Robot Autonomy: 16-662

Team 3 Final Report

Alex Pletta
Bruce Kim
Jaekyung Song
Benjamin Younes

Professor Oliver Kroemer

May 5th, 2022

Table of Contents

1. Motivation

The motivation of this project was to create a robotic system, in which a user requests a number of components and the robot would build a kit in minutes. A camera is used only for precise manipulation of the kit construction. Referring to the industry requirements of collaboration robots guided the team to focus on the properties of robustness, reliability and predictability. These are related to the key challenges faced in the manufacturing industry using a cobot. For instance, when a human operator is working with the cobot, the human body could cover the light and make the block detection difficult. Also, the manufacturing process sometimes fails to exclude foreign substances in the production line, and trigger errors in robotic manipulation. Lastly, industries demand a robot that can cope with various sizes and shapes of objects.

Figure 1: Robotic Pick and Place "Kitting"
(Source: Schubert "Vision System with Pick and Place")

2. Key Challenges

There were several key challenges to the project. Firstly, changing lighting and shadows made detection of blocks very difficult. Along with a vertical bar holding the camera casting a shadow on the workspace, we also had to deal with the robot arm itself casting a shadow. Blocks in lighting, shadow, or both made perception more difficult, thus making accurate and consistent detection of blocks challenging.

Additionally, we wanted to make the system robust to unexpected or undesired objects. We are concerned with red and blue blocks, so the system should ignore blocks of other colors, as well as ignore objects we don't care about in general. This again required perception to be robust and accurate so the system could focus on desired blocks, and disregard objects irrelevant to the kit.

Lastly, the size of the blocks are fairly small, both compared to the workspace as well as the arm. We therefore needed accurate and repeatable performance from the manipulation system.

Additionally, the blocks, though usually rectangular, may be connected to one another, resulting in unusual and unexpected shapes. We therefore must try to accommodate unusual sizes and shapes in the manipulation subsystem.

3. Overview of Approach

The system must move a specified number of colored bricks into a bin. To accomplish this, the system has three components: behavioral controller, perception and manipulation. The behavioral controller is the high level controller which takes in user input, such as how many of a certain number of blocks is desired. Then, the controller will call upon the perception and manipulation subsystems to fulfill the request. The perception system uses a bird's eye view camera to determine the number and color of blocks, as well as their location and orientation. This information is then used by the manipulation system, which moves the arm to the necessary location, grips the block, then moves to the bin, and then finally drops the block off into the bin.

4. Methods Implemented

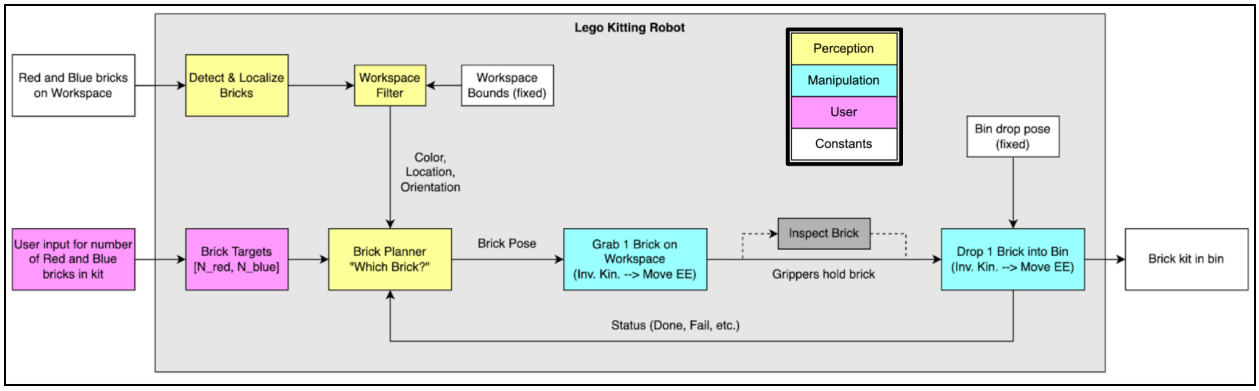

Figure 2: System Cyberphysical Architecture

When the user inputs the number of red and blue blocks, the camera detects and localizes the block. Then it filters out the blocks outside the workspace and feeds color, location, and orientation of the blocks to the Brick Planner. The planner decides which blocks to pick up and sends the brick pose to the manipulator. The manipulator then uses an inverse kinematics to move the end effector to the brick pose, grab the brick, and move the block to the bin. Then it drops the block into the bin, and repeats the entire process until the command is fulfilled.

4.1. Brick Planner

Input to the planner begins with a prompt to get a user request for how many blocks of each specific color should be placed into the kit. The planner was intentionally designed to be very simple to promote system consistency and predictability so users and/or cobot collaborators could feel safe working with the robot in a live setting.

4.1.1. Target Block Selection Logic

After receiving the number of red and blue blocks to include in the kit, the planner polls the perception subsystem for detected blocks. Once at least one red block is detected, the planner directs the arm to pick up the block furthest to the left of the workspace (as facing the robot; sorted by x position). Once a block is placed into the kit, then the planner re-polls the perception system. After all red blocks for the order are placed in the kit, then the planner takes blue blocks from the perception system and repeats this process until all blue blocks are in the kit to complete the order. If no red blocks are in the order then the planner skips directly to blue blocks.

At project completion, an inspection stage was not yet implemented for verifying the grabbed block was the correct color and/or that the block was successfully dropped in the kit. However, the detection and manipulation performance was observed to be fairly robust in a variety of test cases, so the final implementation of the planner assumed that all attempted block grabs were successfully dropped in the kit. Adding the inspection stage could of course make this process more robust, though in practice this assumption was very reasonable for system performance.

4.1.2. Trajectory Constraints for Simplified Obstacle Avoidance

The planner was also designed with somewhat strong assumptions about the workspace in order to simplify collision avoidance with other objects/obstacles in the workspace. These assumptions were rationalized on the basis that the robot would be operating in a somewhat controlled and well-structured environment, which was true for the duration of this project.

In particular, the key planner assumptions were:

1. No obstacles would be in a plane 20cm above the workspace
2. No obstacles would interfere with the manipulator moving vertically down onto the block poses
3. There would be enough room around each block for the gripper

These assumptions greatly simplify the trajectory planning logic, so that the manipulation subsystem can move freely between block poses and the kit bin above the 20cm plane and otherwise grab blocks by moving straight down and up without needing to perform online collision avoidance.

4.2. Perception

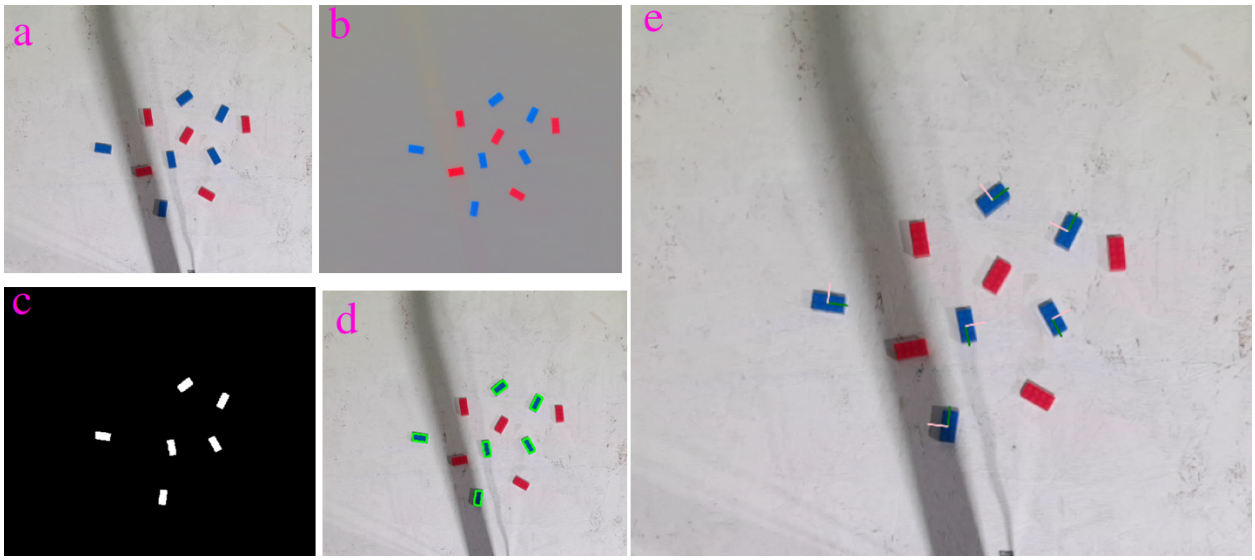

Figure 3: Perception Pipeline

The perception pipeline is done in 5 steps, as outlined below. The images above show the process for detection of blue blocks, but the process for red blocks is nearly identical.

a. The raw image is cropped. The raw image contains a lot of information that is not relevant to the task, such as the platform holding up the white workspace and often the feet of people near the robot. By cropping to only the workspace, we remove a lot of irrelevant information, reducing the amount of computation necessary as well as making the system more robust. Currently, cropping is hard-coded; we tell the system what rows and columns to keep. This is acceptable for now as the workspace
b. The image is normalized. In (a), we can see that there is a very prominent shadow in the center of the figure. This makes color segmentation difficult, as the shadow causes the RGB pixel values to decrease, while light causes the RGB pixel values to increase. Because of this, color segmentation is unreliable and performs poorly on the raw image. In order to get rid of the shadow, normalization is performed. We can think of each pixel as having an RGB value, but we can also think of each pixel as having a 3-dimensional vector in the RGB space. In this formulation, lighting will impact the magnitude of the vector, but not its direction. If we convert every vector in the image to unit vectors, that is we discard the magnitudes of each vector, then we effectively remove the impact of lighting, resulting in (b). The normalized image discards information we do not care about, but preserves all the information we need; this makes color segmentation accurate and robust.
c. Color segmentation is performed to generate a mask. The normalized image is now very easy to do color segmentation on. We can use the normalized image, (b), and see where pixels have a blue pixel value greater than some threshold; whether a pixel is above or below this threshold results in a mask. (c) shows white where the pixels are greater than the threshold, and black where they are below the threshold. This same process is repeated for red blocks, but in that case we care about the red pixel value. It should be

noted that red and blue segmentation is especially easy for the RGB images we use, as they are aligned with the axes of the RGB space. However, other colors can be segmented in a similar way. Rather than comparing the red or blue pixel value to a threshold, we can project the RGB value of each pixel in the normalized image to some color vector, and then see if the projected value is above or below some threshold. In our application, this projection is unnecessary since red and blue values are already values projected onto the RGB space axes.

d. Contours are generated using the mask. Generating contours on raw images is possible, but we found it to be inconsistent and often inaccurate. By providing the contouring with the mask rather than the raw image, we significantly increased how well the perception system works. (d) shows a green outline around the blue blocks since the mask was looking for blue pixels; the same process can be done to detect contours for red blocks. Contours are found using the *findContours* function in OpenCV2. However, the contours generated using this function create a polygon with many sides, often over 10; for our applications, we know the shapes we are looking for are rectangles, so most of these points are unnecessary. To simplify the contours, we use *minAreaRect* on the contours to generate contours with only four points, corresponding to the four corners of the blocks.
e. Finally, the location and orientation of the blocks are determined using the four point contours generated in (d). (e) shows a visualization of pose estimation by superimposing axes onto the center of the blocks, illustrating that the perception system is able to accurately find and determine the orientation of the blue blocks. Nearly the same process is done to estimate the poses of the red blocks. For each block, the x and y location of the center of the block is determined by taking an average of the x and y coordinates of the four point contour. Then, length and width of each block is determined. Here, length is defined to be larger than or equal to width, and it is parallel with the x-axis of the superimposed axes, shown in green in (e); width is the smaller dimension and is parallel with the y-axis, shown in pink. Once length and width are determined, trigonometry is used to determine the angle offset between the brick axes and the picture axes, thus determining the orientation of the block. This means the perception system has fully determined the pose of all blue blocks; the same process is repeated for all red blocks. The list of colors and poses are then passed to the manipulation system.

4.3. Manipulation

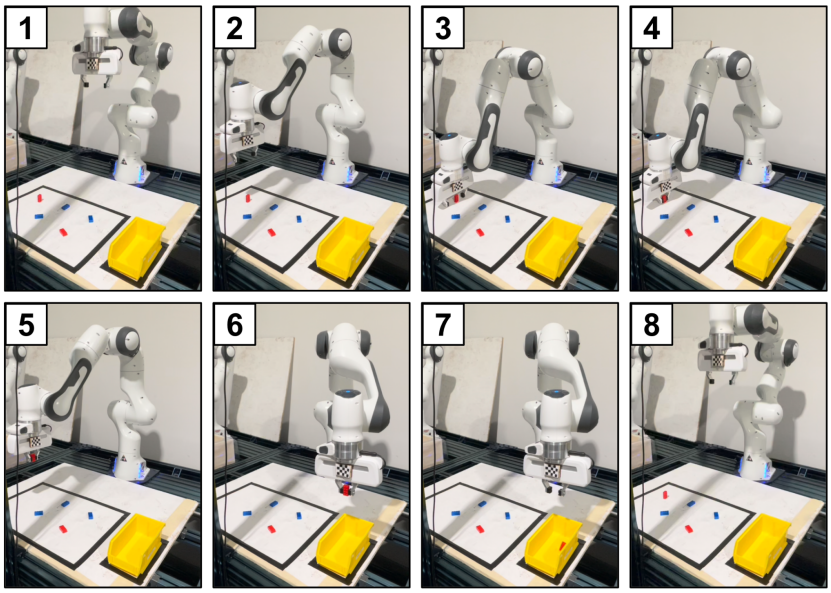

Figure 4: Manipulation Pipeline

The manipulation pipeline steps can be summarized as repeatedly cycling through steps 2-7 with poses from the planner until all blocks for the user order are placed into the kit. All steps are visualized for an example workspace in Figure 4.

1. Begin from home pose/joint configuration
2. Go to intermediate target pose
3. Go down to block target pose
4. Close gripper
5. Go up to intermediate target pose
6. Go to kit pose
7. Open gripper to drop block into kit
8. Return to home pose/joint configuration

In more detail, the robotic arm begins from home with the gripper fully open in order to start with known joint positions well within actuator limits. The robotic arm remains at home until the brick planner selects a target block. The brick planner then sends the associated target block pose to the robotic arm, but with a height at the 20cm work plane as an intermediate target pose. This intermediate target pose is directly above the block, and the robotic arm still rotates to the gripping angle as detected in the block pose. Going to this pose also clearly signifies to nearby operators what the robotic arm's target is before any block is grasped, so any interventions can be made if needed. The planner then gives the robotic arm the actual block pose, and the arm moves vertically down to the workspace plane. The arm slowly closes the gripper and uses force limits to make sure a safe amount of force is applied so that a grasped object could be dislodged easily without harm. Next the planner directs the arm to move vertically up to return to the intermediate pose. The arm then moves along the intermediate plane until the gripper is over the kit. Finally the gripper opens to drop the block into the kit waiting below. Instead of

returning home, the arm waits for the planner to give a new intermediate target pose to pick up a new block. While returning home could help reset the joint configuration, the system was observed to operate reliably without needing to reset joints so going directly to the next intermediate pose improved kit throughput. The arm repeatedly cycles through grabbing blocks from the workspace and dropping them in the kit until the planner decides the order is complete. As a final step, the arm is instructed to return to home and fully reset the joints again to conclude that kit and prepare for any next order(s).

5. Demonstration and Evaluation

The demonstration performed by the system, documented in the video at the end of this report, starts with trivial kitting tasks and gradually becomes more intricate as tasks progress.

Demonstration 1: Simple Demo

The first demonstration involves the trivial case of X red bricks, Y blue bricks kitting on a decluttered worksite. All of the bricks in the first demonstration are the same size, are in the same orientation with a different yaw, and are a minimum one end-effector distance from each other. This was seen as the “hello world” of lego kitting. This demonstration was very successful.

Demonstration 2: New Shapes and Sizes

Once the simple demo was demonstrated, the team moved on to demonstrating multiple sizes and orientations. This second demonstration still maintained the assumption that all bricks were the correct two colors. This demonstration was generally successful.

Demonstration 3: Cluttered Worksite and Novel Color

Following demonstration #2, the team shows the systems performance with multiple sizes, colors, and shapes in a very tight cluttered environment, stress testing the color segmentation algorithms. This demonstration was generally successful.

Demonstration 4: Cobot Demo

The final demonstration shows how the system could be used as a lego kitting cobot. A user was building legos and placing them onto the worksite such that they could be kitted. This demonstration was very successful and was actually usable.

6. Future Work

One identified aspect of this work that would make the packaged system more robust would be the addition of an inspection step to verify a grasped object OR a kit inspection step to verify that the correct object made it into the kit. At the moment the team assumes that once the perception system identifies an object, it makes it end-to-end into the bin. There were cases where the object would; not be grasped, break up upon grasp, not make it into the bin during drop, get dropped mid movement, etc. Another future work could be to introduce a more dynamic way of identifying blocks with something like a CNN or another training based approach, possibly using the depth camera, which was unused during this project.

7. Demonstration Video

The below video link demonstrates the system in action and reflects the final performance.

https://www.youtube.com/watch?v=qngiJ5iDRoQ