Design Choices for Dual-arm Robotic Manipulator Control

Marcel Torné Villasevil
Massachusetts Institute of Technology
Cambridge, USA
marcelto@mit.edu

Idan Shenfeld
Improbable AI Lab, CSAIL
Massachusetts Institute of Technology
Cambridge, USA
idanshen@mit.edu

Alex Kashi
Massachusetts Institute of Technology
Cambridge, USA
alexkash@mit.edu

***Abstract*—Designing controllers for Dual-arm robots brings another level of complexity to the system design over a single-armed robot. In this work, we look into several techniques for controlling a dual-armed robot system in performing various manipulation tasks. We propose three different controllers for a dual-armed robot system: 1) separate controllers for each arm with assumptions on the workspace, 2) separate controllers with a communication channel between them, and 3) single controller for both arms. We constructed a series of manipulation tasks to check the advantages and disadvantages of each approach and concluded with a comparison between them. Our code is available at github.com/idanshen/Bi_Manual_Robot**

I. Introduction

As robots become more available, the variety of problems researchers try to solve with them grow. One robotic arm is insufficient for some of these problems; therefore, it is common to see dual-armed robotic systems. There are several advantages of using a dual-arm robot rather than a single arm. First, it allows the robot to control both parts of the same task, for example, a typical peg-in-hole task, with one arm positioning the peg and one arm the hole. On similar notes, some tasks might require more than a single arm, for example, picking up a big box. Another advantage is operating in environments built for humans that, by nature, are made for dual-arm manipulation. Finally, they allow the robot to perform the same tasks at twice the speed of operating in coordination. [3]
Along with its usability, the dual-arm configuration increases the complexity of designing a controller for the robot. Controlling each arm separately without considering the maneuvers of the others constrains the family of tasks one can solve. Having a single controller for both arms is a common approach [4] but introduces some difficulties. For dual IIWA robots, for example, the number of degrees of freedom goes from 7 for a single arm to 14 degrees of freedom for a dual arm. Furthermore, new constraints appear, such as a collision between the two arms. Unlike collision with a static object, the fact that both arms move simultaneously makes the motion planning optimization problem harder. In addition, executing maneuvers with precision, both in time and space, becomes more important. For example, if we want to pass an object from one robotic hand to the other, the temporal and spatial coordination must be accurate; otherwise, the maneuver will fail. This requires the controller to react quickly, constraining the complexity of the algorithms it can run. Because of the complexity of designing a single controller, another common approach is to have two separate controllers, each in charge of a different arm but with some communication channel between them [3]. This approach makes the design of such a controller simpler. However, it introduces limitations over the manipulation tasks and coordination that the overall system can do since we limit the amount of data the robots can transfer. In this work, we investigated those design choices. In order to get insights into the advantages and disadvantages of each approach, we implemented three manipulation tasks for dual-arm robots that helped us study each approach.

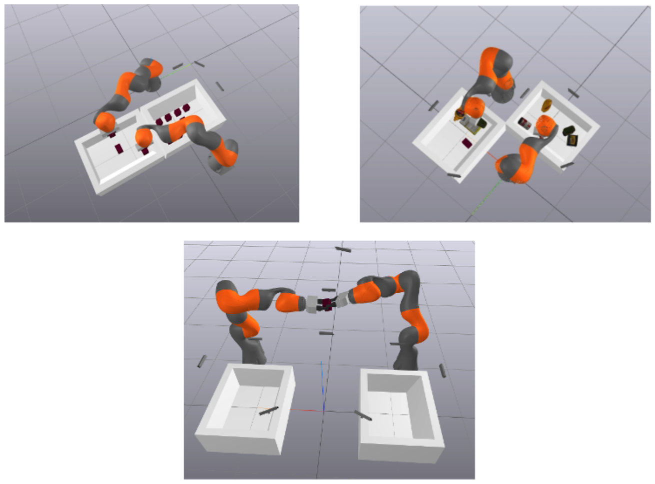
Fig. 1. The manipulation tasks that were implemented as part of this project. (From top-left clockwise) Asynchronous pick-and-place; Pick-and-place in tight space; Passing an object from one arm to another.

II. Related Work

There is abundant work on dual-armed robots, going all the way back to the 1950s [5]. In the robotic literature, it is commonly assumed that a single system control both arms [3]. Because of the complexity it introduces, a lot of attention was given to the right way to do motion planning in such a high-dimensional space [6]–[8]. Separate controllers, with or without communication channels are usually studied under the scope of multi-agent systems [9]. There, a separate controller is in charge of every robot. However, usually, each robot has some information about the other, either through its sensors, communication channel, or prior assumptions [10].

III. Methods and Testing Environments

In the following section, we will describe the various design choices we studied for dual-arm robot controllers and the use cases we used in studying them. All of our experiments have been done using Drake and two instances of the KUKA IIWA 7-DOF robot to create a dual-arm robotic system. To control the IIWAs, we used Drake's Differential Inverse Kinematics Integrator which enable us to command the end effector of the IIWAs using position control. One of our assumptions in this work is that sufficient coverage using calibrated depth cameras exists in the workspace.

A. Separate controllers with assumptions on the workspace

For the first method, we designed a system where the two arms would work independently using separate controllers, without any knowledge of the maneuvers of others. The task we chose was picking objects from the same bin and placing them into another. To avoid collisions, we introduced some assumptions about the workspace of each arm. We assumed that each arm will not cross the halfway line between their two bases. To achieve this end effector geofencing, one robot would select grasps from the left-hand side of the source bin, and the other would select grasps from the right-hand side of the bin. Once the object was grasped, the end effector was commanded to a distinct safe location. Once an end effector arrives at the safe location, it follows a pre-planned path to its drop-off location. After the block is dropped off, the pre-planned path is reversed. This achieves the behavior of geofencing without the need to switch to a path planner that solves a constrained optimization problem.
The planner we implemented was based on the one presented in [2], chapter 5. The point clouds generated by each of the cameras were then fused into a unified representation of the environment. Afterward, normals for each point were generated by fitting a plane to the cloud of points in a small area around it. Using this representation of the objects in the bin, an algorithm generates a series of candidate grasping points and compares them using a hand-tailored cost function. Each candidate grasping pose is an antipodal grasping aligned with the normal of a random point on the object. After a grasping position is selected, a path is planned to the grasp pose using linear interpolation from the current location of the end effector. To bring the object from one bin to another a series of intermediate frames are generated since direct interpolation will result in a collision with bin walls or the cameras. This is the part in the algorithm where we used the assumptions about the workspace of each of the arms to avoid a collision. A similar planner is the base of the algorithms we will present for the other controllers as well.

B. Separate controllers with communication channel

The third design option for controlling a dual-arm robot we checked in this work is separate controllers with communication channels between them that allow them to transfer needed information. To study this type of controller, we designed the following task - the first robotic arm needs to pick up an object from the first bin and pass it to the second robotic arm, which will place it in the second bin. The object is being passed directly from one robotic arm to the other without being set down, similar to the problem described in [1]. Moreover, the two bins are far apart, so a single robotic arm cannot accomplish this task alone. As with all other tasks in this project, we assumed a sufficient coverage of depth cameras. We chose this task because it represents a practical and useful problem. Using both robots enables a larger workspace than a single arm can have.
This task has two main technical challenges - executing coordinate movements between the robot and computing valid grasping positions for the hand-over.

1) Coordinated Movement: The receiver arm should grab the object only after the other arm has reached the hand-over position and stopped moving. Only after the receiver arm gets a hold of the object the other arm should start to release it. To solve this problem, we established a communication channel between the two controllers that pass two basic signals - when the first arm is in position and after the second arm grabbed the object. The communication channel was implemented using two binary input-output ports between the controllers.
Since the object can be passed back and forth between the bins, the controller for each arm is identical, and its general control scheme is described as a state machine in figure 4. The controller is aware of its current job as the object's GIVER or RECEIVER. While the GIVER picks the object from the bin and carries it to the hand-over area, the RECEIVER moves to a waiting position to shorten the time it will take to reach the object. After the GIVER reach its position, it sends the relevant signal and waits there. The RECEIVER queries a valid grasping position (see more details in the next section) and moves to grab the object. When the object has firmly grasped, a signal is sent to the GIVER to release the object. After dropping the object at its bin, the two arms change their roles.

2) Generate grasping positions: Since the items passed between the robots are relatively small, there is a need to carefully choose the grasping poses for the arms. For generating the grasping position for the first arm, we followed the algorithm described in [2], chapter 5. The only difference was that instead of adjusting the gripper to be on the center of the object, we chose to center it around its upper half. That

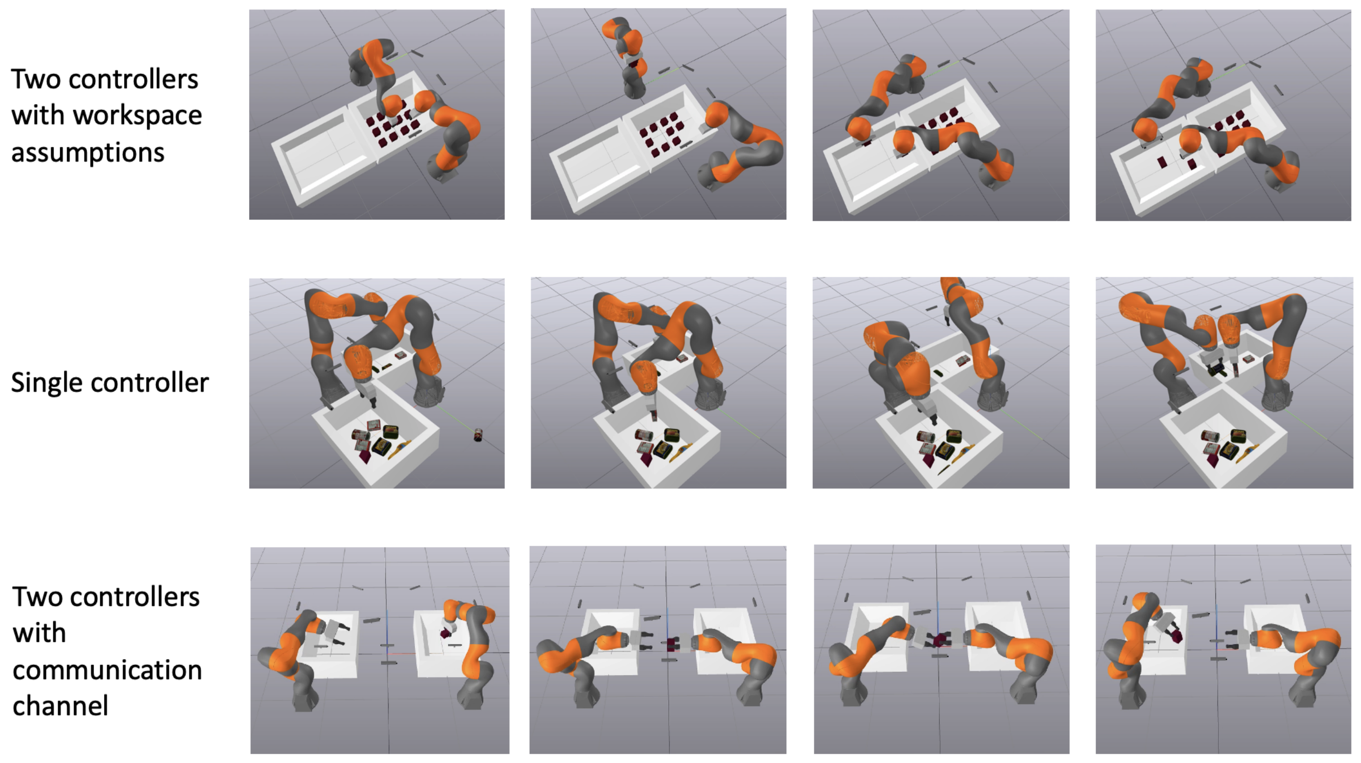
Two controllers with workspace assumptions
Single controller
Two controllers with communication channel

Fig. 2. Rollout sequence for each controller on its designated task. The timestep of each episode goes from left to right.

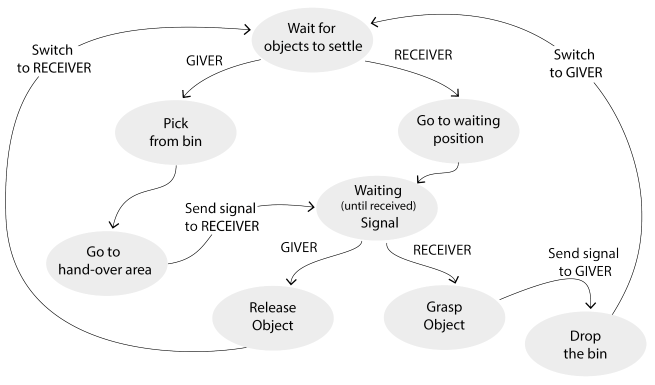

Fig. 3. State machine for the single controller with a communication channel. Each controller is aware of its current role as the GIVER or RECEIVER of the object.

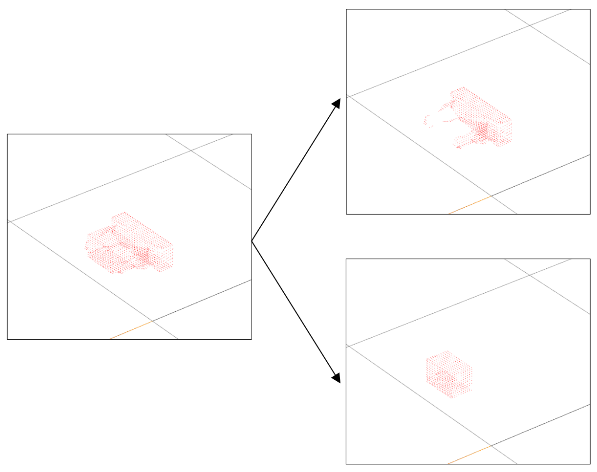

Fig. 4. An example of pointcloud decomposition obtained during the collaborative grasping algorithm.

way, the first gripper leaves more space for the second gripper to get a hold of the object. After the first arm got into the hand-over areas, a point cloud of its gripper and the object was extracted using three cameras that were positioned around that area. Based on the gripper's position and the opening of the gripper's finger, our algorithm separated the points that belong to the gripper from those of the object, See figure 4 for example. Then, a series of random points on the object on the object, and each one is examined as a potential candidate for being one side of antipodal grasping. This examination contains three parts:

1. Giving the known shape of the gripper and the candidate gripper, there is a check that there is no collision between the gripper and any point in the point cloud.
2. Under the assumption of maximum opening of the gripper fingers, there is a check that none of the points that belong to the other gripper reside between the first gripper fingers.

Algorithm 1 Generating Grasping Position for Hand-over

```
    **Input:** point cloud $Ps$, GIVER Gripper frame $X_G^{GIVER}$,
    cost weights $W_1, W_2, W_3$, *number of points*
 2: *poses* $\leftarrow []$
    *costs* $\leftarrow []$
 4: $Ps_{object}, Ps_{gripper} \leftarrow$ *SplitPointCloud*$(Ps, X_G^{GIVER})$
    **for** $d = 1, 2, \dots,$*number of points* **do**
 6:     $idx \leftarrow RandomInteger(Ps_{object}.size)$
        $N_p = Normal(Ps_{object}[idx])$
 8:     $Gx = N_p$
        $Gy = [1,0,0] - ([1,0,0]^T Gx) \cdot Gx$
10:     $Gz = Gx \times Gy$
        $X_{WG} = [RotationMatrix(Gx, Gy, Gz), Ps_{object}[idx]]$
12:     **if** $Collision(X_{WG}, Ps)$ or
        $IsBetweenFingers(X_{WG}, Ps_{gripper})$ **then**
            *poses* $\leftarrow X_{WG}$
14:         *costs* $\leftarrow \infty$
            **break**
        **end if**
16:     *poses* $\leftarrow X_{WG}$
        *costs* $\leftarrow$
        $ComputeCost(X_{WG}, N_p, X_G^{GIVER}, W_1, W_2, W_3)$
18: **end for**
    $MinIdx \leftarrow argmin(costs)$
20: **return** *poses*$[MinIdx]$
```

3. A cost function is calculated the help choose between all the feasible candidates:

$$C = -W_1 \cdot [1,0,0]^T (R_G \cdot [0,1,0]) - W_2 \cdot |{N_p}^T R_{G_x}|^2 + W_3 \cdot |{R_{G_x}}^T R_{G_x}^{GIVER}|^2 \quad (1)$$

Where R_G is the rotation matrix of the gripper, R_G^{GIVER} is the rotation matrix of the other gripper that currently holds the object, N_p is the normal of the chosen point, and W_1, W_2, W_3 are the weights assigned to each cost term. Subscript x symbolize the x-axis vector of the rotation matrix. The first cost term penalize deviation of the gripper from a horizontal position, the second one penalizes deviation from being parallel to the normal, and the last cost term rewards the grip position to be perpendicular to the gripper that currently holds the object. The overall algorithm for generating grasping position is described in algorithm 2.

This task clearly demonstrates the usefulness of this controller's design approach. Since each robot operates around its bin, with only the hand-over area in common, there is almost no risk of collision between the arms. Therefore, motion planning in the joint space only adds unnecessary complexity to the controller. The amount of information that needs to be shared between the robots is minimal and contains only two binary signals that can be quickly and efficiently transmitted over a communication channel.

C. Single controller

The second design option we explored was a single controller to control both arms. Having a single controller means we have complete information about the state of the two arms at every point in time so that we can plan accordingly. To test this approach, we designed the following task: Moving objects from one bin to the other, where one arm moves objects from bin 1 to bin 2 while the other arm moves objects in the opposite direction. Moreover, the bins and robots are located near each other, which requires both arms to operate in the same space. This benchmark is prone to many collisions since, without a collision avoidance mechanism, the trajectories are crossing most times.

1) State machine: Perfect synchronization between arms and a clever path-planning algorithm will solve the problem. The first key is the state machine we designed, which can be seen in figure 5. After each grasping and dropping, the arm that finishes first will wait for the other arm to finish its task. Each arm will wait on top of the bin they were targeting for the grasp. If they were dropping/grasping on that bin, the arm would wait on the same bin for the other to finish. Finally, if the grasping was unsuccessful, the arm that failed would retry the grasping procedure. This synchronization allows us to have more complicated tasks, with more freedom on where the arms can move or grasp objects from but restricting freedom in timing.

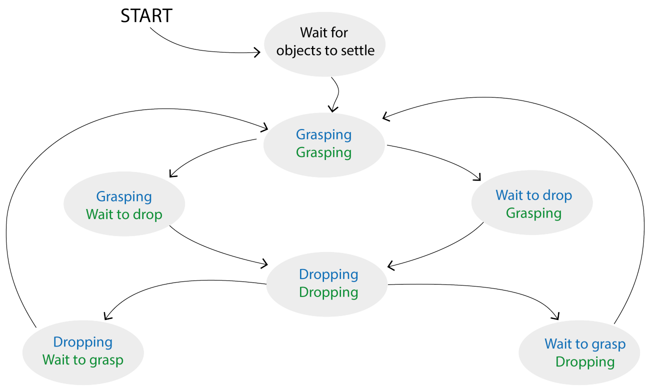

Fig. 5. State machine for the bi-manual manipulation with a single controller.

2) Joint motion planning: The first solution we implemented to avoid collisions was adding a clearance point that was far apart enough for each arm not to collide when moving simultaneously from one bin to the other. This solution is similar to the approach used in the last section. After implementation, we found three main downsides to this approach. First, we need to hand-design the clearance point, which will change for each environment configuration. Second, depending on the space, having this clearance point might not be enough to avoid collisions. Last, making each trajectory pass through this clearance point makes the trajectories suboptimal in time. Therefore, to solve these three issues, we designed a new recursive algorithm that optimizes the trajectory for both arms simultaneously while avoiding collisions between them. In this way, after grasping or dropping the object from a bin, both arms will rollout a collision-free trajectory taking into account the other arm. This algorithm is presented in 2. Given a starting and end frame for each one of the arms, we return a series of subframes that will guide both arms to reach the goal without collisions between the end effectors. Finally, after getting the key subframes, we finish by interpolating between them using linear interpolation.

The optimization problem we need to solve is presented in III-C2. It consists of a cost minimization problem, where we have a quadratic cost with quadratic constraints. Those optimization problems are known as Quadratically Constrained Quadratic Programming (QCQP) [11] and have well-established solvers. We used the solver implemented natively in Drake mathematical program solver. At each iteration, we minimize the distance between the starting point and the target point as well as the endpoint for each arm. Then we set constraints for this target point to be at the same distance between the start and end frame in order to make the recursive interpolation stable. Finally, but most importantly, the last constraint is that both frames for the arm have to be separated by a minimum distance. We chose this distance based on the size of the end effector to ensure there will not be a collision. The overall optimization problem is:

$$\begin{gathered} min_{xyz}||p_1 - X_G^{start1}||_2^2 + ||p_1 - X_G^{end1}||_2^2 \\ +||p_2 - X_G^{start2}||_2^2 + ||p_2 - X_G^{end2}||_2^2 \\ \text{s.t.}||p_1 - X_G^{start1}||_2^2 = ||p_1 - X_G^{end1}||_2^2 \\ ||p_2 - X_G^{start2}||_2^2 = ||p_2 - X_G^{end2}||_2^2 \\ ||p_1 - p_2|| > d \end{gathered} \tag{2}$$

Where p_1 and p_2 are both three-dimensional vectors representing the target point or translation of the frame we are searching for, in world coordinates for each arm. X_G^{start1},X_G^{start2} and X_G^{end1},X_G^{end1} are the starting and ending frames for each arm that we are interpolating between, and d is the minimum distance required between end effectors.

As the reader would notice, in the current implementation of our algorithm we do not take into account the whole arm but only the end effector. This means that our algorithm will return non-colliding trajectories for the end effector position of each arm, however, the arms might still collide on other parts of the other arm. The direct way to solve this issue is to add more constraints to the optimization problem. However, this will require changing our controller from position control using differential inverse kinematics to full joint control. We believe that it is doable but because of time constraints and since it didn't serve our main goal of comparing different ways to control a dual-armed robot we left it for future work.

Committed to still being able to provide a reliable solution, we added a geofencing constraint to the optimization problem, which will pull each end effector to a specific region so that the two arms will not get tangled. These constraints are defined through a vertical plane splitting the boxes by

Algorithm 2 Recursive bi-manual end-effector path-planning optimization

```
1: Input: start frame arm 1 $X_G^{start1}$, start frame arm 2 $X_G^{start2}$, end frame arm 1 $X_G^{end1}$, end frame arm 2 $X_G^{end2}$, $depth$, start_time, end_time
2: $frames \leftarrow [(X_G^{start1}, X_G^{start2}), (X_G^{end1}, X_G^{end2})]$
3: $times \leftarrow [start\_time, end\_time]$
4: for $d = 1, 2, \ldots, depth$ do
5:     $N \leftarrow frames.size$
6:     $new\_frames \leftarrow []$
7:     $new\_times \leftarrow []$
8:     for $i = 1, 2, \ldots, N$ do
9:         $X_G^{mid1}, X_G^{mid2} \leftarrow$
           $SolveMathematicalProgram(X_G^{i1}, X_G^{i2}, X_G^{i+1,1} X_G^{i+1,2})$
10:        $new\_frames \leftarrow (X_G^{i1}, X_G^{i2})$
11:        $new\_frames \leftarrow (X_G^{mid1}, X_G^{mid2})$
12:        $new\_times \leftarrow times_i$
13:        $new\_times \leftarrow \frac{times_i + times_{i+1}}{2}$
14:    end for
15:    $new\_frames \leftarrow (X_G^{N1}, X_G^{N2})$
16:    $new\_times \leftarrow times_N$
17:    $frames \leftarrow new\_frames$
18:    $times \leftarrow new\_times$
19: end for
20: return frames, times
```

half along the symmetric axis. In the case of our current configuration this plane is defined by $x - y = -0.5$ but it can be computed for each scenario based on the arm's base location. We want to emphasize that we should only use this as a temporal solution since it is a problem specific, but adding the collision constraints on the whole arm would be more generalizable. In Figure 6, we show the trajectory for one of the arms with the "invisible wall" visualized in pink.

Our algorithm represents the advantages of having one controller for both arms. The path planning not only takes into account the other arm's location but actually plans both trajectories simultaneously. This leads to a successful, collision-free operation in a relatively small workspace. On the other hand, the overall controller is rather complicated and requires solving a series of optimization problems each time an arm needs to grab an object.

IV. Results

1) Separate controllers with assumptions on the workspace: The separate controllers without information about each other were very effective in moving objects from one bin to the next. However, they worked only because of the careful design of the workspace and the solution was tailored for this specific case. Adjusting our planner to another configuration will require designing the motion planning path anew and in many situations will not be feasible at all. On the other hand, having

TABLE I

A COMPARISON BETWEEN THE DIFFERENT TYPES OF CONTROLLERS:)

Method	Advantages	Disadvantages
Separate controllers with assumptions on the workspace	Simplicity	Will not work in most tasks and require tuning for every environment configuration
Separate controllers with communication channel	Simple controller but support cooperation	The amount of information shared is limited, not suitable for every task
Single controller	Enable full cooperation and coordination	Require solving a hard optimization problem requires direct joint control

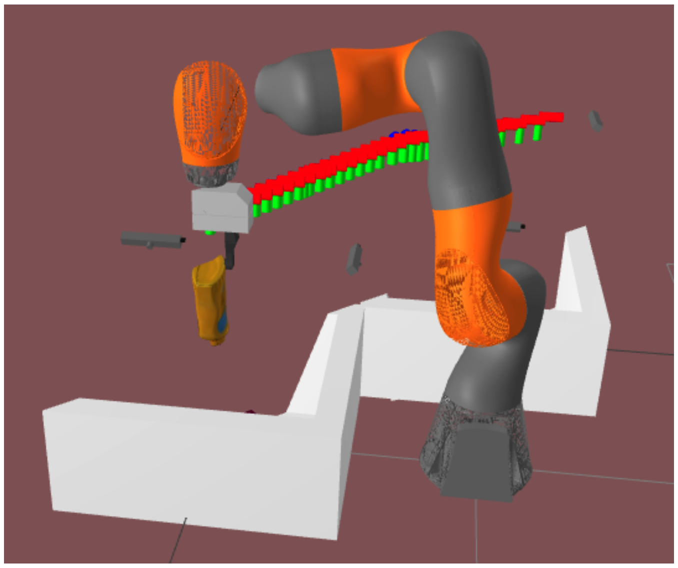

Fig. 6. A trajectory calculated by our path-planing algorithm, with the virtual wall visualized in pink

separate and simple controllers make debugging and adding new features easy, an important quality in system design. Although our system is able to perform the task successfully in most cases, there were still some failure cases. If objects are distributed unevenly, one arm may perform all of the work while the other remains idle. Additionally, since there must be some tolerance between the geofenced areas, there are dead areas where either robot will not grasp the objects placed there.

2) Separate controllers with communication channel: Regarding the third task, we tested our implementation on various objects from the YCB object dataset. We found that our controller achieves quite a reliable result, successfully passing the object from bin to bin in 26 out of 30 experiments. The failure cases were mainly problems during the object's passing from one gripper to another. While releasing the object and moving backward, the first gripper sometimes pushes the object and makes it fall from the second robot's grip. A more careful motion planning algorithm that considers the object's shape can remedy this problem. Another drawback of our algorithm is the time it requires to calculate the grasping pose. On our workstation, it took an average of 2.5 seconds to find a valid pose while the two arms were waiting in position. The main problem is for the calculation to begin, the arm with the object has to stop so that the cameras will see the object's position.

3) Single controller: The second task was successful in moving objects from one bin to the other with both arms executing the task at the same time and crossing paths. We successfully managed to remove the overhead of hand-crafting the geofenced areas since our new algorithm optimizes the paths for both arms at the same time to avoid collisions. In Figure 7, we can see two successfull trajectories for the bi-manual system that will allow them to move to the other bin without colliding. Nevertheless, as we mentioned before, we had to add some additional constraints to our optimization problem due to the fact that we performed position-based control and not direct joint control. Due to time constraints, we could not implement these but we believe that when these are added then this method should produce non-colliding paths for the dual-arm system without the geofencing constraints we introduced to make the demo. One drawback of our algorithm is that it can lead to some abrupt movements. In the middle of the trajectory, there sometimes is an abrupt bump to avoid a collision between both arms. There is no penalty for an overall smooth curve since each frame is optimized independently of the rest. Furthermore, this abrupt behavior is also seen in the timing of the trajectories; Again, our optimization has no view of the whole trajectory but plans each time step independently. We believe this could be improved by adding an additional layer of optimization that will take the whole trajectory into account and will refine all frames to avoid the mentioned issues and leaves it open for future work.

4) Comparison and Discussion: Table 1 summarizes the advantages and disadvantages of each method we checked. Our overall conclusion is when the information that each arm has to know about the other is limited is best to use separate controllers with a communication channel because it keeps each controller simple but still supports coordination. However, in the most common scenario where collision between the arm is a real risk, one controller with full information on the state of both arms is the best approach.

V. CONTRIBUTIONS

This project is divided in three main parts: 1) the separate controllers with assumptions on the workspace, 2) the separate controllers with communication channel and 3) the single controller. Alex was in charge of the separate controllers with assumptions on the workspace and coding the corresponding benchmark together with its corresponding part in the video and report. Idan was also in charge of the separate controllers

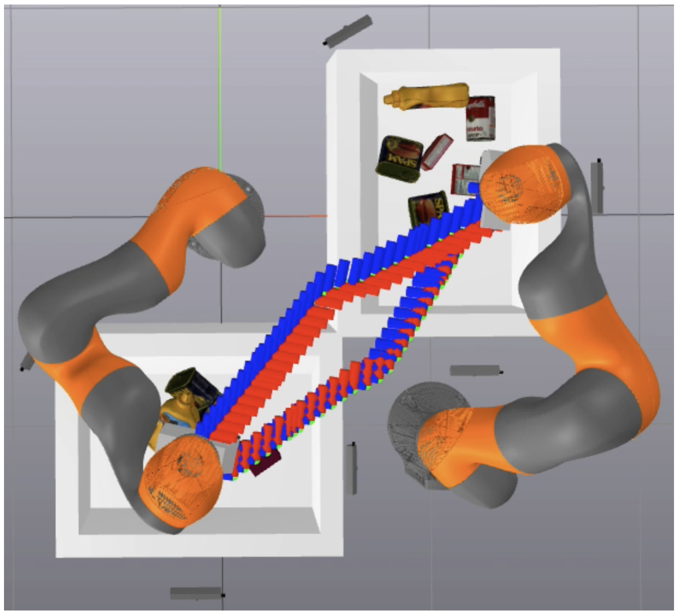

Fig. 7. An example of a successful trajectories for the dual arm robotic system.

with communication channel together with the associated benchmark and corresponding section in the presentation and report. In addition, Idan developed the proposed bi-manual grasping algorithm and the ideation was done together with Marcel. Marcel was in charge of the single controller also with its corresponding benchmark and section of the presentation and report. Moreover, Marcel was in charge of developing the algorithm we proposed for recursive bi-manual end-effector path planning optimization, and the ideation was done together with Idan.

References

[1] Driess, D., Ha, J.S. and Toussaint, M., 2020. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. arXiv preprint arXiv:2006.05398.

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

[3] Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X., Qi, P., Dimarogonas, D.V. and Kragic, D., 2012. Dual arm manipulation—A survey. Robotics and Autonomous systems, 60(10), pp.1340-1353.

[4] Krüger, J., Schreck, G. and Surdilovic, D., 2011. Dual arm robot for flexible and cooperative assembly. CIRP annals, 60(1), pp.5-8.

[5] Goertz, R.C., 1952. Fundamentals of general-purpose remote manipulators. Nucleonics, 10(11), pp.36-42.

[6] Choi, Y., Kim, D., Hwang, S., Kim, H., Kim, N. and Han, C., 2017. Dual-arm robot motion planning for collision avoidance using B-spline curve. International journal of precision engineering and manufacturing, 18(6), pp.835-843.

[7] Wong, C.C., Chien, S.Y., Feng, H.M. and Aoyama, H., 2021. Motion planning for dual-arm robot based on soft actor-critic. IEEE Access, 9, pp.26871-26885.

[8] Wang, J., Liu, S., Zhang, B. and Yu, C., 2019. Inverse kinematics-based motion planning for dual-arm robot with orientation constraints. International Journal of Advanced Robotic Systems, 16(2), p.1729881419836858.

[9] Dorri, A., Kanhere, S.S. and Jurdak, R., 2018. Multi-agent systems: A survey. Ieee Access, 6, pp.28573-28593.

[10] Ismail, Z.H., Sariff, N. and Hurtado, E.G., 2018. A survey and analysis of cooperative multi-agent robot systems: challenges and directions. In Applications of Mobile Robots (pp. 8-14). IntechOpen.

[11] Phan-huy-Hao, E., 1982. Quadratically constrained quadratic programming: Some applications and a method for solution. Zeitschrift für Operations Research, 26(1), pp.105-119.