# MEAM 520 Final Project

Anokhee Chokshi, Yug Ajmera, Divyanshu Sahu, Vanshil Shah

## 1 Method

### 1.1 Static

As our stacking strategy involved dealing with transformations, we first calculated all the tag frames w.r.t the base frame of the robot using the correct transformation matrices as shown below. We needed every tag in base frame. For that the camera pose w.r.t base frame ($T_{Base}^{Camera}$) was calculated using the following equation. In this we knew the tag 0 location w.r.t base($T_0^{Base}$) and we fetched tag0 w.r.t the camera frame($T_0^{Camera}$) from the list returned by $get\_detections()$

$$T_{Base}^{Camera} = T_0^{Base} T_{Camera}^0 = T_0^{Base} (T_0^{Camera})^{-1}$$

This enabled us to get location of each tag in base frame. For a $tag_i$, given its location in camera framae this is done as follows:

$$T_{tag_i}^{Base} = T_{Camera}^{tag_i} T_{Base}^{Camera} = (T_{tag_i}^{Camera})^{-1} T_{Base}^{Camera}$$

Our grasping strategy for static blocks is summarised in Figure 1 and is explained in detail in the following steps below:

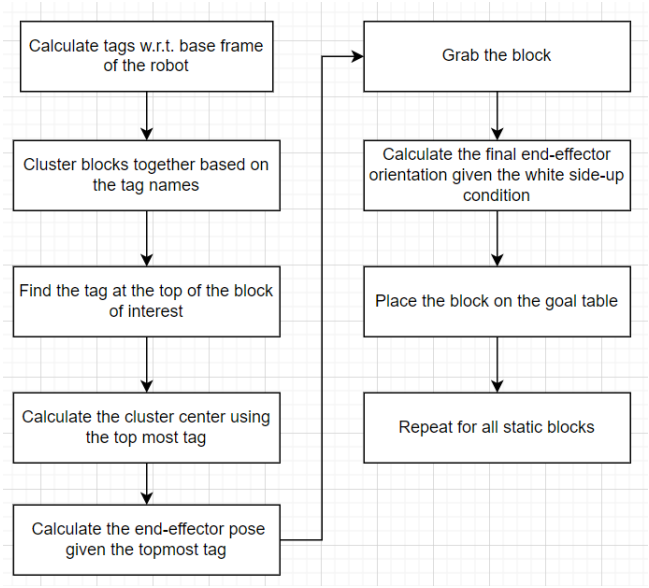



Figure 1: Algorithm for picking and placing of static blocks

### 1.1.1 Clustering

The first step consists of clustering the tags together and figuring out the orientation of the end effector for the block of interest. For this, we first get the tags from the *get_detections*() function. We use a filter described below to get the static block tags and given the prior information about the static tags, given one tag, we can figure out all the other necessary information about the block. Hence, we are only interested in the topmost tags of the blocks. The reason for this is because the topmost tags are the tags which have the highest probability of being visible to the vision system (in simulation at least).

The filter is described as follows:

From the given tag array, we first filter out the tags based on the name that tag possesses. As we know that the static tags only contain tag names from tag1 to tag6, we were able to filter the static tags out easily using this information. Now that we have the static tags, we can simply compare the z axis orientation of each tag and see whether it is aligned to that of the base frame by simply doing a dot product. Hence we now have the array of tags which are on top of the static blocks only.

After we take the information about the topmost tags for all the static blocks and we now know the number of static blocks that we have to pick up. Now that we know the tags, we calculate the orientation of the end effector for the first block of interest.

### 1.1.2 Orient the end effector + side bonus consideration:

Given the current tag info at the top of the block of interest, we calculate the block centre given the prior information about the relation between each tag in the handout. Hence, we can deterministically know where the white side is.Using this information, we can know how to orient our end-effector so that we always grab the block. We divide the grabbing strategy and the calculation for the same into two parts.

1. **White side up + on the sides** The first case is the one when the white is on the sides of the block. For the first case, when the white side is facing sideways, the way of calculating the orientation of the end effector is to rotate the current tag of interest two times one after the other so that we can calculate the end effector orientation to grab the block. These rotations are as follows:

   (a) Rotation by 180 degrees about the current y-axis of the tag of interest so as to orient the z-axis with the end-effector z-axis

   (b) Rotation about the current z-axis of the new frame by 90 degrees so as to account for the direction in which the gripper will open. The equational form of these rotations is as follows:

   The reason why we do this is because we want to grab the block from a face which is perpendicular to that of the white face so that while reorienting the block to keep it at the goal position we don't have to make extra changes to the orientation of the block for the side bonus. This effectively zeros out the amount of times we have to reorient the block after placing it at the goal position.

   For the case where the white side is up (TAG6 is facing upwards), the side by which the end effector grabs the block does not matter and hence, we keep the same rotations as above.

2. **When white side is down (TAG5 is facing upwards):** This case is particularly difficult as it required us to grab a block such that we could use only one rotation to keep the block to the goal position. For this, we had to update our strategy to orient our end effector such that the end-effector's z-axis was parallel to that of the base frame. The reason for this is that one we grab the block in this orientation, we could simply rotate the block by 180 degrees about the z-axis of the end effector frame and put it at the goal position which would give us the white side up. This strategy is as shown in the figure below.

   Now that we know the end-effector pose given the tag for the block of interest, we use the solveIK to calculate the configuration for the transformation matrix containing the orientation we calculated above with the position vector same as that of the tag with an addition of a z-offset w.r.t the base frame. This makes the robot orient itself above the block of interest with the desired orientation

3. **Grab the block** Now that we are oriented correctly, we simply grab the block using solveIK with the same orientation of the end effector and change the translation to that of the block centre of the current block of interest.

4. **Block Pickup + Safety considerations** After grabbing the block in the desired orientation, as calculated in the above step. We take the block to a position that has more height than our pickup table with the same orientation of the end effector. This is done to prioritise safety-first approach as there is a chance that while moving towards the final goal orientation, the end-effector could collide with other blocks in the vicinity of the pickup table.

5. **Final orientation of the end effector for placement of the block on the goal table** The current orientation of the block centre is calculated by using structural priors from the tag that was detected on the topmost surface of the block. We know the relative orientation of a tag and its block centre using the diagram in the report for every tag ($T^{tag_i}_{Block\_center}$). Now we apply this transformation to each tags pose to get the block centre's orientation in base frame. .

$$T^{Base}_{Block\_center} = T^{Base}_{tag_i} * T^{tag_i}_{Block\_center}$$

   Moreover, after we grab the block, the end effector and the block become a single entity. Hence, we know the relative orientation between the block centre and the end-effector frame as well($T^{end\_effector}_{Block\_center}$).

$$T^{end\_effector}_{Block\_center} = T^{end\_effector}_{Base} T^{Base}_{Block\_center}$$

   To get the side bonus, we wish that the block centre's desired orientation with z-up is such that it aligns with the base frame($T^{Base}_{Block\_center}(desired)$).We get this by putting the roatation matrix as identity and the translation matrix as hadcoded by us w..r.t base frame such that it is placed on the reward table. Additionally, knowing the fact that the end effector and block are now a rigid body, the transformation between the end effector and the block remains the same at both goal position and pickup position. Using this continuous chain of transformations, we calculate the end effector's desired orientation in base frame for achieving the side bonus in which the white side of the block will always face upwards.

$$T^{Base}_{end\_effector}(desired) = T^{Base}_{Block\_center}(desired) T^{end\_effector}_{Block\_center}$$

   The whole process is also expressed through Figure 2.

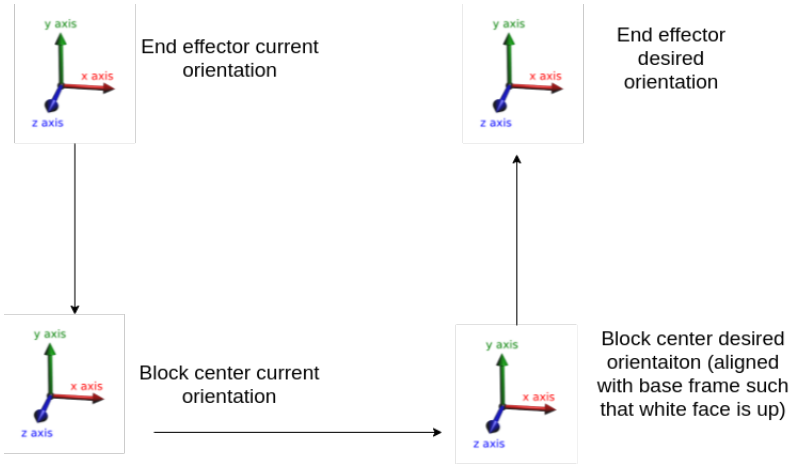



Figure 2: Calculation of end effector goal position for side bonus

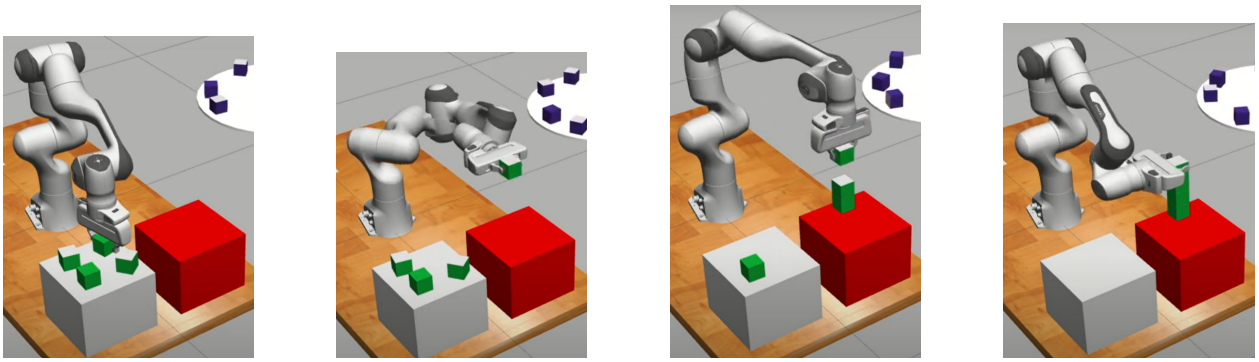


Figure 3: Picking and placing of static blocks in simulation

## 1.2 Dynamic

Our strategy for dynamic blocks differed for simulation and real world case as we had different priors for both cases.

### 1.2.1 Simulation

For performing dynamic block grabbing in simulation we had much more information to estimate the position of the block. Using this information we calculated the most favourable block which we could grab. Our strategy is summarised in Figure 4.

1. **Calculating the most favourable block** The first step for grabbing a block in simulation was to calculate the block which would be the most favourable one to grab out of all the rotating blocks on the table. For this, we first assume that the table is rotating in the clockwise direction.

   For calculating the most favourable block, we separate the blocks in the quarter of the table which has the most negative x and y. This is because as the table is rotating in the clockwise

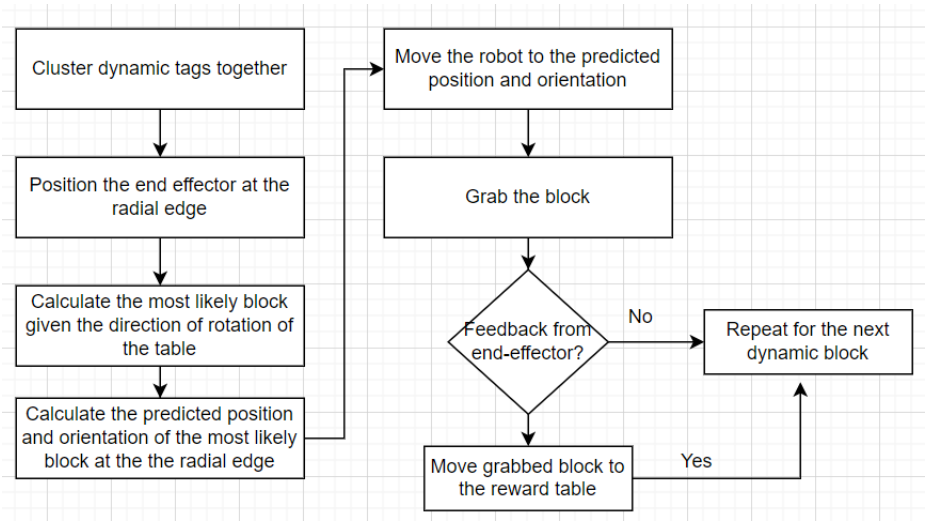
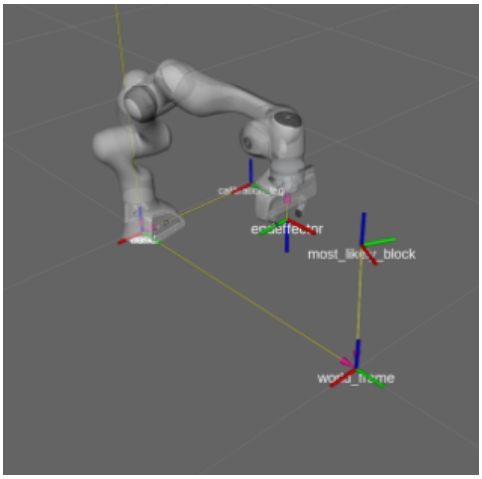
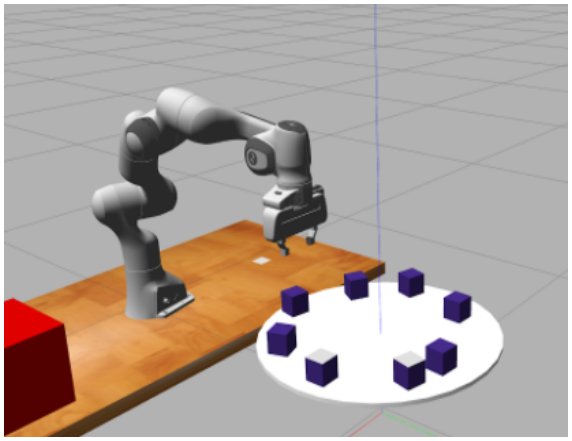



Figure 4: Dynamic block grasping strategy for simulation

direction, the block with the most negative x and y will have the highest probability of being grabbed by the end effector



(a) RViz



(b) Gazebo

Figure 5: Grabbing dynamic blocks in simulation

One of the biggest uncertainties in the simulation was the time taken by the solveIK to give a solution for the goal position. Hence, we first calculate the desired position and orientation for the most favourable block and then we divide the part of grabbing the dynamic blocks into two parts for maximum efficiency.

2. **Position and orient the end-effector as per the most favourable block** After calculating the most likely block, we calculate the radial distance at which it is kept from the world frame $r = \sqrt{x^2 + y^2}$. This information would be used to decide the translation that the end effector needs to undergo to pick up this block when it reaches the x axis of the table as our predefined position aligns with the x-axis

   We are orienting the end effector by calculating how much the block would have rotated when

it reaches our predefined position. This is done by calculating the current angle it makes with the x-axis which is $\theta = tan^{-1}(y/x)$. Now we premultiply the current transformation matrix of the block with a matrix that would rotate it by that much theta in the world frame getting the predicted transformation matrix of the tag.

$$T_{tag_i}^{world}(predicted) = T(\theta)T_{tag_i}^{world}(current)$$

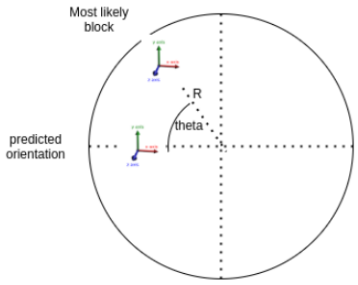



Figure 6: Calculation of predicted pose for dynamic block

3. **Position the end effector and grab the desired block**: Now that we have calculated the predicted orientation and position of the block of interest, we divide the grabbing strategy into two parts which are explained as follows:

   (a) For the dynamic blocks in simulation, we always predict the position and orientation of the block of interest at the radial quarter of the table. In order to maximise the efficiency of grabbing the block of interest, we know that everytime we will have to position the robot at the radial edge. Hence, first, we always the position the robot at this edge. For this, we have used the same idea of seed optimisation and use the best seed which minimises the time needed to go from the neutral position or the given reward table orientation.

   (b) Now that we have the robot ready to grab the block for out predicted orientation We then start from this position and grab the block using the seed as the current transformation matrix of the robot. This helped us maximise the efficiency with which we were able to grab the dynamic blocks.

### 1.2.2 Real life

Our strategy for the real world dynamic block grabbing changed a lot as our robot had no visual feedback in real time. Hence, our strategy applied to pick up the blocks for the competition changed significantly.

As we know that the blocks are stacked radially and their white side is always up. We need not calculate the most favourable block as we did in the simulation scenario. Instead, we take the advantage of this prior information and optimise the grabbing strategy by introducing aperiodic motion into our grabbing manoeuvre.

For this, we fist move the end effector to the closest radial end of the rotating table and then perform the grabbing maneuver aperiodically with an aim of establishing a sync between the grabbing of the

blocks on the rotating table. The randomness in grasping is induced by using time.sleep(), which stops the execution of the code for a certain time interval.

To check whether the end effexor got hold of a block or not ,we use $arm.get\_gripper\_state()$. The function returns the position of the end effector and the force of the end effector which it is applying on the object. We use the position information of the end effector if it is greater than a certain threshold, it implies that there is an object between the gripper. We then use this information to make the end effector go to the reward table for placing the block on the existing stack.

# 2 Evaluation

## 2.1 Testing

### 2.1.1 Testing in simulation

For experimenting with our modules that we implemented we extensively visualised our frame transformations in RViz. Before giving the transformation matrix as a goal to the end effector, we broadcasted the transformation matrix using a *show_pose* function in our code. This used the $tf\_broadcaster()$ that we implemented in the ROS node and gave us the visualization of the goal that the end effector would reach pararelly to the feedback that we got from gazebo.

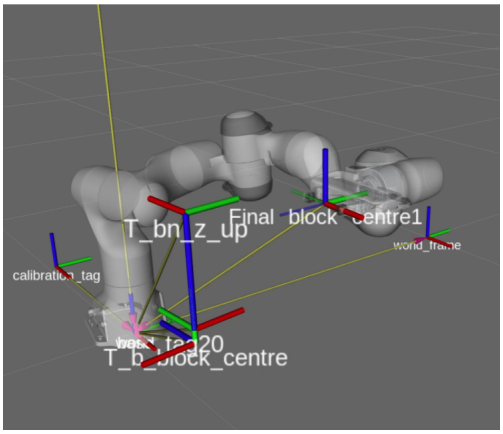



Figure 7: Visualisation of frames in Rviz

### 2.1.2 Considerations for blue team and red team

The red team and blue team's reward table and pickup table were mirror images of each other. We accounted for this in our transformation of the base of the robot with the world frame by changing the sign of translation in the y-direction. Moreover the seeds that we collected for the red team during the pick up process of the block were used while solving the placement of the blocks during the trials as blue team

### 2.1.3 Testing in real world

We tested our solution multiple times on the real panda arm after getting successful results in simulation to make it more robust. One of the major takeaways from this testing was accounting for the orientations and position of the end effector in which there were self-collision and also collision with other blocks.

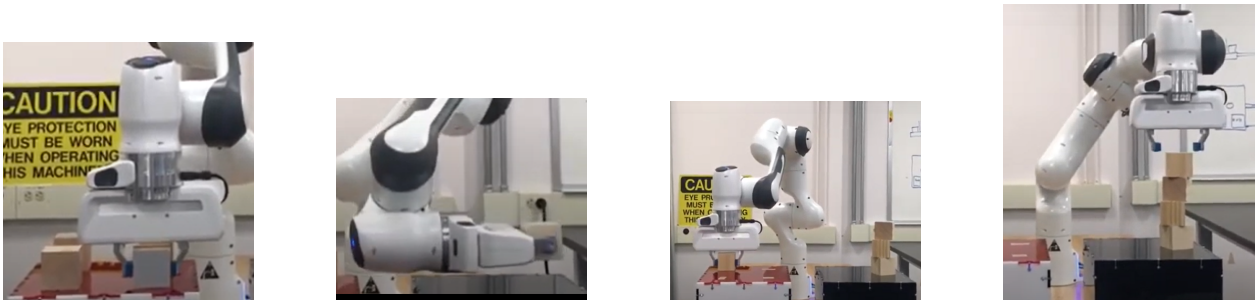



Figure 8: Picking and placing of static blocks in real world

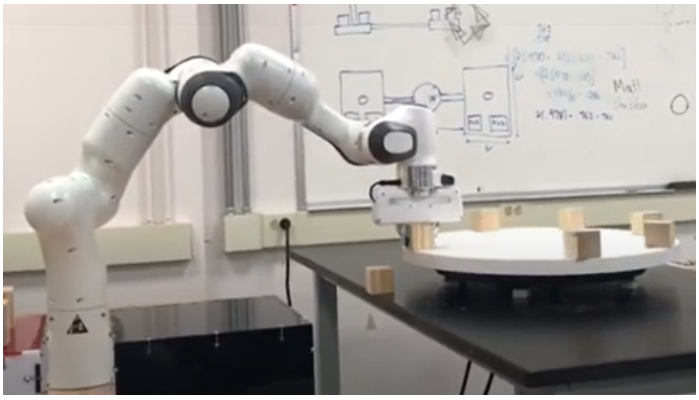


Figure 9: Picking and placement of dynamic blocks in real world

### 2.1.4 Metrics used for evaluation

The evaluation metrics that we chose to test out solution were:

- The time taken for the complete pick up and place challenge
- Number of static cubes that were picked and placed (considering white face down) out of the total cubes [success rate]
- Number of dynamic cubes that were picked against the number of tries that we did to grasp them

As can be seen from the table, we were able to pick and place static blocks with a high success rate for white side on top and side with side bnus configuration. We were also successfully able to pick up some white side down blocks more successfully in simulation than in real world. In the real

| Step | Orientation | Success rate (Red) Simulation | Success rate (Blue) Simulation | Success rate (Red) Real world | Success rate (Blue) Real world |
|---|---|---|---|---|---|
| Block Grasping | Tags 1,2,3,4 | 12/12 | 12/12 | 8/8 | 8/8 |
| | Tag 5 | 7/8 | 6/8 | 5/8 | 5/8 |
| | Tag 6 | 12/12 | 12/12 | 8/8 | 8/8 |
| Goal Delivery | Tags 1,2,3,4 | 12/12 | 12/12 | 8/8 | 8/8 |
| | Tag 5 | 7/8 | 6/8 | 5/8 | 5/8 |
| | Tag 6 | 12/12 | 12/12 | 8/8 | 8/8 |

Table 1: Static Blocks simulation and real world evaluation

| Trial Number | Time taken in seconds(Simulation) | Time taken in seconds(Real world) |
|---|---|---|
| 1 | 122.24 | 132.88 |
| 2 | 118.02 | 126.55 |
| 3 | 125.16 | 135.88 |
| 4 | 126.32 | 132.65 |
| **Average** | **122.935** | **131.99** |

Table 2: Time taken for static block stacking in simulation and real world in seconds

| Step | No. of blocks grasped (Simulation) | Total no. of tries (Simulation) | No. of blocks grasped (Real world) | Total no. of tries (Real world) |
|---|---|---|---|---|
| Block grasping | 3 | 8 | 2 | 15 |
| | 3 | 10 | 1 | 9 |
| | 2 | 6 | 1 | 13 |

Table 3: Dynamic Blocks simulation and real world evaluation

world, considering the safety of the robot, we did a lot of parameter tuning to achieve the white side down configuration with side bonus successfully. Moreover, we were also able to complete most of our runs in about 2 mins both in simulation and in real world. Our performance greatly increased after doing seed optimisation for static and dynamic blocks.

We were also able to pick up dynamic blocks both in simulation and in real world. As we had more information about the blocks in real world, we were able to grasp more number of blocks. For the real world scenario, we were able to grasp blocks after doing hyperparamter tuning for our aperiodic algorithm so that it could sync up efficiently against uncertanities.

## 3 Score optimization strategy

We have highlighted the ways through which we tried to maximise our score during the final competition.

### 3.1 Static first approach

First we prefer placing the static blocks first than the dynamic blocks because even though dynamic blocks have more points than static blocks, the picking of dynamic blocks is a random process and based on chance. Hence after completing the placement of all static blocks we command our robot towards the turntable for picking dynamic blocks

### 3.2 Handling white side down block

The previously described strategy of handling white down blocks worked but made the robot come too close to collision with the surface of the table for both the start and the goal position sometimes. To overcome this challenge we incorporated another way of tackling the white down, in which we place the block on the stack then re-orient the gripper at 45-degree angle, with respect to the side faces. Grasping the block in such a way and rotating it twice about the side face resulted in the white face on the top. We tested this solution in simulation and were able to pick up and place the white face down block on the goal table with the side bonus configuration in a collision-free manner. However, keeping in mind the time constraints of the competition, we didn't use this method and treated this like a white face up block. Rather than achieving side bonus configuration for this block we wanted to capture as many dynamic blocks as possible.

Also, this is the only solution we considered to be optimal as other strategies to solve this problem, included us to keep the block at the goal position and reorient the block again at the goal position multiple times which is not good from a strategy viewpoint.

## 4 Analysis

1. **Solve IK**:
   The major takeaways from the project were regarding the inverse kinematics solver and the way it functions. There were two major areas that we focused on to get the best performance of our inverse kinematics solver.

   First one was to tackle the problem of starting the optimization problem with a bad initial guess which is commonly referred to as the "cold start problem". The initial seed configuration that was used to solve for each goal point was systematically saved in a .txt file each time the IK converged. We then used this txt file to iterate over seeds for a new goal point which was near to our initial goal. We also accounted for the number of iterations that each seed was taking to solve the given goals on an average, then sorted it in a list from least to most iterations. This led to faster convergence and superior performance.

   Another challenge in using the IK solver is to tune its hyperparameters. We carefully chose the step size, so that it isn't big such that it results in erratic descent over the loss surface or it isn't too small which leads to slower convergence. Our solution failed many times because the final configuration solved by the algorithm was out of joint limits. Tuning the joint centering rate led to better solutions . This also had to be done in a careful fashion as a higher joint centering rate would lead to bad solutions for the primary task of the ik solver which is to reach the desired goal.

   The cold start problem and the tuning of the hyper parameters for solveIK led to a dual convergence. In it, as we improved our seeds, the solveIK was able to give faster solutions.

On the other hand, as we improved the hyperparameters of solveIK, we got better seeds for our robot. Hence, both our seeds and the efficiency of our solveIK improved significantly. This strategy allows the robot motion to be smoother and faster in the real world tests.

2. **Differences between the sim and real world that we realised while testing**:

   The other thing that we observed was that our tuning that we did in simulation for picking and placing the blocks didn't directly transfer from simulation to real world. While testing on real robot, the height at which we were intially dropping the blocks from resulted in toppling of the blocks. This was due to less friction of blocks which wasnt modeled in the gazebo simulation. To account that we needed to change the height of our goal of the end effector while placing.

   Additionally, while placing the blocks, for some orientations that the end effector achieved resulted in collision with the blocks on the pick up table. We took this into consideration by changing the order of picking the blocks, we were picking the blocks first which were near the reward table and the goal was selected in such a way that it didnt touch the other blocks.

3. **Accounting for SolveIK non-convergence for difficult orientations**:

   Although we have accounted for all the end effector orientations and have generated the best possible seeds for both the blue and the red teams, very rarely, the orientation of the end effector would be such that the solveIK would fail to converge. As it was not feasible to account for all of such possible cases, we used a simple heuristic to make our solution converge. It is worth noting that this problem is only faced while orienting the end-effector to pick up the block and when calculating the goal orientation of the end-effector.

   For solving the problem during the block pickup, we found that certain times due to some specific orientations of the tags, our solveIK failed to solve for the current calculated orientation of the end-effector. This was mainly because the last joint limit was exceeded every time. A simple and elegant heuristic we found was to simply rotate the calculated orientation of the end effector by 180 degrees about it's z-axis. This solution preserved our solution for grabbing the block on a face perpendicular to the white face and also helped our solveIK converge easily.

   When our solveIK failed to converge for the goal orientation, it was mainly because the calculated orientation for the end-effector for the side bonus made the solveIK exceed joint limit. While doing practical tests, we realized the orientation of the block did not matter as long as the white side of the block was up. Hence, the solution to this was simply to rotate the current frame by 90 degrees about the x-axis clockwise. It is worth noting that the direction of rotation here does not matter as explained above. Hence, using this failsafe, we were able to guarantee a solution with a side bonus every time.

# 5 Insights gained from this project

During the entire course of this semester, we built library modules for solving the different aspects of the Panda manipulator. During the final project, we saw the application of all those modules together to solve the final project. The use of gradient descent to cleverly solve the inverse kinematics problem was one that fascinated us the most. Throughout this project, we were able to learn the interplay between different transformations how they generalised to solve a pick and place problem. Moreover, we also learnt ways to cleverly hack the gradient descent algorithm by using

the warm start methodology to minimise the time it take for out solution to be calculated. We also were amazed to see the strategies different teams implemented to win the competition. The idea of grabbing the blocks dynamically by orienting the end effector parallel to the table and sliding it from sideways into the table rather than from the top was a strategy that we thought was ingenious. We were also able to implement the real world strategy for picking up the block and realised the differences from the simulation for the same. All in all, we had an amazing time working on this project.