ME 133a Final Report: Dual-Arm Ball-Setting Robot

Deon Petrizzo, Gio Huh, TK Lee

I. Introduction

In this project, we explore the implementation of a closed kinematic chain in RVIZ. Specifically, we implement a volleyball-setting robot, which mimics two human arms with hands together at all times. For simplicity, the hands are replaced with a circular paddle, which links the end-effectors of both arms together. Given a random ball trajectory, the arms position and orient the paddle to cancel the x- and y-components of the velocity upon impact to achieve purely vertical bouncing on the paddle. A particular focus was to leverage the redundancy of the system (the extra DOFS) to break the objective into multiple tasks, allowing the robot to prioritize its behavior in edge cases where the full behavior is not achievable.

II. Robot Description

We implement our objective on the two-arm setup of the Franka Emika Panda robot depicted in Fig 1. The arms are grounded to a tabletop and are positioned 1 unit distance apart from one another, with their positions being $(0, -0.5, 1)$ and $(0, 0.5, 1)$ expressed in the world frame (located at the center of the bottom surface of the table). We modified the URDF to include a circular paddle of radius $r_{\text{paddle}} = 0.15$ appended as a link to the end effector of the right arm. Note that we refer to the left arm as "arm 1" and the right arm as "arm 2" for the purposes of notation.

Each arm is 7-DOF, making the combined system 14-DOF. The workspace is limited by the tabletop and the constant relative position and orientation of the end effectors required to maintain the closed chain condition. So, the workspace is roughly half-ellipsoid-shaped.

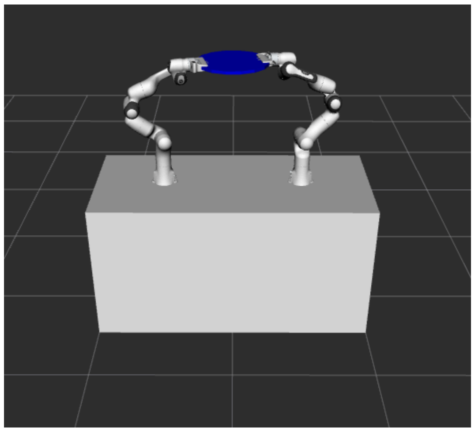

Fig. 1: Robot Starting Configuration (RVIZ)

III. Task Description

Given a ball trajectory with a randomized initial position and velocity, the robot attempts to complete the following tasks:

- Primary Task: Maintaining the closed chain between the two arms and the paddle (6 DOF)
- Secondary Task: Orienting the paddle normal such that the ball velocity is purely z-directed after collision with the paddle (2 DOF)
- Tertiary Task: Moving the center of the paddle to the point of impact (3 DOF)
- Quaternary Task: Maintaining natural joint angles that improve the condition of the current pose (3 DOF)

Our one hard requirement is to keep the hands on the paddle with constant relative position and orientation, so we therefore define this as our primary task. We then observe the tradeoff between orienting the normal of the paddle at collision and reaching the target position at collision. We note that the normal of the paddle fully determines the bounce motion of the ball, regardless of the location of the collision on the paddle. Thus, the normal orientation task is prioritized over the exact positioning of the paddle center to the ball. Finally, if all these tasks are achieved, we use the remaining DOFs to create a more "natural" joint configuration.

To avoid potential singularities the dual arm setter might encounter in its workspace, we apply a weighted inverse in the computation of the above tasks, given by

$$J^{\dagger} = \left(J^T J + \gamma^2 I_N\right)^{-1} J^T$$

Before we look into each task in more detail, we first define J_{v_1}, J_{ω_1}, J_{v_2}, J_{ω_2} to be the Jacobians of the first and second arm for position and orientation with respect to the world frame. We express them as expanded Jacobians J_{v_1}, J_{ω_1} are 3 by 14 matrices with zero padding for the 1st to 7th columns, and J_{v_2}, J_{ω_2} are 3 by 14 matrices with zero padding for the 8th to 14th columns.

A. Primary Task

Let R_1, R_2 be the rotation matrix of the first and second arm relative to the world frame. Let p_1, p_2 be the position of the two arms with respect to the world frame. It then follows that the position and orientation of the first arm relative to the second arm are respectively given by

$$p_{12} = R_2^T (p_1 - p_2)$$
$$R_{12} = R_2^T R_1$$

From this, we compute the position and orientation Jacobians of the primary task, "12" denoting that we are expressing changes in position and orientation of the first arm's tip w.r.t. the second arm's tip (frame located at paddle center).

$$J_{v_{12}} = R_2^T \left(J_{v_1} - J_{v_2} + [p_1 - p_2]_\times J_{v_2} \right)$$
$$J_{\omega_{12}} = R_2^T \left(J_{\omega_1} - J_{\omega_2} \right)$$

The Jacobian of the primary task is then

$$J_p = \begin{pmatrix} J_{v_{12}} & J_{\omega(12)} \end{pmatrix}^T$$

We then get the desired position and orientation of the primary task, $p_{d_{12}}$, $R_{d_{12}}$, using information from the URDF:

$$p_{d_{12}} = \begin{pmatrix} 0 & 0 & r_{\text{paddle}} \end{pmatrix}^T$$
$$R_{d_{12}} = \text{Rot}_x\left(\tfrac{\pi}{2}\right)$$

We then use velocity inverse kinematics to find the desired joint velocities, $\dot{q_p}$.

$$e := \begin{pmatrix} e_p(p_{d_{12}}, p_{12}) & e_R(R_{d_{12}}, R_{12}) \end{pmatrix}^T$$
$$\dot{x_d} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T$$
$$J_p = \begin{pmatrix} J_{v_{12}} & J_{\omega_{12}} \end{pmatrix}^T$$
$$\dot{q_p} = J_p^\dagger (\dot{x_d} + \lambda e)$$

B. *Secondary Task*

For the secondary task, we want the paddle's normal to align with some desired normal upon ball collision. From the URDF of the system, we get the final desired normal, $\hat{n}_f$, from the first column of R_2, the x-axis of the rotation matrix of the paddle w.r.t. the world frame.

Given the final desired normal, we can then use an angular spline, $\alpha(t), \dot{\alpha}(t)$, that goes from 0 to θ in a given time 0 to T, where for a starting normal of the paddle, $\hat{n}_0$, we know

$$\theta = \arccos\left(\hat{n}_0^T \hat{n}_f\right)$$

Since the normal is defined by the y- and z-axes of the paddle frame, we define a linear transformation for orienting the normal, ignoring rotation about the paddle frame's x-axis:

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

It then follows that the Jacobian for the secondary task is

$$J_s = A R_2^T \left(J_{w_2} \right)$$

Let e_n be the error of the current normal of the paddle $\hat{n}$ relative to the desired normal $\hat{n}_d$. Then,

$$\hat{a} = \frac{\hat{n}_0 \times \hat{n}_f}{\sin(\theta)}$$
$$\hat{n}_d = \text{Rot}_n(\hat{a}, \alpha(t)) \cdot \hat{n}_0$$
$$\omega_d = \dot{\alpha}(t) \cdot \hat{a}$$
$$e_n = \hat{n} \times \hat{n}_d$$

For rotation about $\hat{a}$, the secondary task velocity is then given by

$$\dot{x}_r = A R_2^T (\omega_d + \lambda e_n)$$

Let J_{s_2} be a Jacobian associated with the 8th and 14th columns of J_s, in other words, the second arm. Thus, the joint velocity of the second arm is

$$\dot{q_{s_2}} = \left(J_{s_2} \right)^\dagger \dot{x_r}$$

We then must find the corresponding joint velocity, $\dot{q_{s_1}}$, of the first arm for both arms to work together in achieving the second task. Notice that, to do this, both arm's joint velocities must first obey the requirements of the primary task. As such, for J_{p_1}, J_{p_2}, the 1st to 7th columns (first arm) and the 8th to 14th columns (second arm) of the Jacobian of the primary task, it then follows that

$$J_{p_1} \dot{q_{s_1}} + J_{p_2} \dot{q_{s_2}} = 0$$
$$J_{p_1} \dot{q_{s_1}} = -J_{p_2} \dot{q_{s_2}}$$
$$\dot{q_{s_1}} = -\left(J_{p_1} \right)^\dagger J_{p_2} \dot{q_{s_2}}$$
$$\dot{q_{s_1}} = -\left(J_{p_1} \right)^\dagger J_{p_2} \left(J_{s_2} \right)^\dagger (\dot{x_r})$$

Since $\dot{q}_s$ is the concatenation of $\dot{q}_{s_1}$ and $\dot{q}_{s_2}$ and we know $\dot{q}_{s_1}$ and $\dot{q}_{s_2}$ in terms of $\dot{x}_r$, we then know that

$$\dot{q_s} = \begin{pmatrix} -\left(J_{p_1}\right)^\dagger J_{p_2} \left(J_{s_2}\right)^\dagger \\ \left(J_{s_2}\right)^\dagger \end{pmatrix} \dot{x_r}$$

C. *Tertiary Task*

Since the tertiary task is moving the paddle to a desired final position, we additionally define a spline that returns position and velocity p_d, v_d from the initial position of the paddle p_0 to the final p_f for a given time T.

The Jacobian of the tertiary task, $J_t = J_{v_2}$ and the error $e_p = p_d - p_2$. We then find the joint velocities, $\dot{q_{t_2}}$, for the second arm to be

$$\dot{x_r} = \left(v_d + \lambda e_p \right)$$
$$\dot{q_{t_2}} = \left(J_{t_2} \right)^\dagger \dot{x_r}$$

where J_{t_2} is a Jacobian associated with the 8th and 14th columns of J_t, in other words, the second arm.

By a similar argument to the secondary task derivation for the joint velocities, we find the joint velocities of the first arm, $\dot{q_{t_1}}$, to be

$$\dot{q_{t_1}} = -\begin{pmatrix} J_{p_1} \\ J_{s_1} \end{pmatrix}^\dagger \begin{pmatrix} J_{p_2} \\ J_{s_2} \end{pmatrix} \dot{q_{t_2}}$$

$$\dot{q_{t_1}} = -\begin{pmatrix} J_{p_1} \\ J_{s_1} \end{pmatrix}^\dagger \begin{pmatrix} J_{p_2} \\ J_{s_2} \end{pmatrix} \left(J_{t_2} \right)^\dagger \dot{x_r}$$

Thus, we know the concatenated joint velocities of the arms, $\dot{q}_t$, are

$$\dot{q_t} = \begin{pmatrix} -\begin{pmatrix} J_{p_1} \\ J_{s_1} \end{pmatrix}^\dagger \begin{pmatrix} J_{p_2} \\ J_{s_2} \end{pmatrix} \left(J_{t_2}\right)^\dagger \\ J_{t_2}^\dagger \end{pmatrix} \dot{x_r}$$

D. *Quaternary Task*

Finally, for the quaternary task, we define a custom cost function $-\nabla C(q)$ that penalizes the joint positions of each arm. Ideally, we would like the condition number of our robot to be relatively low throughout operation, with little fluctuation. By choosing the desired joint positions q_d to match those of the starting pose—which was chosen to have a good condition number and to look similar to the natural arm configuration of a volleyball setter—the quaternary task then becomes

$$\dot{q}_q = -\lambda_q \nabla C(q) = \lambda_q (q_d - q).$$

IV. Algorithms and Implementation

A. Inverse Kinematics

In order to enforce the hierarchy of importance of each task, we ensure that lower-priority tasks do not interfere with higher-priority ones by mapping the joint velocities of less important tasks into the null space of all preceding ones. For our four-task system defined above, the inverse kinematics formula that achieves this behavior is

$$\dot{q_d}(t_k) = \dot{q}_p + \left(I - {J_p}^{\dagger} J_p\right)\dot{q}_s + \left(I - \begin{pmatrix} J_p \\ J_s \end{pmatrix}^{\dagger} \begin{pmatrix} J_p \\ J_s \end{pmatrix}\right)\dot{q}_t$$
$$+ \left(I - \begin{pmatrix} J_p \\ J_s \\ J_t \end{pmatrix}^{\dagger} \begin{pmatrix} J_p \\ J_s \\ J_t \end{pmatrix}\right)\dot{q}_q$$

where $\dot{q}_i$ are functions of their respective task coordinates.

We then perform forward Euler to get the desired joint angles q_d:

$$q_d(t_k) = q_i(t_{k-1}) + \dot{q}_d(t_k)\ dt.$$

The projection of the joint velocity of the secondary task into the null space of the primary is well-known in the aforementioned expression. We further explore the generalization of the projection of the joint velocity of lower-level tasks in the appendix.

B. Paddle Impact Selection

To generate the ball's trajectory, we randomly select an initial position p_0 and velocity v_0. Additionally, we experimentally restrict the range of possible values of the x, y, and z components of both variables such that the ball trajectory under most circumstances coincides with the workspace of the robot.

From p_0, v_0, we then would like to select the final position p_f, which is also the point of impact, using a heuristic approach. It would be ideal for the robot to move minimally from its initial configuration to the point of impact.

Let p_c be the starting position of the paddle of the robot with respect to the world frame and g be the gravity constant in the simulation. Additionally, let $p_0 = (x_0, y_0, z_0)$ and $v_0 = \left(v_{0_x}, v_{0_y}, v_{0_z}\right)$. We find

$$t^* = \text{argmin } d(t)$$

where

$$d(t) = \|p(t) - p_c\|$$
$$p(t) = \left(v_{0_x}t + x_0, v_{0_y}t + y_0, v_{0_z}t + z_0 - \tfrac{1}{2}gt^2\right)$$

Since there can be multiple solutions, we apply a root-finding algorithm to one possible solution of t^*. Thus, we find

$$p_f = p(t^*)$$

If, however, the z-component of p_f is below the height of the table, the dual arms are limited in reaching the ball's impact position. Thus, we alternatively find t^* such that the z-component of $p(t)$ is equivalent to the height of the table.

The summarized algorithm can be found below:

1) Solve for $t^* = \text{argmin } d(t)$
2) If $p(t^*)$ is above the table, $p_f = p(t^*)$.
3) Otherwise, find t^* s.t. the z-component of $p(t^*)$ is equivalent to the height of the table and $p_f = p(t^*)$.

C. Ball Collision Detection

The algorithm for determining if the paddle has hit the ball is as follows:

1) Let R be the radius of the circular paddle, $\hat{n}$ be normal to the paddle face, and $\vec{p}_b$ and $\vec{p}_p$ be the position of the ball and paddle in the world frame, respectively.
2) Compute the distance between the ball and the paddle, $\vec{r} = \vec{p}_b - \vec{p}_p$.
3) If $\hat{n} \cdot \vec{r} = 0$, then the ball is in plane with the paddle. If $\|\vec{r}\| \leq R$, then the ball is within the circular boundary of the paddle. If both conditions are met, then the ball has collided with the paddle.
5) Repeat the check every dt.

Moreover, if the ball's position is below the floor or within the x and y bounds of the table and is less than the height of the table, we invert the sign of its z-component.

D. Ball Velocity Cancelation in x and y

We perform the following calculation to determine the desired paddle normal $\hat{n}_d$ that results in purely z-directed ball motion upon collision with the paddle:

1) Let $\hat{v}_0$ be the normalized ball velocity one time step dt before the collision with the paddle. Let $\hat{v}_f = (0, 0, 1)^T$ be the desired normalized velocity of the ball.
2) Compute the desired normal as $\hat{n}_d = \frac{\hat{v}_d - \hat{v}_0}{\|\hat{v}_d - \hat{v}_0\|}$

Notice that this makes intuitive sense given the law of angle of reflection. Geometrically, $\hat{v}_d - \hat{v}_0$ represents the direction of the normal that bisects the angle between the incident $\hat{v}_0$ and reflected $\hat{v}_d$.

E. Updating Ball Velocity

If the ball collides with the paddle, we compute the new ball velocity $\vec{v}_d$ as the difference between the component in the plane of the paddle, $\vec{v}_{\|}$, and the component perpendicular to the paddle, $\vec{v}_{\perp}$, since geometrically, $\vec{v}_d$ would be the reflection of the incoming velocity $\vec{v}$ across $\vec{v}_{\perp}$, pointing away from the paddle surface.

$$\begin{aligned} \vec{v}_d &= \vec{v}_{\|} - \vec{v}_{\perp} \\ &= (\vec{v} - \vec{v}_{\perp}) - (\vec{v} \cdot \hat{n})\hat{n} \\ &= \vec{v} - 2(\vec{v} \cdot \hat{n})\hat{n} \end{aligned}$$

V. Evaluation of Task Ordering

To justify our task hierarchy, we compare the effect of prioritizing the normal matching task over the positioning task (our proposed ordering) with the effect of the opposite task ordering. We also compare the operation of the robot with and without the quaternary task, to see if the quaternary task really does result in a more "natural" joint position (and better condition number).

A. Secondary Task and Tertiary Task Ordering

To highlight the differences in behavior between the two secondary and tertiary task orderings, we make observations of the dual arm system near a singularity, where it is not necessarily possible to complete both the positioning and normal tasks perfectly. Recall, the goal of our system is to cancel the x- and y- components of the ball's velocity (w.r.t. the world frame) upon collision with the paddle, so we can bounce the ball up and down on the paddle. Thus, the ball velocity error, $e_{\text{ball}} = \sqrt{v_x^2 + v_y^2}$, is a good metric for measuring the effectiveness of both task orderings, while operating near the singularity. Additionally, we compare the error in the position reached by the paddle at the time of collision, which we define as the difference between the actual collision location and the center of the paddle. The following table captures these errors when positioning is prioritized over the normal task and vice versa:

TABLE I: **Secondary & Tertiary Task Error at Ball Impact**

Error Type	**Normal over Position**	**Position over Normal**
Norm of Ball's x, y Component Velocity	**0.4334e-2**	0.4591
Position of Paddle	**0.5288e-1**	0.2531

According to Table 1, when the normal task is prioritized, the velocity error is reduced by approximately 99.06% and the positional error also decreases by 79.11% compared to the opposite task ordering. There does not seem to be a clear tradeoff between the errors when switching the order of secondary and tertiary tasks; rather, both errors are smaller when the normal task is prioritized. What could point to this result is the fact that the final joint of the arm is a wrist joint that provides a roll to the paddle orientation. Thus, the task of reaching the position is relatively orthogonal to that of aligning the normal—when the configuration reaches the desired paddle position, the final wrist joints can rotate in place freely without sacrificing position.

The large discrepancy in the velocity error agrees with the behavior of the ball's motion after the paddle collision.

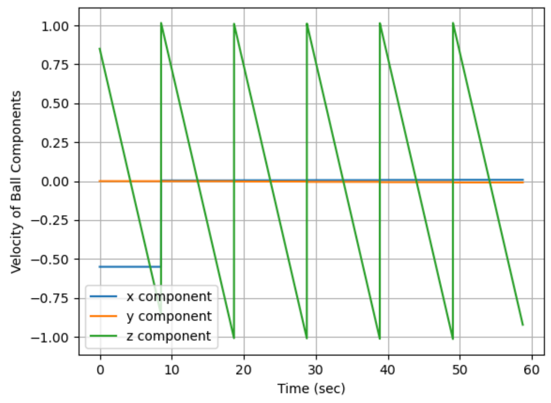

Fig. 2: **Velocity of Ball Over Time (Normal Prioritized Over Position)** The x and y components of the ball velocity are $v_x \backsimeq 0.0, v_y \backsimeq -0.54$ before the collision with the paddle, while the they become $v_x \backsimeq 0.01, v_y \backsimeq 0.0$ after collision. The collision happens near t = 8.0s.

For the case when the normal task is prioritized over positioning, the x and y ball velocities are effectively zeroed out after t = 8.0s (time of collision), only leaving the modulating z velocity, as desired.

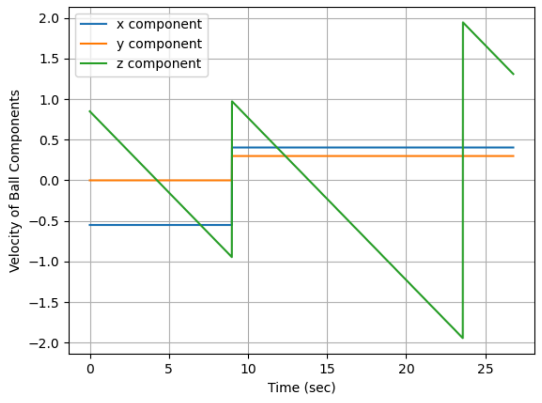

Fig. 3: **Velocity of Ball Over Time (Position Prioritized Over Normal)** The x and y components of the ball velocity are $v_x \backsimeq 0.0, v_y \backsimeq -0.54$ before the collision with the paddle, while the they become $v_x \backsimeq 0.4, v_y \backsimeq 0.3$ after collision. The collision happens near t = 8.0s.

However, when the positioning task is prioritized over normal, the x and y components of the ball velocity still remain non-zero after t = 9.0 s (time of collision). Moreover, notice that the number and frequency of z-velocity spikes show how the smaller velocity error ensures that the ball continues bouncing on the paddle as desired, whereas Fig 3. clearly suggests that the error is too large, so the ball only bounces off the paddle once before falling to the floor (see video for more visuals).

Thus, in agreement with our intuitive evaluation in the Task Description section, there is a clear advantage in the proposed ordering of the secondary and tertiary tasks.

B. Adding the Quaternary Task

Now we compare the behavior with and without the quaternary task. Namely, we compare the condition number and visually evaluate which case leads to a more natural pose.

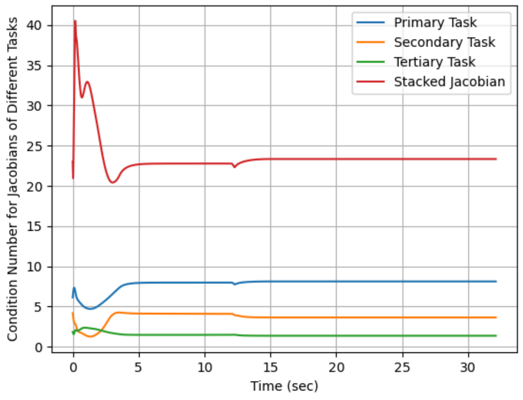

Fig. 4: **Condition Number of Jacobians Across Tasks (Including Quaternary Task)** The condition number of the Jacobian obtained by stacking the primary, secondary, and tertiary Jacobian starts near 21, increases to 40 near 0.5s, decreases back to 21 at 3.5s, before settling to roughly 23 in the final pose.

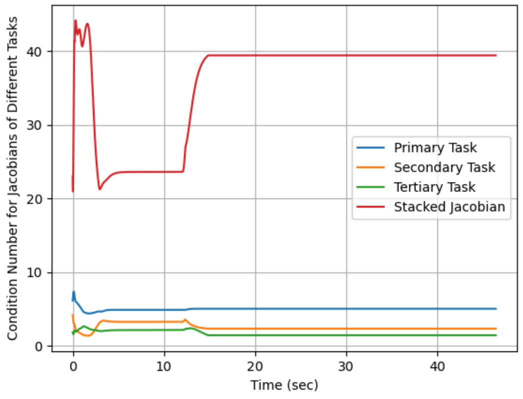

Fig. 5: **Condition Number of Jacobians Across Tasks (Excluding Quaternary Task)** The condition number of the Jacobian obtained by stacking the primary, secondary, and tertiary Jacobian starts near 21, increases to 45 near 2.0s, decreases back to 21 at 3.5s, and increases back to a value near 40 in the final pose.

To holistically analyze the effect of the additional quaternary task, we focus on the condition number of the total Jacobian of the system—the primary, secondary, and tertiary Jacobians vertically stacked. According to Fig. 4 and 5, the condition number of this Jacobian is roughly halved (roughly 23 vs 40) after the robot achieves its final configuration if the quaternary task is included.

For a more numerical comparison, we acquired the average and standard deviation of the condition number over time.

TABLE II: Analysis of Condition Number of Stacked Jacobian

Metric related to Condition Number	**Quaternary Task Included**	**Quaternary Task Excluded**
Average	**23.60**	35.81
Standard Deviation	**2.352**	6.684

Table 2 confirms that when the quaternary task is included, it results in a better overall configuration for the system as it has a lower condition number by 34.10% compared to when the quaternary task is excluded. Qualitatively, the numerical results agree with Figure 6 and 7. Namely, the joint positions for Figure 6 more closely resemble the natural human arm configuration when receiving a ball compared to Figure 7, which shows a much more awkward, contorted configuration due to the unnecessary rotation of the paddle about its normal.

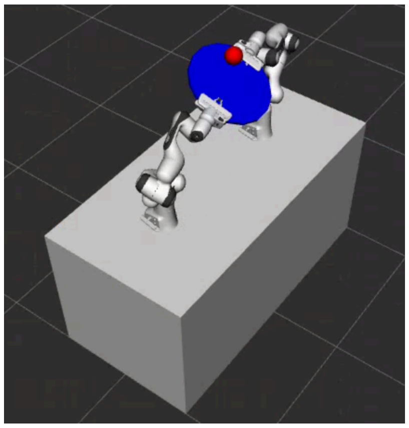

Fig. 6: **Configuration of Dual Arms with Quaternary Task at Ball Impact** In this arm configuration, the elbows are stretched outwards of each other, resembling an identical pose of when a human is receiving a ball.

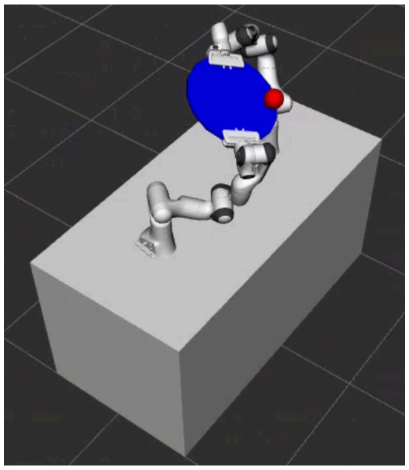

Fig. 7: **Configuration of Dual Arms without Quaternary Task at Ball Impact** In this arm configuration, the wrists are twisted, which is not optimal because it is possible to meet the ball in the same position without the wrist rotation.

VI. Conclusion

The dual-arm volleyball-setting robot demonstrates the effective use of redundancy in the system to prioritize tasks and manage edge cases. By implementing a four-task hierarchy, from maintaining the closed kinematic chain to achieving natural joint configurations, the robot can effectively set balls vertically, even near singularities. Prioritizing the paddle's orientation over its precise positioning significantly reduced velocity errors along the x and y axes, ensuring the desired ball trajectory. Moreover, incorporating the quaternary task to optimize joint configurations reduced the condition number of the system, resulting in a more natural human-like final pose.

Despite its successful performance, limitations such as workspace constraints of the fixed table-top positions of the arms (resulting in a nearly half-ellipsoid workspace for position) still exist. If the URDF were adjusted so that the arms were attached to a side face instead, the system would more realistically mimic the way human arms are configured. It might also be interesting to compare this system with one in which all task coordinates are governed by a single combined Jacobian. The choice of cost function for the "natural configuration" task is also completely arbitrary. Exploring different cost functions could lead to better behavior.

VII. Appendix

A. Code Appendix

For the full code base, see Github. Note, the `final.py` file contains the bulk of the implementation described here.

B. Math Appendix

For completeness, here is a longer derivation of the null space projection formula used in the inverse kinematics.

Let $\dot{q_{i_n}}$ be the projection of $\dot{q_i}$ for ith task into the null spaces of the Jacobians above it. Let J_i be the Jacobian of the ith task.

We show for the $i = 2$ case. Thus, we know that

$$J_1 \dot{q_{2_n}} = 0$$

while minimizing the norm between $\dot{q_{2_n}}$ and $\dot{q_2}$.

Our goal is then to λ_1

$$\dot{q_{2_n}} = \text{argmin } \frac{1}{2} \left\| \dot{q_{2_n}} - \dot{q_2} \right\| - \lambda_1^T (J_1 \cdot \dot{q_2})$$

Taking the derivative to $\dot{q_{2_n}}$, we get

$$\begin{aligned} \left(\dot{q_{2_n}} - \dot{q_2}\right)^T - \lambda_1^T J_1 &= 0 \\ \left(\dot{q_{2_n}} - \dot{q_2}\right) - J_1^T \lambda_1 &= 0 \\ \dot{q_{2_n}} &= \dot{q_2} + J_1^T \lambda_1 \end{aligned}$$

Additionally, since $J_1 \dot{q_{2_n}} = 0$, we know

$$\begin{aligned} J_1 \dot{q_{2_n}} &= 0 \\ J_1(\dot{q_2} + J_1^T \lambda_1) &= 0 \\ J_1 \dot{q_2} &= -J_1 J_1^T \lambda_1 \\ -(J_1 J_1^T)^{-1} J_1 \dot{q_2} &= \lambda_1 \end{aligned}$$

Substituting the λ_1, we then get

$$\begin{aligned} \dot{q_{2_n}} &= \dot{q_2} + J_1^T \lambda_1 \\ \dot{q_{2_n}} &= \dot{q_2} - J_1^T (J_1 J_1^T)^{-1} J_1 \dot{q_2} \\ \dot{q_{2_n}} &= \dot{q_2} - J_1^\dagger J_1 \dot{q_2} \\ \dot{q_{2_n}} &= \left(I - J_1^\dagger J_1\right) \dot{q_2} \end{aligned}$$

For the $i = 3$ case, we can equivalently state for λ_2

$$\begin{aligned} \dot{q_{3_n}} &= \text{argmin } \frac{1}{2} \left\| \dot{q_{3_n}} - \dot{q_3} \right\| - \lambda_1^T (J_1 \cdot \dot{q_3}) - \lambda_2^T (J_2 \cdot \dot{q_3}) \\ &= \text{argmin } \frac{1}{2} \left\| \dot{q_{3_n}} - \dot{q_3} \right\| - \lambda^T J \cdot \dot{q_3} \end{aligned}$$

where λ is the concatenation of λ_1 and λ_2 and $J = \begin{pmatrix} J_1 \\ J_2 \end{pmatrix}$.

Thus, by the symmetric argument of $i = 2$, it then follows that

$$\dot{q_{3_n}} = \left(I - \begin{pmatrix} J_1 \\ J_2 \end{pmatrix}^\dagger \begin{pmatrix} J_1 \\ J_2 \end{pmatrix}\right) \dot{q_3}$$

By adding the additional constraint of $J_3 \cdot \dot{q_4} = 0$ and applying the same line of reasoning as the $i = 3$ case, we then know that

$$\dot{q_{4_n}} = \left(I - \begin{pmatrix} J_1 \\ J_2 \\ J_3 \end{pmatrix}^\dagger \begin{pmatrix} J_1 \\ J_2 \\ J_3 \end{pmatrix}\right) \dot{q_4}$$