272.

[3] Oliver Groth, Fabian B. Fuchs, Ingmar Posner, and Andrea Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[4] Alex X. Lee, Coline Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg, Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, Claudio Fantacci, Jose Enrique Chen, Akhil Raju, Rae Jeong, Michael Neunert, Antoine Laurens, Stefano Saliceti, Federico Casarini, Martin Riedmiller, Raia Hadsell, and Francesco Nori. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *Conference on Robot Learning (CoRL)*, 2021. URL https://openreview.net/forum?id=U0Q8CrtBJxJ.

[5] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[6] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.