Armlab Robot Control Report

Jinze Liu, Sibo Wang, Yulun Zhuang
{jzliu, sibo, yulunz}@umich.edu

***Abstract*—This report focuses on the robot control aspect of the Armlab project. The forward kinematics is implemented using product of exponential method. The inverse kinematics is then solved analytically by decomposing the set of joint variables. The path planning generate a list of joint angles for the arm when picking or placing blocks. In the state machine, the logic of the competition events are implemented.**

I. Forward Kinematics Solution

Forward kinematics (FK) focuses on finding the end effector position in the workspace as a function of the state in configuration space. In this project, it is manly used for verifying the results from inverse kinematics (IK). Specifically, the product of exponential method is used. The structure equation is

$$T(q) = e^{[\mathbf{S}_1]\theta_1} e^{[\mathbf{S}_2]\theta_2} \dots e^{[\mathbf{S}_n]\theta_n} M$$

where T is the end effector state, $q = [\theta_1, \theta_2, \dots, \theta_n]^T$ is the joint angles state, $e^{[\mathbf{S}_i]\theta_i}$ is the matrix exponential for a given θ_i, and M is the home position [1].

A. Home Position

The home position is the state of the end effector where all the joint variables θ_i are defined as being zero. The arm's "Initialize" state, depicted in Figure 1 below, is selected to be the home position.

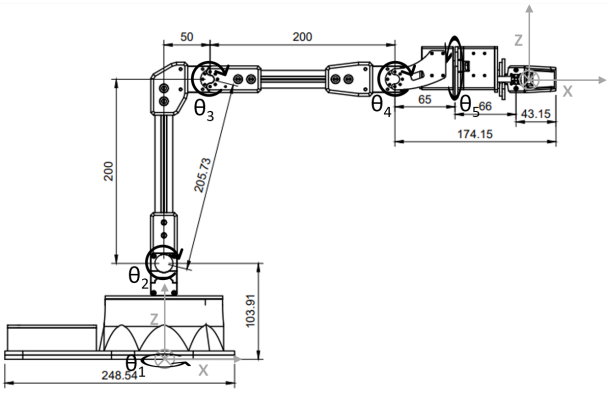


Fig. 1: Configuration of home position [2]

The above configuration is given by the manufacturer Interbotics[2]. Based on this configuration, the home position matrix M that describes the transformation from the base frame to the end effector frame is

$$M = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 408.575 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 304.57 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\mathbf{R}$ is the 3×3 rotation matrix and $\mathbf{p}$ is the 3×1 translation vector.

B. Matrix exponential for each joint

A 6×1 screw vector, $\mathbf{S}_i$, is composed to describe the movement of each joint,

$$\mathbf{S}_i = \begin{bmatrix} \mathbf{S}_{\boldsymbol{\omega} i} \\ \mathbf{S}_{\mathbf{v} i} \end{bmatrix} = \begin{bmatrix} \text{angular velocity when } \dot{\theta} = 1 \\ \text{linear vel. of origin when } \dot{\theta} = 1 \end{bmatrix}$$

Taking θ_2 in Figure 1 as an example, since the joint is rotating around the y^+ axis, $\mathbf{S}_{\boldsymbol{\omega} i} = [0, 1, 0]^T$. The linear velocity of origin due to this rotation is in the x^- direction, with a magnitude of 103.91mm, thus $\mathbf{S}_{\mathbf{v} i} = [-104.57, 0, 0]^T$. The latter magnitude is taken directly from the screw vector matrix provided by Interbotics and has small discrepancies with the dimensions on the engineering drawing [2]. After confirming with actual measurements, the latter is taken as the truth. The complete screw vector matrix is shown below, with each row as the transpose of each screw vector.

$$\mathbf{S}_{list} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -104.57 & 0 & 0 \\ 0 & 1 & 0 & -304.57 & 0 & 50 \\ 0 & 1 & 0 & -304.57 & 0 & 250 \\ 1 & 0 & 0 & 0 & 304.57 & 0 \end{bmatrix}$$

With $\mathbf{S}_{\boldsymbol{\omega}} = [\omega_1, \omega_2, \omega_3]^T$ and $\mathbf{S}_{\mathbf{v}} = [v_1, v_2, v_3]^T$, the matrix form of $\mathbf{S}_i$ is given by

$$[\mathbf{S}_i] = \begin{bmatrix} [\boldsymbol{\omega}] & \mathbf{v} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 1 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Then, the matrix exponential for each joint can be found by $\exp([\mathbf{S}_i]\theta_i)$, which could be solved analytically by the Rodrigues' Formula presented in lecture [3].

With the above components and the structure eqution, for a given state of configuration defined by joint angles $\theta_1, \ldots, \theta_5$, the homogeneous transformation matrix T for the end effector could be found, which expresses the end effector's position and orientation in the world frame.

II. Inverse Kinematics Solution

Inverse kinematics (IK) finds the state of configuration as a function of the end effector position. The 3D problem is first projected into a 2D problem. Then, it is further decomposed into the wrist section and the shoulder-elbow RR arm.

A. Projection to 2D problem

The input end effector pose is given by (x, y, z, ϕ), which is the Cartesian world coordinate and the desired rotation around X according to the (Z, X, Z) rotation configuration.

Since the waist joint, θ_1, solely governs the rotation around z, it can be computed independently. The remaining problem is project to 2D space, with the new frame (r, z'), as illustrated in Figures

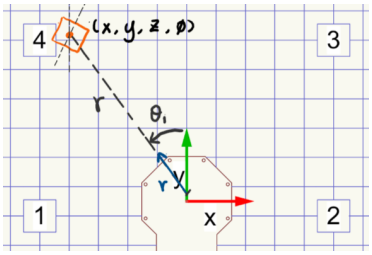


Fig. 2: Top-down view of θ_1 and r

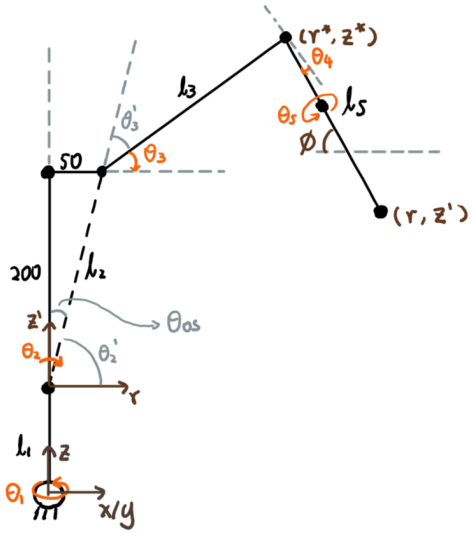


Fig. 3: Projected 2D scheme

As illustrated in Figures 2 and 3, $\theta_1 = \arctan(-x, y)$

$$r = \sqrt{x^2 + y^2}$$
$$z' = z - l_1 = z - 104.57$$

Note that $\theta_1 = \pi + \arctan(-x, y)$ is also a valid solution. In this case, only the solution with the smaller absolute angle is considered.

B. Wrist position and ϕ-angle

As shown in Figure 3, if wrist position (r^*, z^*) is known, then θ_2 and θ_3 simplify to an RR arm problem with some geometric transformations. Note that wrist only has two in-plane d.o.f. and the desired ϕ is exactly the angle between the r-axis and the wrist arm. Thus,

$$r^* = r - l_5 \sin\phi$$
$$z^* = z' - l_5 \cos\phi$$

C. Shoulder-elbow RR arm

Firstly, use the 2-revolute joint (RR) arm equations to find θ_2' and θ_3' [4],

$$\theta_3' = \arccos\left(\frac{r^{*2} + z^{*2} - l_2^2 - l_3^2}{2 l_2 l^3}\right)$$
$$\theta_2' = \arctan(z^*, r^*) - \arctan(l_3 \sin\theta_3', l_2 + l_3 \cos\theta_3')$$

where $l_2 = \sqrt{200^2 + 50^2}$, $l_3 = 200$

Note that there also exist a solution with $\theta_3' = -\theta_3'$. Due to the nature of the required workspace range and arm's mechanical constraints, the elbow-up solution presented above is always used.

Then, based on geometry,

$$\theta_2 = \pi - \theta_{OS} - \theta_2'$$
$$\theta_3 = -\pi + \theta_{OS} - \theta_3'$$
$$\theta_4 = \phi - (\theta_2 + \theta_3)$$

where the offset angle $\theta_{OS} = \arctan(50, 200)$.

D. Block orientation and θ_5

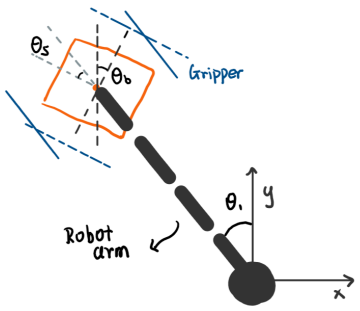


Fig. 4: Top-down view for geometric aspects of θ_5

θ_5 is only within concern when trying to vertically pick and place blocks at specific orientations. As shown in Figure 4, $\theta_5 = \theta_1 - \theta_b$

where the block orientation $\theta_b \in (0, 90]$ is the angle between the y-axis and the closest edge of the block. To ensure minimal rotation, θ_5 is optimized to be

$$\theta_5 = \begin{cases} \theta_5 - \frac{\pi}{2} & \text{if } \theta_5 > \frac{\pi}{4} \\ \theta_5 + \frac{\pi}{2} & \text{if } \theta_5 < -\frac{\pi}{4} \end{cases}$$

E. Corrections for gravity load down

Load down due to gravity has been discovered to cause significant discrepancies, especially for locations far in r-direction, as shown in Figure 5.

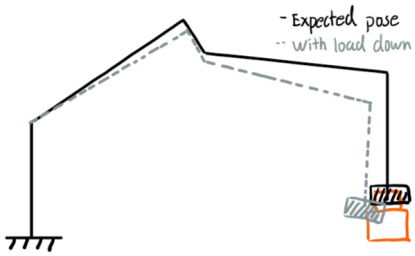


Fig. 5: (r, z) frame view of effect of load down due to gravity

To minimize the effect of the load down, firstly, θ_5 is further optimized to enforce the grippers are parallel to the longitudinal direction, not the radial direction. As shown in Figure 6, in this configuration, the discrepancy in radial location due to load down would not cause the grip to be off-centered.

$$\theta_5 = \begin{cases} \theta_5 - \frac{\pi}{2} & \text{if } \theta_5 > 0 \\ \theta_5 + \frac{\pi}{2} & \text{else-wise} \end{cases}$$

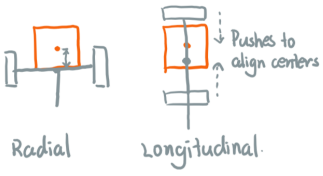


Fig. 6: Radial vs. longitudinal pick

Furthermore, with trials and errors, the discrepancy in the z-direction is approximated to be linear with $r = \sqrt{x^2 + y^2}$. Therefore, a linear offset in the z-direction is added when picking and placing blocks vertically in the range $r \in [315, 430]$,

$$z := z + \frac{1}{55}\sqrt{x^2 + y^2}$$

III. Path Planning

When we use the ReactorX-200 Robot Arm to pick, place blocks or even finish more complex tasks, path planning is used to generate a list of joint angles making sure the process is safe. In this section, we will explain how we generate the list of joint angles in order to pick or place blocks accurately without colliding other blocks.

A. Teach and Repeat

After teaching the arm to swap blocks at locations $(-100, 225)$ and $(100, 225)$ through an intermediate location at $(250, 75)$, our arm can cycle more than 10 times.

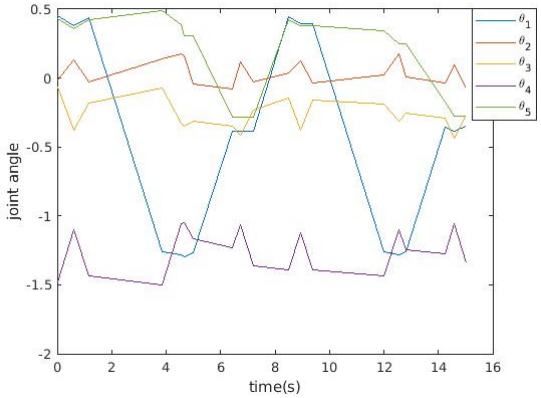


Fig. 7: Joint angles used to swapping blocks at locations $(-100, 225)$ and $(100, 225)$ through an intermediate location at $(250, 75)$ over time for one cycle. The joint angles are got from the teaching process.

B. Click to Pick

Algorithm 1: ClickPick(u, v)/ClickPlace(u, v)

Data: u, v: Clicked pixel
$p_{target} \leftarrow$ GetWorldCoordinate(u, v);
$\alpha \leftarrow 0$;
$p_{block}, \alpha_{block} \leftarrow$ GetNearestBlock(p_{target});
if *dist*$(p_{target}, p_{block}) < d_{click}$ **then**
  $p_{target} \leftarrow p_{block}$;
  $\alpha \leftarrow \alpha_{block}$;
end
GoToHomePose();
Pick$(p_{target}, \alpha_{target})$/Place$(p_{target}, \alpha_{target})$

In the Click to Pick function Algorithm 1, we have the clicked pixel (u, v). We can transform it to the world coordinate p_{target}, which is explained in our CV report. Then, we can get the nearest block position p_{block} and orientation α_{block}, which is also explained in our CV report. If the distance between clicked position in world frame p_{target} and the block position p_{block} is smaller than the click distance threshold d_{click}, we assume that

Algorithm 2: Pick($p_{target}, \alpha_{target}$)

Data: $p_{target}, \alpha_{target}$
$p_{target} \leftarrow p_{target} + p_{pick,off}$;
$p_{above} \leftarrow p_{target} + p_{above,off}$;
MakeReachable($p_{target}, \phi_{target}, p_{above}, \phi_{above}$);
$\theta_{target} \leftarrow$IK($p_{target}, \phi_{target}$);
$\theta_{above} \leftarrow$IK($p_{above}, \phi_{above}$);
SetJointAngle(θ_1);
SetJointAngle(θ_{above});
SetJointAngle(θ_{target});
CloseGripper();
SetJointAngle(θ_{above});
SetJointAngle(θ_{safe});

the user is going to pick that block and update p_{target}. In our experiments, we set d_{click} to be $5cm$. Finally, we set the arm to home position and pick the block at p_{target} with the orientation α_{block}.

In the Pick function Algorithm 2, we define two positions in world coordinate, p_{target} and p_{above}. In Pick function, p_{target} is the position of end effector when the arm is grabbing the block, which is generated by adding the pick offset $p_{pick,off} = (0, 0, 29)$. p_{above} is the position of end effector when the arm is above the block and preparing to pick the block, which is generated by adding the above offset $p_{above,off} = (0, 0, 80)$.

Then, we make two positions p_{target}, p_{above} and ϕ explained in IK reachable with IK calculation. If the block can be picked vertically, i.e. when $\phi_{target} = \pi/2$, we will pick the block vertically. If p_{target} is unreachable when $\phi_{target} = \pi/2$, we will pick the block horizontally, i.e. set $\phi_{target} = 0$. p_{above} is designed for picking and placing safety. If p_{above} is unreachable, we first lower the above position step by step until the distance between p_{target} and p_{above} will be smaller than the safe threshold d_{safe}, which is set to be $40cm$ in our code. If p_{above} is still unreachable, we will make the value of ϕ_{above} smaller step by step until 0, where ϕ_{step} is set to be $\pi/18$ in our experiments. In summary, in MakeReachable($p_{target}, \phi_{target}, p_{above}, \phi_{above}$) function, we make target position p_{target} easier to reach by setting $\phi_{target} = 0$. And we make above position easier to reach by lowering p_{above} and making ϕ_{above} smaller. If either position is still unreachable, the pick process will fail.

After that, we use IK to calculate the joint angles for target position p_{target} and above position p_{above}. We assume that the arm is at a safe sate with waist rotation at the beginning of pick and place. To make the assumption valid, we will move the arm to a safe state at the end of pick and place. We first rotate the waist to θ_1 to avoid the arm colliding with blocks. Then we set the joint angles to be θ_{above}, in order to move the gripper to the position above the block. After that, we set joint angles with θ_{target} to make gripper ready to pick the block. Then we close the gripper and move arm back to the above state by setting joint angles with θ_{above}. Finally, we set joint angle to be θ_{safe} in order to make the arm ready for the next pick or place.

C. Click to Place

Algorithm 3: Place($p_{target}, \alpha_{target}$)

Data: $p_{target}, \alpha_{target}$
$p_{target} \leftarrow p_{target} + p_{place,off}$;
$p_{above} \leftarrow p_{target} + p_{above,off}$;
MakeReachable($p_{target}, \phi_{target}, p_{above}, \phi_{above}$);
$\theta_{target} \leftarrow$IK($p_{target}, \phi_{target}$);
$\theta_{above} \leftarrow$IK($p_{above}, \phi_{above}$);
SetJointAngle(θ_1);
SetJointAngle(θ_{above});
$\Delta\theta \leftarrow (\theta_{target} - \theta_{above})/2$;
$\theta_{lower} \leftarrow \theta_{above}$;
while *not touch* **do**
  $\theta_{lower} \leftarrow \theta_{lower} + \Delta\theta$;
  SetJointAngle(θ_{lower});
  $\Delta\theta \leftarrow \Delta\theta/2$
end
OpenGripper();
SetJointAngle(θ_{above});
SetJointAngle(θ_{safe});

The Click to Place function is almost the same as the Click to Pick function. The only difference is using Place function or Pick function. In Pick function Algorithm 3, the calculation of p_{target} is difference. We add $p_{place,off} = (0, 0, 33)$ to p_{target}. The generation of θ_{target} and θ_{above} are the same of Algorithm 2.

To place the block, similarly, we first rotate the waist to θ_1 and set the joint angles to be θ_{above}. When we move the arm end effector from above position to the target position, we use the binary search algorithm to lower the arm step by step and stop until the block touches ground. At the beginning of lowering, the block is unlikely to touch ground, so we can let the move larger. With the block lowering, the block becomes easier and easier to touch ground, therefore, we make the movement smaller and smaller using binary algorithm. We calculate the norm of effort difference of the second and third joints. When the norm is larger than 100, we will think that the block touches ground and stop. After that, we open the gripper, move back to the above position and move back to the safe position, which is similar to the Pick process.

IV. State Machine

In this section, we will explain the logic of competition events.

A. Event 1: Pick 'n sort!

In the first event, we begin with detecting the blocks on the positive half plane and sorting them by the distance to $(0, 0)$ in x-y plane. Every time, we pick the closest the block and place it to the negative half plane. If it is a big block, we will place it to p_{big} and update p_{big}. If it is small big, we will place it to p_{small} and update p_{small}. Every time, after placing one block, we will detect the blocks on the positive half plane again. We define the process that picking blocks on positive half plane by distance and place them on the negative half plane as sorting on negative plane. In this event, we set values to p_{big} and p_{small} with the lists $list_1\{p_{big}\}$ and $list_1\{p_{small}\}$. After the i-th value in the list is used, the (i+1)-th value will be used next. The sorting result on the negative plane is shown in Fig. 8.

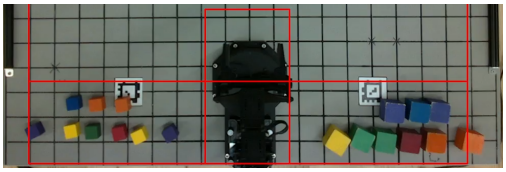


Fig. 8: In event 1, the sorting result on negative plane when 9 small blocks and 9 large blocks are placed on the positive half plane.

B. Event 2: Pick n' stack!

In the second event, we also detect and pick blocks on the positive plane by distance. When small blocks are picked, similar to event 1, we sort them on the negative plane. But when big blocks are picked, we directly stack them on the negative plane and update the stack position p_{stack}. The sorting result of small blocks are shown in Fig. 9. Because we need to pick the small blocks again, we place them separately.

After no blocks exist on the positive half plane, we begin to detect the right half plane and stack the small blocks to p_{stack} by distance. As shown in Fig. 10, the stack blocks are placed where the arm can place blocks accurately.

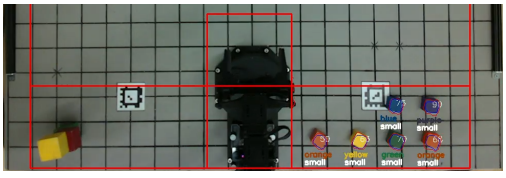


Fig. 9: In event 2, the sorting result of small blocks on negative plane when 6 small blocks and 3 large blocks are placed on the positive half plane.

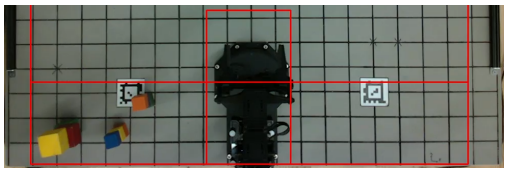


Fig. 10: The final stacking result of event 2, when 3 big blocks and 6 small blocks are used.

C. Event 3: Line 'em up!

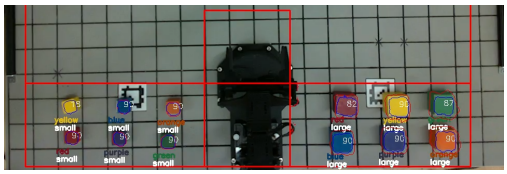
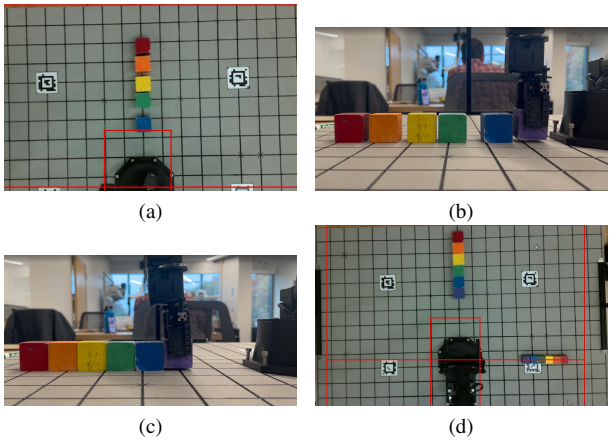


Fig. 11: In event 3 and 4, the sorting result on negative plane when 6 small blocks and 6 large blocks are placed on the positive half plane.

In the third event, we first sort all blocks on the negative half plane by distance as shown in Fig. 11. And then we first place the big blocks on the line $x = 0$ by color. When placing the last big block, instead of placing it on the board, we place it a little above the board and move it along the line $x = 0$ in order to push all the big blocks as shown in Fig. 12. After pushing the last big block, we open the gripper and finish the lining up



(a) (b) (c) (d)

Fig. 12: (a) The first five blocks placed position. (b) The last big block is placed a little above the board. (c) Use the last big block to push all the blocks. (d) The final result of event 3.

process of big blocks. The process for the small blocks is similar. We place the blocks on the line $y = 0$ and push the last small block along the line $y = 0$. The final result is also shown in Fig. 12.

D. Event 4: Stack 'em high!

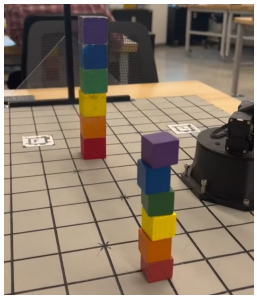


Fig. 13: The stacking result of event 4.

In the fourth event, same as the third event, we first sort all blocks on the negative half plane by distance as shown in Fig. 11. Then we pick blocks by color and place them to $p_{stack,big}$, $p_{stack,small}$ and update $p_{stack,big}$, $p_{stack,small}$ until all the blocks on the negative half plane are stacked. In the update of $p_{stack,big}$, $p_{stack,small}$, we keep the x, y values unchanged and add the size of block to the z value every time.

E. Bonus Event: To the sky!

In the bonus event, since blocks can be placed at will and only large blocks are under consideration, we feed blocks on the negative half plane one by one. Generally, we divide the whole process into 7 phases (Figure 14) as following, and utilize the basic pick and place functions mentioned above with specialized offsets.

(a) Stack 5 blocks using vertical place at p_{stack}.
(b) Stack 6 blocks using horizontal place at p_{stack}.
(c) Stack 4-5 blocks using vertical place at p_{pile}.
(d) Pick the first level of blocks at p_{pile} horizontally.
(e) Raise the pile to the desired height with $p_{raise,x} = -p_{stack,y}$ and $p_{raise,z} = p_{stack,z} + d_{large}$.
(f) Rotate the waist joint until $p_{raise,y} = p_{stack,y}$.
(g) Place the block pile to the top of the stack.

F. Evaluation

For events, 1, 2 and 4, we have got full points in the competition. We satisfied all the requirements of highest level and finished the task in the given time. But for event 3, we got 497 points out of 500 because our blocks are not aligned perfectly. The blocks placed on the board are slightly rotated. But after pushing the blocks, the alignment is much better.

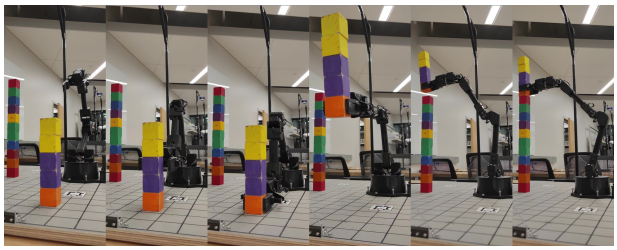


Fig. 14: Bonus event operation pipeline from (d) to (g). The best result we achieved in the competition was stacking 14 blocks, while the limit in the test was 16.

For the bonus event, in order to stack blocks as many as possible, we have to make sure that each block is picked at the same pose and placed within minimum deviations, that leads to a lot of offsets tuning and fixed picking, placing, stacking and even feeding positions. Although the vision detection system is still used, blocks were aimed to be fed form optimal locations, which were found from trials and errors. In the end, the task was more like a planned teach and repeat process.

G. Improvement

In event 3, although we choose the line $x = 0$ and $y = 0$ to place the blocks, there is still a small offset in block orientation. To reduce the offset, we can use experiment to find the lines which can minimize the offset. Moreover, we could somehow utilize, perhaps push against, the plastic board on each side of the board to force block alignment.

One improvement of the bonus event we can make is to completely remove the detection process and fix p_{pick}. If we ensure each the blocks are manually fed at the same location, then ideally dropping the detection system would lead to perfect feedback control system, or completely open-loop control strategy.

REFERENCES

[1] K. Lynch and F. Park, *Modern Robotics*. Cambridge University Press, 2017.
[2] Interbotics. Reactorx-200 - interbotics x-series arm documentation. [Online]. Available: https://www.trossenrobotics.com/docs/interbotix_xsarms/specifications/rx200.html
[3] J. S. Dai, "Euler–rodrigues formula variations, quaternion conjugation and intrinsic connections," *Mechanism and Machine Theory*, vol. 92, pp. 144–152, 2015.
[4] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2020.