Shape Replication with 3D Blocks

performed by the Kuka iiwa robotic arm

Shreya Gupta
Massachusetts Institute of Technology
Department of Computer Science
shreyag@mit.edu

Sharmi Shah
Massachusetts Institute of Technology
Department of Mechanical Engineering
sharmis@mit.edu

***Abstract*—Block stacking is a long standing problem in robotic manipulation. We seek to design a robotic system that can decompose any given object into unit blocks and replicate the given object as a 2D shape with building blocks. By implementing a connected perception and planner module, complete with a DiffIK system and specifically oriented gripper trajectories, we have created a robot capable of replicating a variety of shapes. We analyzed the performance of this replication by identifying the pixel depth differences between the original shape and the replicated one. It was concluded that while the robot can recreate a shape with an average difference of $7.36 * 10^{-7}$meter, vertical stacking results in more error than horizontally placing blocks.**

I. Introduction

Over the years, improvements in robotics have drastically improved the efficiency and abilities of the manufacturing industry. The construction of objects from smaller components is a very relevant problem in manipulation. For example, robots that pack products on assembly lines use similar techniques to pick up objects and place them into their desired packed configuration. When packing objects close together and in a specific desired configuration, there are also challenges involving how the robotic arm interacts with objects that have already been placed. There are also limitations in moving objects using a robotic arm with different workspaces as well as understanding the characteristics of block stacking in various configurations. Additionally, while robots are often used to perform repetitive tasks, research has shown that via perception and planning, robots can accomplish much more. There may be a need for robots to accomplish semi-repetitive tasks in the manufacturing industry that can build and assemble more customized objects via camera sensing and planning while also maneuvering between other objects to prevent collisions in a tight space.

The premise of this project is to have a robotic arm replicate a given shape using blocks. The simulation is able to place blocks in a variety of different given configurations. The main question we ask is how well can a robotic arm, specifically the iiwa, replicate a series of different shapes. As a follow up, how can we specifically design the gripper trajectories to prevent collisions with already assembled blocks. If contact modelling, gripper orientation, and block release height are tuned, the system will be able to successfully replicate the given shapes. Furthermore, the shapes with more horizontally placed blocks will have less error in replication than those with vertical stacking.

In order to have the robot pick and place the individual cubes, the shape has to be decomposed into individual blocks. The difference in the positions of the given shape versus the final arrangement of blocks will define a metric for the system's performance. The system's design goal was an ability to replicate a shape in a 2D plane with minimal error using a perception system and a planner. The results suggest that the system can meet this goal with a correct contact geometry and block release height. Furthermore, the vertical stacking is more challenging for this system when compared to horizontal placement.

II. Related Work

The crux of our project is stacking blocks into various shapes, which seems like a relatively mundane task. However, block stacking is often times the baseline task for showing that a robot has a certain capability. The advantages of the the task is that it can be well constrained, you have control over the objects, and the objective is clear, stack the objects. To contextualize our project, we will go through some papers in which block stacking has been used as a jumping point for interesting research. The first of which is work on the CoSTAR Block Stacking Dataset by Hundt et al [1]. The authors argue that block stacking is not a trivially simple problem, due to aspects such as many solutions not taking advantage of the full range of motion of the end effector or being limited by the senors being used. The paper showcases a database called CoSTAR, where the robot stacks blocks in a less environment with obstacles. The authors showed that many deep learning models failed on the the CoSTAR database and performed a search for ones that do. This is just one example of how block stacking can be used to test both the capabilities of a robot and models. Other examples of this include another paper by Hundt et al. called ""Good Robot!": Efficient Reinforcement Learning for Multi-Step Visual Tasks with Sim to Real Transfer" where the authors attempt to improve reinforcement learning efficiency on tasks where there are "dead end" actions [2]. They use block stacking in order to do this as it's a long term multi-step task, which is more difficult than a single pick and place. These papers motivate our final project since it demonstrates that stacking blocks is

a task that can encompass all parts of a robotic manipulation system and is a good benchmark to showcase the capabilities of a system.

A component of our project is decomposing an object into unit building blocks. This is a problem that has been explored in literature, and we are going to take a look at an example of one such paper. In the paper "An efficient algorithm to decompose a compound rectilinear shape into simple rectilinear shapes", Sharif et al. explore a methodology for decomposing a rectilinear shape into smaller rectangles while optimizing for the minimum difference between the perimeter of the original shape and the sum of the perimeters of the shapes that it was decomposed into [3]. Sharif et al. motivate their work by looking at the ability to decompose areal photos of buildings where the combination of buildings makes it unclear where one starts and the other one ends. In order to do the rectilinear decomposition, Sarif et al. start by finding interior and exterior corner points by matching them to templates that represent corners. One drawback of this is that their algorithm does not work well on distorted or skewed images. After identifying the corner points, the paper describes an algorithm that works from the interior corner points to decompose the image. The first part of our project, decomposing the shapes we are trying to build is clearly very closely tied to this work. As you will read later in the paper, our decomposition algorithm ended up being an extremely simply raster order decomposition. The reason we were able to do that was because all of our sub-units were the same size and shape. The method proposed in this paper has the advantage of decomposing the shape into a small number of rectangles. So, if given blocks of different shapes and sizes, it would be advantageous to use a more complex algorithm such as this one. One trade-off that would come with using such an algorithm is that it does not take gravity into consideration. Due to the use case of this algorithm, there is no need for the sub-shapes that the shape is decomposed to be "stack-able", so if this algorithm were to be used for manipulation purposes, some modifications would have to be made. So in order to use blocks of different shapes and sizes, one not only have to use a more computationally intensive algorithm but also make modifications to fit the needs of a manipulation system that is stacking objects.

The examples used in Drake and in the MIT's Robotic Manipulation class provide a decent amount of the technical groundwork for developing such a system. For instance, a planner subsystem is present in the Clutter Clearing example that was referred to in this research [4]. There are also examples for contact geometry and picking up simple foam blocks. Within these examples however, none explore creating shapes with blocks or extensively stacking these blocks. Previous research has used machine learning models to perform multi-block stacking [5]. Yet another research paper focused on stacking with a gantry crane in the steel industry with constraints including time windows and minimum relocation error uses dynamic optimization to find a solution. However, in this paper we focus on a non-optimized stacking solution. This can not only help us identify the impact of machine learning on such tasks, but also identify how far performance can reach without training a model.

III. Methods

A. Setup

A kuka iiwa robot arm joined with a wsg gripper hand was used to perform the series of manipulation tasks in this project. The simulation environment was composed of 3 zones: a building materials zone, a construction zone, and an object zone (Figure 1). In the building materials zone, an array of 1x1x1 cube blocks with a length of 0.05m was placed (we use 1x1x1 to represent a "unit block" where the sides are all the same length but not necessarily a specific unit). In the object zone, we placed a 3 dimensional object (a Tetromino) that could be feasibly replicated using unit blocks (Figure 2). This means the Tetrominos that were going to to be replicated could not have any overhangs, or floating blocks. A camera was placed in front of the object zone to perform the perception task. Finally, the construction zone was the area where the iiwa would replicate the given Tetromino by building the shape block by block. There is a camera pointing at the construction zone as well. This construction zone camera has the same pose relative to the construction zone, as the object zone camera relative to the object zone. The depth image from construction zone camera and the the depth image from the object zone camera are used to measure accuracy of placement.

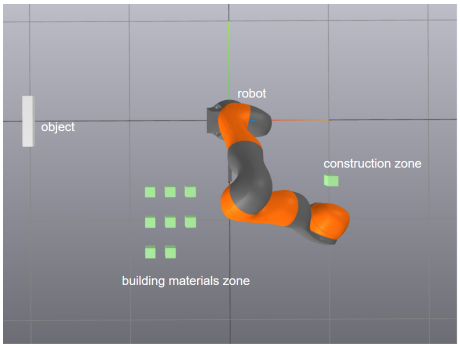

Fig. 1. The simulation environment was composed of 3 zones: a building materials zone, a construction zone, and an object zone. All three are shown from an aerial view in the simulation visualizer.

The set of Tetrominos used in this paper is shown in Figure 2. They were created for various purposes. For example, the "i block" was used to see how stable the blocks could be in a tall configuration. We also created some of the Tetrominos randomly to see how the iiwa would behave for different configurations, and if there were any variations in stability for different categories of shapes and if any conclusions could be drawn from those variations.

The unit blocks used in the simulation were created with small spheres at each of the eight corners of the block to improve contact dynamic simulation.

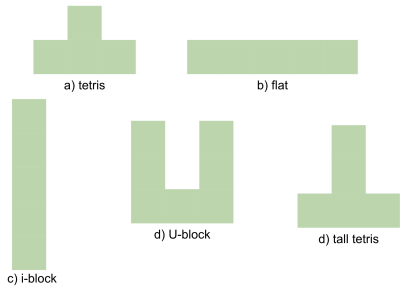

Fig. 2. 5 different blocks were used to test how the iiwa would behave with different configurations. The parameters of block release height were tuned to the flat and i-block configurations - the most simple of the shapes.

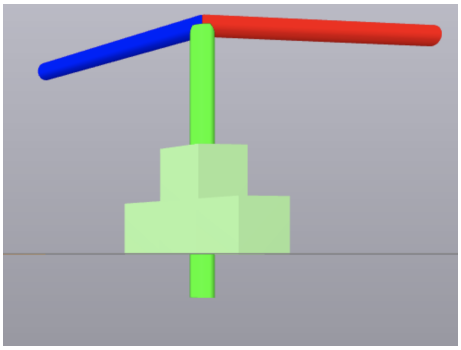

Fig. 3. An example of a 3 dimensional version of the block that the robot was attempting to recreate

B. Vision

The first step of the process of replicating a Tetromino is perception. As mentioned in the setup section, we have a camera pointed directly at the Tetromino we are trying to replicate. We use the the RGB sensor to get both the color image and the depth image of the Tetromino, but we only use the depth image to generate poses for placement. After we get the depth images from the camera, we need to convert these values to points in the camera frame. In order to do this, we use the pinhole camera model to get p_C, the Tetromino in the camera frame.

From there, we need to decompose this Tetromino into unit cubes. We start this process by first finding where the Tetromino starts in our data. So, we want to find the bottom left point of the Tetromino as a starting point, which we simply do by finding the first row from the bottom where not all depth values are infinity, and then left to right searching that row until we come across the first value that is not infinity. This gives us an anchor point for the Tetromino, because we are given no information of where within the object zone the Tetromino will be. From there, the process is relatively simple. Since for the Tetromino, we have both an image in pixels and all the point values in the camera frame, we can easily calculate the conversion between pixels to units in the camera frame (in the "real world"). We also have the size of the unit blocks in the world frame, which is the same as their size in the camera frame because frames don't change size or shape. So we can do the conversion and see how many pixels the unit block would be picked up as by the camera.

Now that we have all the information we need, we want to create a representation of the shape. We build a data structure, a 2D array that is 1 anywhere there is a cube and 0 anywhere there is not a cube (Figure 4). This data structure is as small as possible, in the sense that there are no rows or columns that are all zeros. Along with this, each 1 represents one cube, so each cube sized region is represented by one bit. From there, we take the bottom left of the Tetromino and move up and right by half of the pixel side length of the unit cube. This gets us the middle of the first block. From there we move right by the pixel side length of the unit cube and put in a 1 every time we encounter a cube and a 0 when there's no cube. We move from bottom left to top right of the image, adding to our data structure from top left to bottom right. This just means that the first row of our data structure is the bottom of the Tetromino. This is good because we build from bottom to top.

Fig. 4. Each shape was converted into an array of 1's and 0's representing the reflection of the shape over the x-axis

C. Pick and Place

In order to pick and place each of the blocks in the correct location, the iiwa had to know where to begin and end. The locations of the array of building blocks were known to the iiwa and this list of poses was iterated through for every building block the iiwa required. The perception system described above outputted an array that acted as a grid to describe with a binary indicator whether or not a block existed in that location. That array was then reconstructed into a list of poses. Since the location of the building was arbitrary, we took in a $x_{initial}$ and $y_{initial}$ value as the bottom right corner of the shape that was being built. Next, for each value of "1" in the array, the position was calculated by equations below (1-2). The x value was always set to the same value of $x_{initial}$ because the reconstruction is 2D.

$$y = y_{init} + column * blocklength \tag{1}$$

$$z = z_{init} + row * blocklength \tag{2}$$

A series of gripper key frames were constructed and then interpolated to achieve the desired trajectories from the building materials zone to the construction zone. The list of key frames: "$initial$", "$prepick$", "$pick_{start}$", "$pick_{end}$", "$postpick$", "$clearance$", "$preplace$", "$place_{start}$", "$place_{end}$", "$postplace$", were calculated from

the current gripper position (equivalent to "$initial$") and the known block's initial and goal positions. The "$preplace$" and "$postplace$" locations were specifically chosen to be equivalent to the "$place_{start}$" in x and y while having larger z values. This allowed the gripper to center itself at the goal location aerially, while preventing collisions with any of the other built blocks during its path. Additionally, the gripper closing and was commanded with the "$pick_{end}$" and "$place_{start}$" key frames while the gripper opening was commanded for "$place_{end}$" and "$postplace$". Another important aspect of the trajectories was the orientation of the gripper as it placed the blocks. The gripper had to be orthogonal to the 2D building plane when placing the blocks to prevent collisions with other blocks in the construction zone.

To move the iiwa corresponding to the created trajectories, Drake's IiwaDifferentialIK was used. The inverse kinematics allowed the iiwa to take desired poses and convert them to joint angles that each link on the iiwa could achieve. Occasionally, the iiwa would reach the limits of its workspace, in which case the jacobian would being singular. In these instances, the locations of the blocks in the building materials zone or construction zone had to be relocated to an area the iiwa could reach.

The success of the reconstruction was also dependent on contact geometry of the modelled cube in the simulation. The final contact geometry was set as 8 spheres of radius $1e^{-3}$, one at each corner of the box. These spheres were inside the visual geometry of the block. Furthermore, a box inset from the visual geometry and from the spheres was used to capture most of the volume of the box, allowing proper contact to be modelled with another object if it did not come in contact with the spheres.

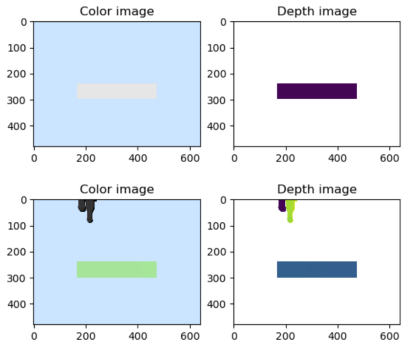

Fig. 5. Flat shape as the original (top) and as replicated by the iiwa (bottom). There are no visible qualitative differences.

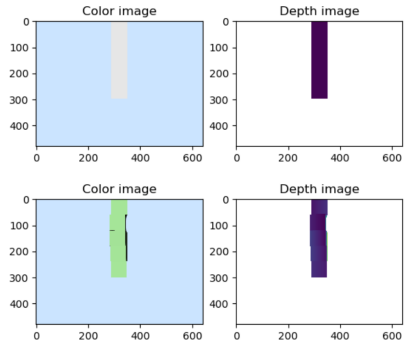

Fig. 6. i-block shape as the original (top) and as replicated by the iiwa (bottom). There are slight differences between the original and the replicated version. Some of the blocks are offset based on a qualitative analysis.

IV. Results

Depth values for the control image versus an image of the replicated shape were compared. Using depth values as the metric penalized displacements between the two images within the 2D plane of the shape with a greater weight than any forward-backward displacement. The reason the left-right displacement was weighted more was because it was more important that the 2D cross section resemble the original than any out of plane motion. The qualitative difference between the original shapes of "flat" and "i-block" and replicated counterparts are visible in Figure 5 and Figure 6. Figure 7 shows the error for each of the shapes after stacking another block.

The depth values were obtained by placing an RGB sensor camera at equal distance and same relative pose from both the shape to be replicated (the control) and the shape constructed by the robot. From there the system generated a color image and a depth image for each new block. For the depth images, if the camera did not "see" an object at a pixel location, it set the value to infinity (which was set to 0 in the code). A depth image was generated for every block in a given shape, so the i-block shape would have 5 depth images, and each image would have one more block than the last. For each of these depth images, the corresponding ground truth image was cropped to show the correct number of blocks. For example, in the case of the i-block, if the robot had only placed two blocks, the ground truth image would be cropped to only include 2 blocks. From there, the image taken mid-build of the replicated shape and the ground truth cropped shaped were subtracted from each other and the absolute value of that subtraction was summed up. Another metric we looked at was variance of the final replicated shape. If every block was in plane, this variance would be 0. These metrics were chosen because they measure deviation from the a ground truth, and since the objective of our project is successfully replicating an object, we want to minimize deviation.

Flat and i-block both use 5 blocks to construct their shape, yet the average depth error for i-block is significantly higher. As you can see in Figure 7, there is noticeable deviation in the depth values of the i-block. Compared to that, there no deviation visible to the human eye (Figure 5). We think that the difficulty in vertical stacking is partially due to an unoptimized physics engine. When stacking blocks one on top of the other, we deal with very difficult contact geometry. Since we place spheres at the vertices of each cube, to perfectly

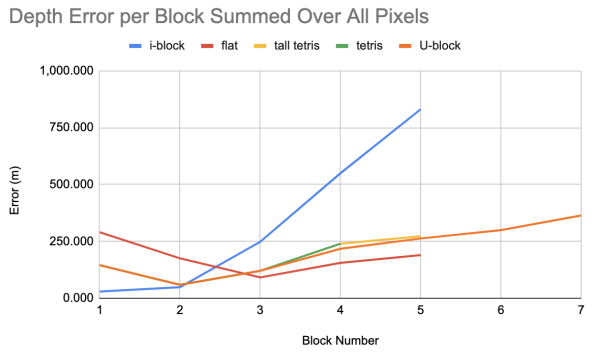

Fig. 7. Error measured as difference between ground truth depth and replicated object depth. The error was summed across all pixels in the depth image and was measured in meters.

stack these cubes on top of each other, the spheres must line up perfectly otherwise there will be slipping, as can be seen in Figure 6.

Another interesting thing to note is that when looking at the depth error for i-block, we first see an increase in error of approximately 20 between block 1 and block 2 and then a subsequent increase in error of around 200 between blocks 2 and 3. This jump can be qualitatively seen in Figure 14 in the Image Appendix, where the first visible sign of deviation is when the third block is placed. The deviation only continues getting worse, which can be seen in the steady upwards trajectory of the i-block error in Figure 7.

Looking at the graph, one surprising thing was that for tall tetris, the change in error between blocks 4 and 5 was less than the change in error between blocks 3 and 4. This is surprising because looking at Figure 14, there seems to be a larger change when placing the fifth block.

One point of note is that the visualization of the images from the depth sensor is relative to other values in the images, so the difference between the yellow fifth cube and the purple cube may not be as large as it visually seems. One thing that this may be attributed to is the way we get the 'ground truth' depth. Since we are cropping the image based on the known size of the cube in pixels, there could be compounding error between the cropping of the first layer and the second layer. Due to distances existing in the real number space in the world frame and integer space in the image frame, there may be some cropping error that causes the results to deviate from their true value. This will be expanded upon further in the Discussion section.

We can also take a look at the variance of the final replicated block (Figure 8). i-block and U-block have the two highest variances. Both i-block and U-block have 4 blocks that are not on the first level compared to the 1 of Tetris, 0 of flat, and 2 of tall Tetris. This would support our earlier hypothesis of stacking vertically leading to more deviation. One surprising point was tall Tetris having the lowest variance. We currently do not have a theory for this as we would expect tall Tetris to have a higher variance than normal Tetris, especially since the error increases.

Table 1: Average depth error over blocks (m)

block number	tetris	flat	i-block	U-block	tall tetris
1	145.397	290.327	29.217	145.397	145.397
2	59.318	175.405	48.556	59.318	59.318
3	120.450	91.955	248.098	120.450	120.450
4	239.698	155.541	549.575	217.523	239.698
5	272.783	189.600	831.083	262.730	272.783
6				299.584	
7				363.872	

Table 2: Variance of final replicated block (m)

tetris	flat	i-block	U-block	tall tetris
3.40E-07	6.70E-07	1.53E-05	2.02E-05	3.01E-07

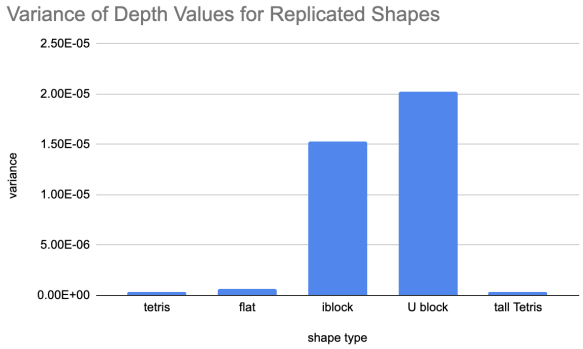

Fig. 8. Variance of depth values in the blocks that were recreated. Low variance means that blocks were closer to being in plane.

V. Discussion

After analyzing the results, one thing that repeatedly seemed to cause issues in the evaluation protocol was the way we generated ground truth images. The ground truth images were cropped from a fully constructed shape. Let's walk through why this could be an issue. Let's say we are trying to construct the tetris shape. Our camera is pointed directly at the center of the shape. When we place our first cube, the right cube, the camera will pick up on part of the side of the cube along with the front face. Due to this, even if the right cube is perfectly placed, there will still be error due to the fact that the camera captures the side of the cube as well as the front. Once you "get past" the middle cube and start placing to the left, this is no longer an issue since the face that would normally be pointing towards the camera is flush with the center cube. This is why, in Figure 5, we see the error start off high and then go down and reach a minimum when the middle cube is being placed.

There are several solutions to this. One possible solution is to directly keep track of the true pose of the blocks and quantify error based on that. This methodology can only work in simulation, as in the real world, there is no way to "get" the pose of an object. Another solution is that instead of cropping the ground truth, generate a separate ground truth for each block. If this is done, although the camera will pick up the

sides for the blocks we are using to construct a shape, it will also pick up the sides in the ground truth.

Additionally, we could have multiple cameras pointed at the building zone to improve the evaluation. For example, placing two cameras such that their lines of sight are perpendicular to each other would allow us to capture more complex details about the error, such as a more accurate capture of off plane deviation.

In terms of our original goal of replicating a shape, based on both our qualitative and quantitative results our system did well. Looking at the replicated figures in the Image Appendix, qualitatively it is clear what shape the robot was trying to replicate. Furthermore, the color and depth images also match the images of the shape we were trying to replicate by the human eye. Throughout our project, we had numerous iterations where the contact geometry caused the blocks and the robot to get stuck, or the blocks to tip over, or push the whole set of blocks around. None of these occurred in our iteration shown here. Along with this, our variance is quite low and even shapes with a large number of vertical blocks (U-block) had relatively low depth error (looking at the U-block line in Figure 5). Due to these factors, I would consider this system to have fulfilled the initial project objective, although improvements could definitely be made.

VI. Conclusion

A. Concluding Thoughts

A method of replicating 2D shapes with 3D blocks was devised using the Kuka iiwa arm and wsg gripper. With the system architecture, a vision subsystem evaluated the shape to replicate. A pick and place planner was able to use the initial location of the building materials and goal locations of the blocks to create a series of trajectories capable of stacking or placing the blocks in their locations.

Our system performs better on horizontal placing problem than it does on vertical stacking problems. In general, shapes that have a lot of vertical blocks have higher variance and more error. To evaluate this system's performance, depth values between the original shape and replicated shape were compared. The methodology of evaluation was found to possibly introduce error, as the camera gets data values for faces that should not be part of the evaluation. Qualitatively, our system performs its objective. Looking at the cross sectional views of the replicated figures, it is clear what shapes they are intended to be.

This work can be used in locations like warehouses where a different configuration of stacked items may be needed for different types of products. The robot arm would be able to assemble the items without colliding with other stacked items.

B. Next Steps

The first thing that would make this project more robust, is to add methods that allow the robot to recover and be robust to interference. After perceiving the Tetromino, the robot picks up a cube from a grid of blocks. Currently, the positions of the cubes are known to the robot, but if an adversary moves one of the cubes or if the robot itself knocks into one of the cubes, it subsequently will not know the location. The solution to this is to have a camera that points at the building materials zone and for each cube, the robot actively selects a cube to pick up based on the vision data. Along with this, we could include states in our state machine regarding successfully picking up a block form the building materials zone. This would result in a system that is more robust to both internal errors and external adversaries, since the data from the camera would let the robot know, in real time, where the cubes are. The next step is to move the cube from the pick location to the place location which is robust unless the robot is being told to reach outside its range onf motion. The final step of placing the cube is also not very robust. If the robot has an error in the placement of the a cube, the subsequent cubes have a high chance of being poorly placed due to the fact that there is no correction of the error. A next step to improve the robustness of the system is after each placement, to check if the cube is correctly placed. If not, plan a trajectory that re-picks up the cube and moves it to the correct position.

Another future improvement would be to have blocks of various different sizes and shapes. This would require some modifications to the system, but would mean that shapes could be generated more efficiently and with fewer cubes. The first modification would be to introduce a more complex decomposition algorithm. There is a lot of examples of polygon decomposition in literature, so we would have to implement one of those algorithms to get the decomposition of the Tetromino. We would also have to modify our data structure since the different blocks do not all occupy the same amount of space. Along with this, the system would have to have a camera pointing at the building material zone. From there, depending on the block the robot wants to pick up, the robot would have to adjust the grasp accordingly.

We can also increase the complexity of the shape we are trying to recreate. One way to do that would make the shapes more organic. This would require there to be a step between perception and placement pose creation where the image is pixelated and then converted into an array composed only of 0s and 1s. There would also have to be a step to ensure that this image is able to be constructed with blocks. For example, if there is an overhang, it would not be possible to make it "standing up" with blocks, because they would just fall down. An alternative way to make the shape more complex is for it to be a properly 3 dimensional shape. In order to do replicate this shape, you would have to have multiple cameras in order to capture all of the shape. There is also more complexity when it comes to the grasp and how the block is placed. When the shape is 3 dimensional, it requires making sure that the gripper does not interfere with the blocks placed around it.

Contributions

The perception portion of the system was completed by Shreya. She developed methods to acquire data from a given shape and decompose it into an array for replication and result validation.

The trajectory planning portion of the project was completed by Sharmi. This included creating the initial array of building blocks, as well as creating trajectories for the iiwa.

Both Shreya and Sharmi worked on contact modelling such as ensuring the contact geometry was being placed in the correct positions, and that they were stacking without tipping over and the setup of the simulation including creating the shapes to be replicated and the blocks for building.

External Research Overlap

Neither author has done prior research in this area nor is it related to projects being done for other classes or UROPs.

References

[1] A. Hundt, V. Jain, C.-H. Lin, C. Paxton, and G. D. Hager, "The Costar Block Stacking Dataset: Learning with Workspace Constraints," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.

[2] A. Hundt, B. Killeen, N. Greene, H. Wu, H. Kwon, C. Paxton, and G. D. Hager, "'Good robot!': Efficient reinforcement learning for multi-step visual tasks with SIM to real transfer," IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 6724–6731, 2020.

[3] I. SHARIF, D. CHAUDHURI, N. K. KUSHWAHA, A. SAMAL, and B. M. SINGH, "An efficient algorithm to decompose a compound rectilinear shape into simple rectilinear shapes," TURKISH JOURNAL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCES, vol. 26, pp. 150–161, 2018.

[4] R. Tedrake, Robotic Manipulation: Perception, Planning, and Control. Course Notes for MIT 6.4210, 2022. Available: http://manipulation.mit.edu

[5] J. B. Lanier, S. McAleer, and P. Baldi, "Curiosity-Driven Multi-Criteria Hindsight Experience Replay," 2019.

Image Appendix

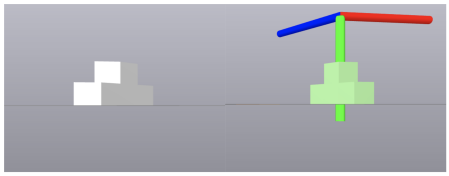

Fig. 9. Tetris: the shape to be replicated (left) and the shape created by the robot(right)

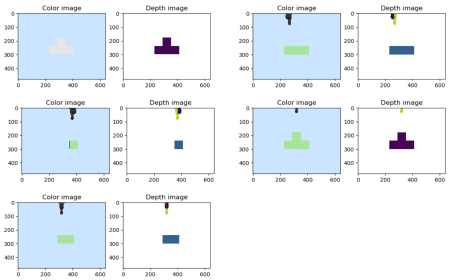

Fig. 10. Tetris: the shape to be replicated (top left) and the color and depth images obtained from the RGB sensor at each block placement

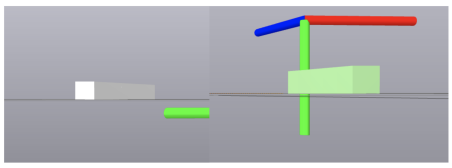

Fig. 11. Flat: the shape to be replicated (left) and the shape created by the robot (right)

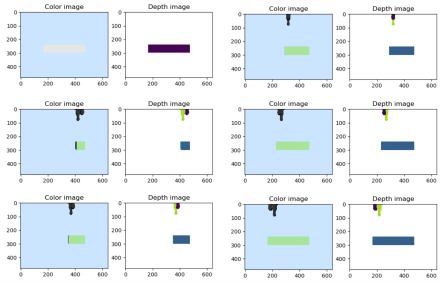

Fig. 12. Flat: the shape to be replicated (top left) and the color and depth images obtained from the RGB sensor at each block placement

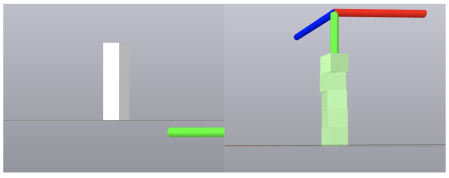

Fig. 13. i-block: the shape to be replicated (left) and the shape created by the robot right)

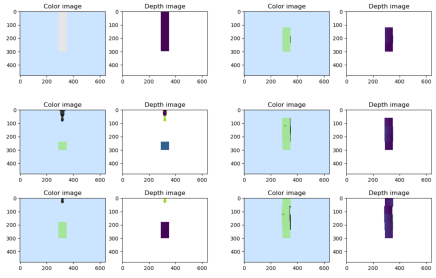

Fig. 14. i-block: the shape to be replicated (top left) and the color and depth images obtained from the RGB sensor at each block placement

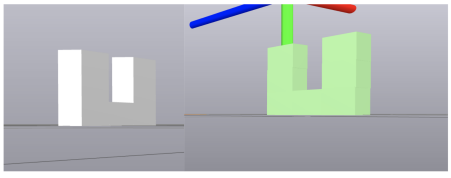

Fig. 15. U-block: the shape to be replicated (left) and the shape created by the robot (right)

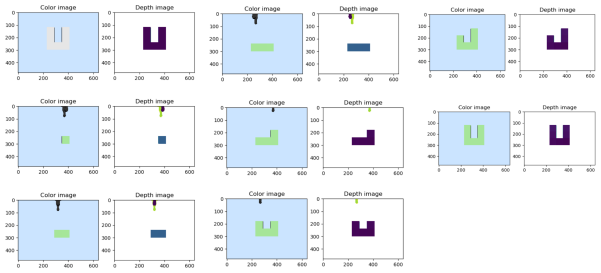

Fig. 16. U-block: the shape to be replicated (top left) and the color and depth images obtained from the RGB sensor at each block placement

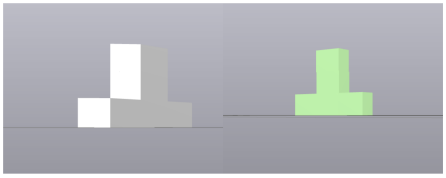

Fig. 17. Tall Tetris: the shape to be replicated (left) and the shape created by the robot (right)

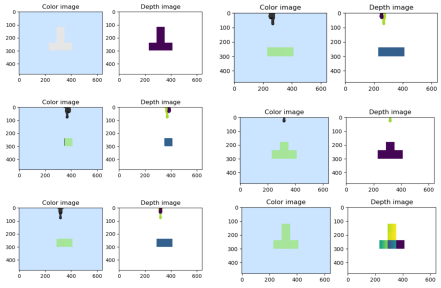

Fig. 18. Tall Tetris: the shape to be replicated (top left) and the color and depth images obtained from the RGB sensor at each block placement