ME/EE/CS 134

Team 7: SpongeRob - BotPants

Skye Reese, Alexander Bouman, Sasha Bodrova, Bianca Yang

Final Report

Baby Bot

The Block - Stacking Robot

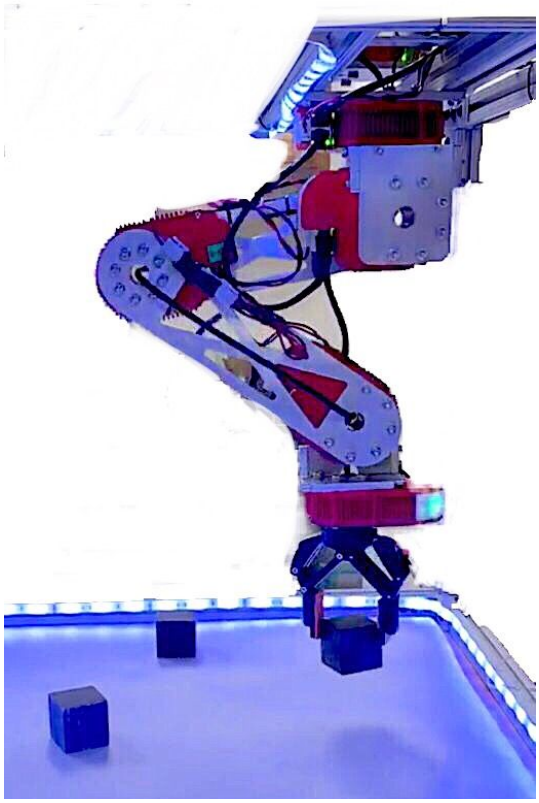

Figure 1. Baby Bot in action.

1. General Overview

Our robot is a 5DOF robotics arm which locates and stacks 1”x1” painted wooden cubes. Once the first six have been stacked into pyramid, the arm continues to stack onto the top block. There is one motor per DOF and four linkages. We have an additional motor for opening and closing the gripper. The arm is mounted to an aluminum frame that is clamped to a table. The power block is mounted to the top of the frame. We have also mounted a web camera to the frame which sees the blocks from a bird’s eye view. The system is lit by a LED strip that spans the inside of the whole frame. The blocks can be placed within a 19.3cm radius, measured out from the center of the first motor, which is mounted to the frame. We covered the table with a white paper to provide greater contrast for the camera. The motivation and objective was to teach a robot to stack blocks faster than a baby would learn to perform the same task.

Our BabyBot follows the following high-level steps:

1. Once the robot is switched on, the arm moves to a home position which is out of the camera's field of view.
2. The camera detects the centers of the blocks, converts those centers from pixels to cartesian coordinates in the world frame, and sends those converted centers to the "brain".
3. The arm moves to the position of the first block, which is arbitrarily chosen by our detection algorithm. The movement and final configuration of the arm is determined by our inverse kinematics calculations.
4. The arm slowly lowers the gripper around the cube. To ensure that the gripper closes on a flat face of the block, we have the gripper close partly and then rotate by π radians. The gripper closes fully and the cube is transported to just above its "stacking position" in the corner of the frame.
5. Before the cube is released, the gripper orients the cube, lowers itself, and then releases the block gently.
6. The arm returns to the home position. Steps 1-6 are repeated until the camera no longer detects any more cubes. The stacking position is adjusted on each iteration as described below.

Details of each component are discussed below. On the software side, we used ROS and C++. Any reference to nodes are reference to ROS nodes. On the hardware side, we used a variety of Hebi Robotics' X-Series Actuators.

2. Detector and Integration with Arm Movement

A Logitech web camera with a frame rate of 10 Hz is mounted on top of the aluminum frame. This camera has a bird's eye view of the floor area and any cubes that are in the work space. The camera has been calibrated using monocular calibration to account for any radial distortion. The camera is positioned just above the center of the workspace. We used an open CV-based shape detection algorithm from pyimagesearch.com to detect the blocks. The algorithm computes the coordinates of all the blocks and sends the position of one block each time the arm is ready to pick up another cube. We also used openCV's minAreaRect to extract the orientation of the block in degrees.

In order to have the camera interface with the arm, we had to convert from pixels and camera coordinates to centimeters and robot frame coordinates. We determined conversion by placing a couple blocks down, running the shape detector to get their centers in pixels, and then making some measurements with a measuring tape. These transformed coordinates are sent from the camera node to the brain node, which then tells our inverse kinematics node to move the arm to specific locations in the workspace.

The algorithm first processes the image by applying a Gaussian Blur to smooth out roughness then thresholds the blurred image to get basic contours. We determine which contours are cubes by fitting polygons to the contours and selecting for those with the right number of sides and appropriate sizes. To

detect the orientation, we used the minAreaRect function, which fits a neater cube than boundingRect, which only fits rectangles with sides aligned to the axes of the camera.

The accuracy of the detection algorithm is high, especially after applying the size thresholds, but the orientation of the cubes is unreliable. That is why we had the gripper rotate through π before it fully closes on a cube.

3. Mechanism overview

Figure 2. Solidworks model of the Baby Bot.

Since block stacking requires a high degree of precision, we chose aluminum for the material of our frame and linkages to reduce any effects from bending or oscillations. The frame is made out of 1'' thick aluminum t-slots that were cut to size on a band saw. The frame size is $0.91m \times 0.61\ m\ \times 0.50\ m$. The $0.91m$ horizontal bars on the bottom of the frame are extended so we can clamp them to the table. The enclosed floor space is $0.55\ m\ \times 0.55\ m$. The area seen by the camera is $0.36\ m\ \times 0.42\ m$ with an offset of $0.08\ m\ \times 0.07\ m$ from the corner of the frame.

The linkages were designed in solidworks and waterjetted from ¼'' Aluminum. The linkage from the base (first) motor to the shoulder (second) motor is 0.06 *m* long. The linkage from the shoulder (second) motor ro the elbow (third) motor is 0.2 *m* long. The linkage from the elbow (third) motor to the wrist vertical (fourth) is 0.2 *m* long. The linkage from the wrist vertical (fourth) motor to the wrist horizontal (fifth) motor is 0.04 *m* long. The length of the arm fully extended is 56 *cm*. The length of the arm when fully contracted is 30 *cm*.

We chose the arm to extend down from the top of the structure rather than out from the side because it allowed for a larger workspace. This configuration also reduced the amount of torque the base,

or shoulder motor would experience. We did need to make sure to avoid running through the singularity right under the robot when moving, but this was a fairly minor consideration.

As mentioned before, we are using six motors:

0. Gripper motor - requires a small amount of torque to pull on a cable to open and close the gripper. We used a X5-9 motor.
1. Base motor - Is in the yaw configuration, requires 4.8 nm of torque to hold the full weight of the arm, so we used X5-9 motor because these have max torque capability > 4.8 Nm..
2. Shoulder motor - Is in the pitch configuration, requires 4.8 Nm of torque too since it has to lift up the full arm, so we used X8-16 motor because these have max torque capability > 4.8 Nm.
3. Elbow motor - Is in pitch configuration, requires 2.14 Nm of torque, so we used X5-9 motor pitch because these have max torque capability > 2.14 Nm.
4. Wrist vertical motor - Is in pitch configuration, requires 0.43 Nm of torque, so we used X5-4 motor, which has max torque > 0.43 Nm. 0.43 Nm is a small amount of torque so we were able to use a weaker motor.
5. Wrist horizontal motor - Is in yaw configuration for elbow-up configuration stacking, and roll configuration for elbow-down configuration stacking, requires a negligibly small amount of torque, so we used X5-4 motor.

We commanded the arm to move at a relative speed of 0.02 m/s, which was sufficiently quick for the relatively small workspace.

The kinematics for the arm movement are based on the 5 DOF and keeping the gripper link either perpendicular (for elbow-up design) or parallel (for elbow-down design) to the ground. The elbow-up kinematics are used for the lower rows of blocks. The elbow-down kinematics are used for the top block and any block above that. The elbow-down kinematics are necessary for stacking the higher blocks because the joints of the arm hit the top rim of the frame when the gripper is raised above 10cm.

We tuned the gains on the blocks using an approach similar to the one described here, on StackExchange. We first tuned the P gain on position until the robot oscillated when disturbed. To control oscillations, we increased the derivative, or velocity P gain. To increase accuracy and convergence towards a desired position within a desired time, we increased integral gain.

4. SW Architecture

We have four nodes: the Brain node, the Camera node, the Gripper node, and the IKin node..

1. Camera node: Detects and locates blocks, publishes centerpoints and orientation angle to brain node.
2. Brain node: Subscribes to information on centerpoints, contains the logic for generating a trajectory for picking up and stacking blocks, publishes orientation angle and opening or closing command to gripper.
3. Gripper node: Subscribes to angle and opening/closing command from brain node, rotates gripper, opens and closes the gripper.

4. IKin node: Subscribes to tip position commands from brain node, computes kinematics and moves each actuator.

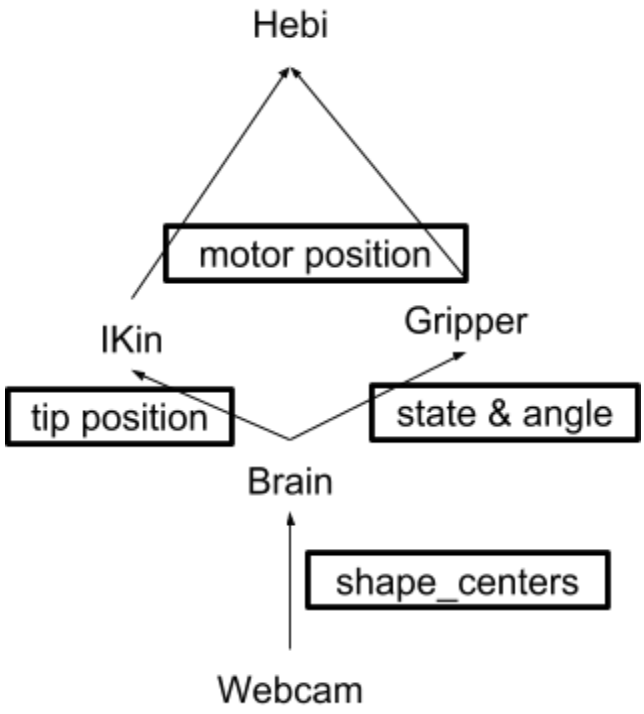

Figure 3. ROS Node Architecture

5. Challenges Faced

1. In order to prevent the camera from detecting non-block objects, like the border of the frame, we added an upper and lower limit to the size of bounding boxes it would consider valid.
2. The camera had difficulty detecting blocks that did have high contrast against the white background. Thus, we colored all blocks black, blue, and purple
3. The original elbow down configuration could not reach above the height of two blocks. To resolve this, we developed the inverse kinematics for an elbow down configuration which allowed us to stack to 3 rows and beyond.
4. We had many challenges debugging the iKin and wrote a forward kinematics function to verify our work.
5. Improperly tuned gains would result in motor oscillation during debugging. There were also cases where the virtual machine could not interface with the robot at a fast enough refresh rate to allow for smooth movement.
6. Originally, the gripper node was having a difficult time correctly identifying the block's angle of orientation. We resolved this issue by rotating the block back and forth while the gripper loosely held the block, using the torque applied on the block to rotate it into a position where the gripper

would close onto two faces rather than two points. The gripper motor malfunctioned at the very end of the project and could no longer be zeroed.

7. We had to avoid the singularity, which is the vertical axis which extends through the center of the first /base motor. We did this by by setting the arm's home position to the side of the space, which automatically prevented the construction of almost all straight line trajectories which would lead the arm through the singularity. We also placed the camera in a position where blocks that would cause the arm to move through the singularity could not be seen.

6. Final Remarks

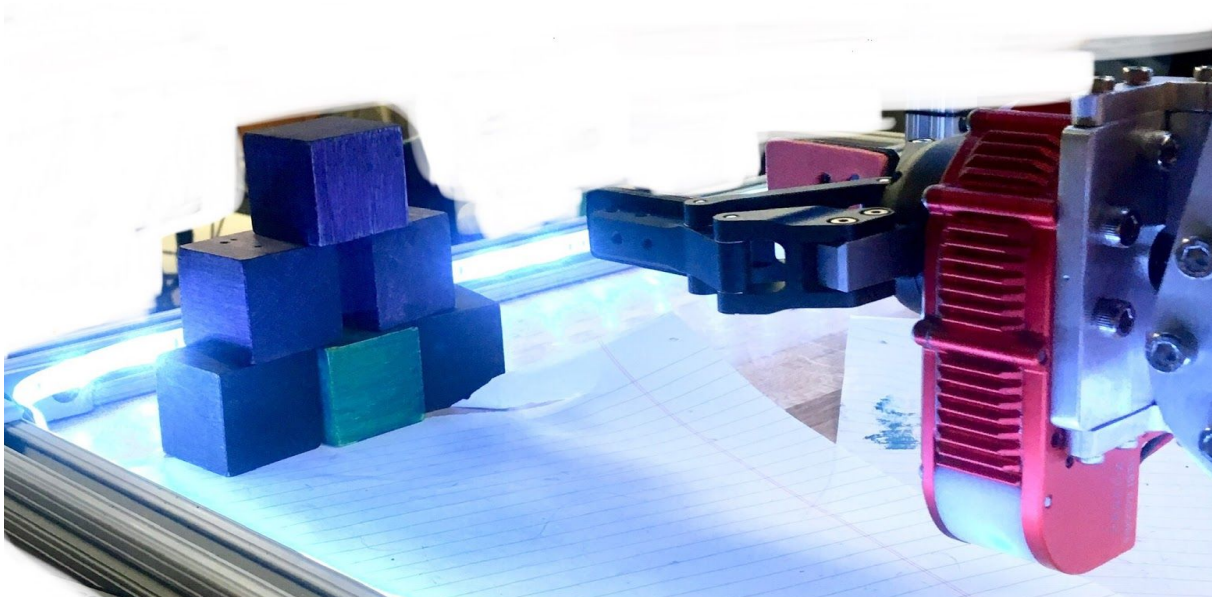

Figure 4. Baby Bot stacked a pyramid.

Our robot was able to successfully stack a full pyramid during our tests (Fig. 3). However, because of the troubles with the gripper node, it did not perform as expected during the presentation. The gripper motor had what seemed like a firmware issue where it would rotate in the wrong direction, so it would never reach its goal. We had to decommission that motor and the arm after that malfunction.

Overall we are happy with the results, as we taught a robot to stack blocks in less than five weeks.

Some future steps we were considering were to integrate all the code into a package that could be run on a Raspberry Pi. This would allow the robot to exist as a standalone unit that you could run by pressing some buttons on a touchscreen attached to the Pi. We also considered improving the detection node so it could "see" more colors, recognize letters and create meaningful letter sequences, and unstack the pyramid.