CS 225A: Final Report
Bend It Like Panda: Penalty Kick Taking Robot

Christina Li, Ron Domingo, Dahlia Radif, Esteban Mejia

June 11, 2019

1 Abstract

This report reviews the penalty kick taking robot, focusing on the implementation and functionality of the system as well as the main challenges and limitations posed. The robot used in this project is the Franka Emika Panda robot arm. It is modified with a personally designed end effector modelled as a human foot in order for the motion to more closely resemble a penalty kick. The controller allows for three main kicking motions- aiming to the left, middle, and right of the goal, and computer vision has been integrated with the controller. In this way, the robotic arm is able to detect an obstacle (i.e. a goalie) in the goal, and can choose the direction in which to kick accordingly.

This project lies in the intersection of many aspects of robotics that allowed us to explore different methodologies in order to create a working kick. Simulating a kicking motion is only the first step; working within the limitations of a robotic arm, generating enough momentum to be able to kick a ball towards a goal, and incorporating vision into the system all add complexity to what appears to be a simple motion. There is also balance to be found between simplifying the requirements of the kick and workspace in order to achieve a working controller, and over-reducing the complexity of the task. In other words, this project introduced a wide array of decisions that had to be made in order to successfully kick a ball within the physical limitations of the robot. For example, the success of the kick is sensitive to the setup of the workspace and the alignment of the ball with the end effector; however, we have been able to simulate three working directions in which to kick with the same workspace setup, which adds robustness to the task. This project has therefore allowed for flexibility when attempting to achieve a working kick, with many different approaches being possible; with this flexibility also comes compromise in selecting the correct approach: one that works consistently without oversimplifying the task.

Many challenges were encountered throughout this project, ranging from unsuitable trajectories, inadequate force being applied to the ball, and difficulties in establishing a suitable workspace setup. Overcoming these hurdles was paramount to creating a consistent and robust controller.

We will discuss in detail the methodology taken in order to achieve a final penalty kick, describing the final controller integrated with computer vision, and the intermediate attempts leading up to the working solution. In addition, we will describe the workspace and end effector setup, and how this interacts harmoniously with the controller. We will also explain the various modifications made to the task in order to overcome the challenges encountered.

2 Final Implementation

The implementation of this project is encompassed in four overarching methods; creating a working controller by generating a working trajectory in simulation and testing this controller on the robot arm, incor-

porating computer vision, designing a suitable end effector, and modifying the workspace to work simultaneously with the controller. In this section, we will describe the intermediate steps taken in order to reach the final controller and workspace setup.

In reaching our final strategy for trajectory generation, we first simulated and tested several options, each of which were very different. The first of these was a polynomial trajectory in Cartesian space with an operational space controller for position and orientation of the end-effector with dynamic decoupling. We designed this polynomial to pass smoothly through three points: the initial position of the end effector, the point of contact with the ball, and a follow through point shortly after contact. We found this strategy to lack both the power and accuracy needed to successfully kick the ball, so we modified our approach towards a method over which we had more direct control. An example plot of one of the 3D trajectories from this polynomial strategy is below.

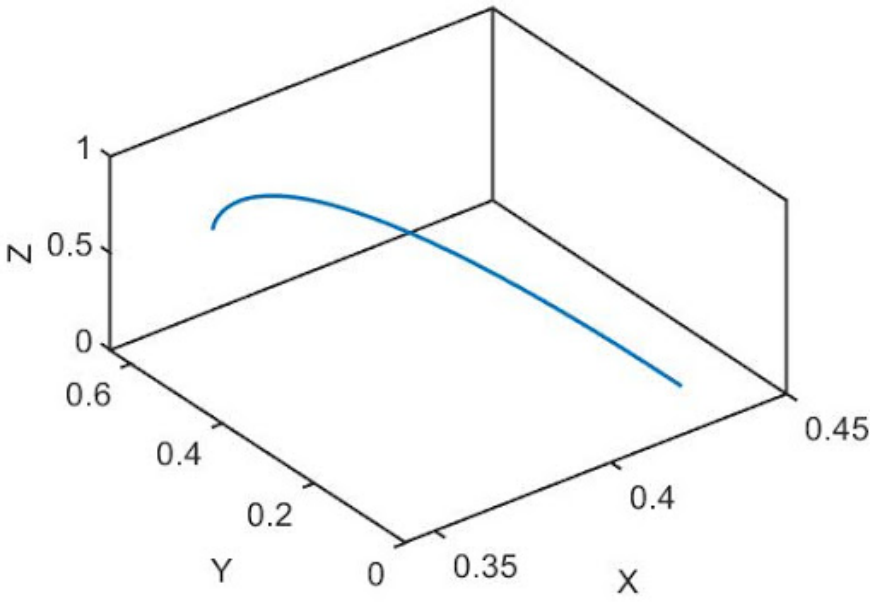

Figure 1: 3D Cartesian Space Trajectory

Testing the polynomial trajectories on the physical robot came with the realization that we were not making full use of many of the joints. Since our task, i.e. hitting a ball, involves generating significant momentum, we decided to move away from Cartesian space and towards simple joint control. We started by only moving the first robot joint and fixing the others at a particular configuration which placed the end effector low enough to make contact with the ball. By doing this and only moving the first joint, we effectively made the robot sweep the end effector in a semicircle with the point of contact between the end effector and the ball lying halfway through the trajectory. We found mild success with this approach - it lacked the power required to be a convincing kick but gave us fewer issues regarding velocity limits and disjointed motion, and this was enough motivation to develop our approach using joint controls. From this point forward, we used joint task controllers with dynamic decoupling.

With guidance from the teaching staff, we found a trajectory that seemed to work well in three main aspects; generating as much power as the hardware would allow, hitting the ball reliably, and being able to tweak the trajectory to aim the kick appropriately. This trajectory is best described as a volley - the robot arm starts curled up behind the ball and quickly unfolds to make contact with the ball, follows through the point of contact, and returns to a resting position for another kick. To achieve smooth trajectories between the different joint configurations, we use simple linear interpolation at the maximum allowable speed in conjunction with a state machine to transition between the different parts of the trajectory (preparation,

kick, follow through, return). In this way, we can slow down the robot after contact with the ball has been made, and return slowly to a configuration that lies far from joint limits. The joint trajectories are below, each showing their respective joint limits. All linear interpolation follows the same equation:

$$q_i = q_o + \frac{(q_f - q_o)t_i}{\Delta t} \tag{1}$$

where q_i is the interpolated joint angle, q_o is the starting joint angle, q_f is the ending joint angle, t_i is the current time, and Δt is the time between starting and ending joint configurations.

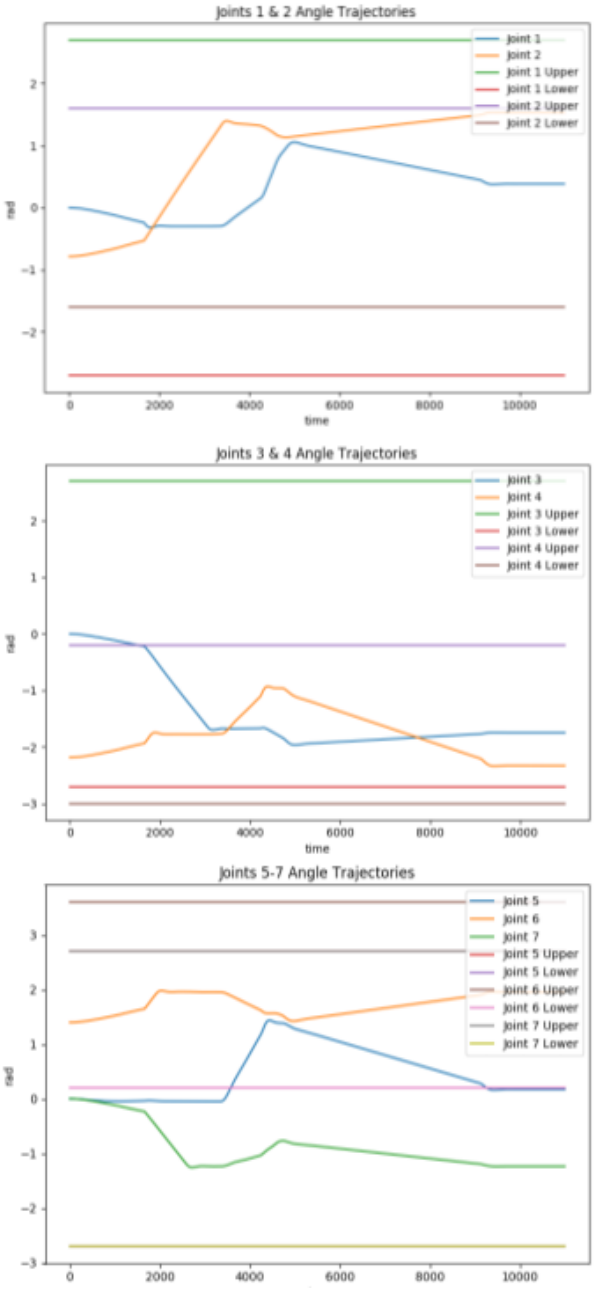

Figure 2: Joint trajectories and their respective upper and lower bounds given our joint controller with linear interpolation between joint configurations.

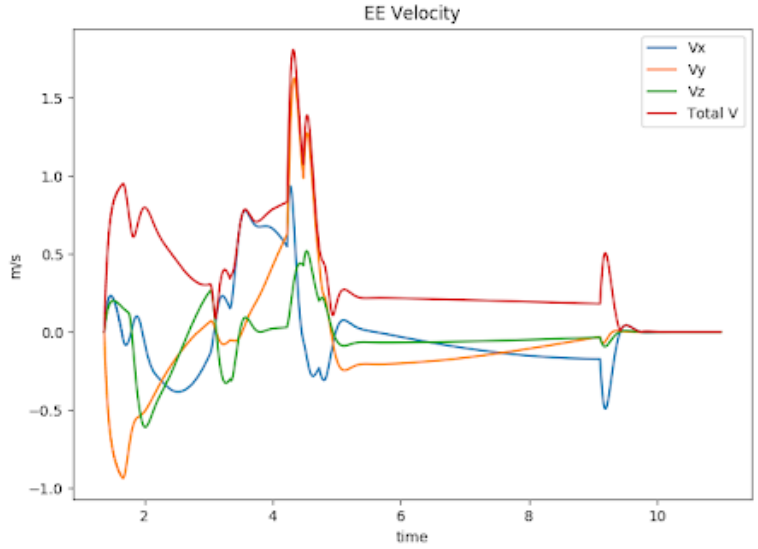

Figure 3: End-effector velocity over the course of a kick aiming to the left of the goal. At the peak, the total velocity of the end-effector reaches 1.67 m/s, pushing against the hardware limitation imposed of 1.7 m/s.

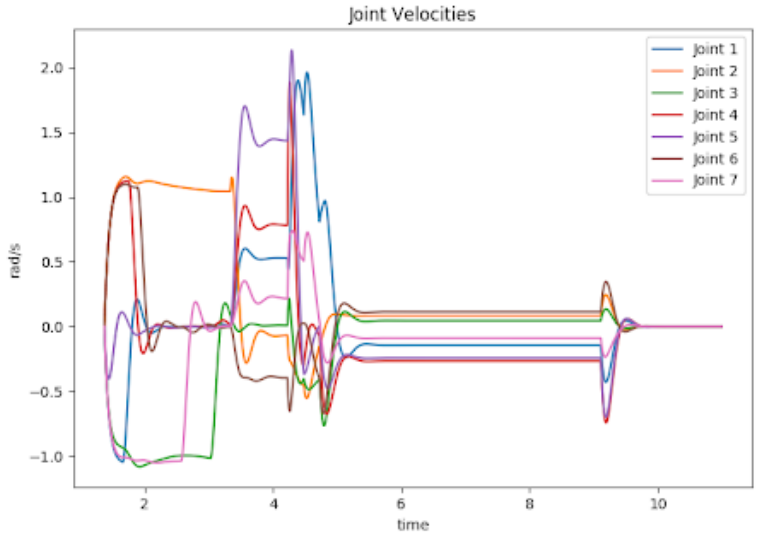

Figure 4: Individual joint velocities over the course of a swing. The maximum allowed joint velocity at any joint is $\pm$ 2.0 rad/s and while the simulation shows that joint 5 exceeds this value, this is not the case when running on the real robot.

After we were able to aim accurately, we added a user interface allowing for selection of a particular direction in which to aim. Afterwards, we replaced this manual selection with integrated vision whereby the robot can decide for itself where exactly to kick, based on the presence of a goalie. For the vision component, we decided to use OpenCV in Python for quick testing and integration. Using a simple USB webcam, we look for a certain range of HSV color values that correspond to the neon yellow/green border of the goal, then mask it so only the target blob appears. We pass it through some erosion and dilation filters to

get rid of any blobs that might be false positives. Next, the vision algorithm finds the biggest clump of that color to draw a bounding box and circle around it. In this way, when testing the values for the color ranges, we can see exactly where the algorithm thinks the goal is. Since the algorithm looks for the biggest clump of continuous color, when we block part of the goal with the paper goalie, it changes the effective goal size and location based on which part (left, center, or right) is covered. Finally, based on the center point of the largest bounding box, we decide whether to shoot left, right, or center. We send that as a Redis key back to the controller, and the controller then takes the key as input to a state machine to shoot correspondingly.

Figure 5: The output of the computer vision program on different goalie configurations.

3 Challenges

In this section, we will outline and discuss the various challenges encountered throughout the project, and the ways that we solved these challenges in order to achieve a successful controller.

Firstly, in the initial vision component, our penalty kick robot would be able to detect any small opening and effectively shoot the ball with great precision to any point within the goal. However, it was not far into the implementation process that we realized that our vision would not be possible with the robot available to us. Therefore, instead of having a broad range of possible directions, we decided to have our vision determine the aiming direction between left, center, and right based on where the goalie obstructs the goal.

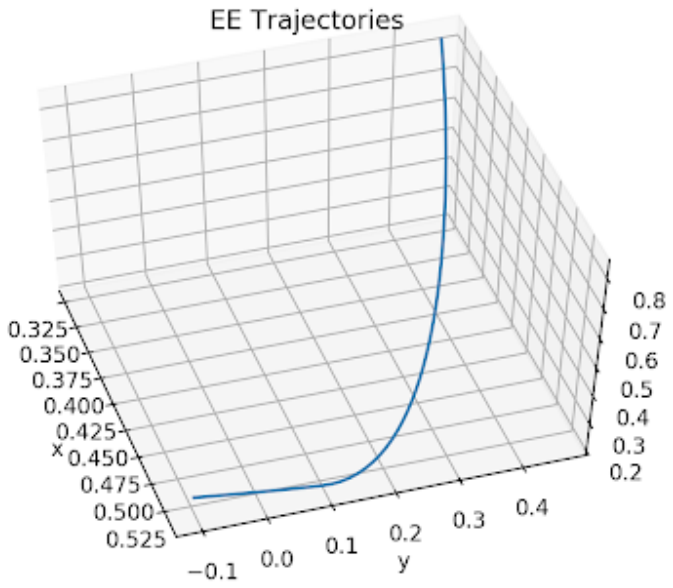

Figure 6: Original end-effector trajectory when utilizing an operational space controller.

The first iteration of the penalty kick robot utilized an orientation space controller that tracked a desired kicking trajectory with the end effector. The generated trajectory was intended to be a parabolic swing in 3D space at the bottom of which lies the contact point of the ball, as shown in Figure 6. We used 3 Cartesian coordinates, an initial, final, and intermediate velocity, and a start and end time to generate a 3rd order polynomial equation that defined the parabolic swing. In simulation, the trajectory was promising and enabled the robot to somewhat mimic a swing of the foot in a real penalty kick. In practice, however, the trajectory featured discontinuities that caused jerky and inefficient motions of the robotic arm during the swing. This resulted in redundant motion, and a lack of adequate momentum. We tried continually to improve our orientation space controller but ran into other issues, such as an inability for the robot to consistently reach its initial position. We also had to deal with spikes in torques or joint velocities due to the jerky movements from the trajectory. Eventually we decided to rethink our controller in order to get the motion that we desired, instead of spending too much time trying to fix an approach that we might have decided against regardless.

Figure 7: End-effector orientation upon contact with the ball.

Our next iteration involved a joint space controller that moved the robot between two joint configurations. Our controller used linear interpolation travel between the initial and final joint values, allowing us to use the time factor of the interpolation to change the speed of the swing. We also modified the swing to mimic more of a scissor kick to the side with the robot arm fully extended instead of a traditional parabolic swing that starts at a high point and ends at a low point. The resulting swing was much smoother, but we began to run into issues with violating Cartesian velocity limits at the end effector. Since the arm was fully extended, the speed of the end effector would very easily exceed the maximum 1.7 m/s imposed by the robot. As a result, we had to tune the timing factor in the joint space linear interpolation to stay within the velocity limits of the robot. We soon found that the Cartesian space velocity limit generated more issues for the robot as we were unable to generate the power necessary to hit the ball hard enough for a flying trajectory. In order to adapt to this challenge, we decided to elevate the ball and robot in relation to the goal so that the robot's kick can bounce on an intermediate surface before reaching the desired position in the goal. We also oriented our end-effector shoe to the side, as shown in Figure 7, so that we could maximize the surface area of contact with the ball and thus more effectively aim the ball after the kick. Once we were able to reliably kick the ball with the joint space controller, we improved our swing further until we reached our final implementation. These modifications include adding more states in the joint space controller to help us aim accurately and in different directions, as well as giving us added flexibility in interpolation times between states and generate as much power as possible into the kick. These added states also allowed us to slow down the robot after kicking the ball and smoothly return to a 'safe' configuration, i.e. one away from joint limits, so that the controller could be rerun without having to manually move the robot.

4 Conclusion

After completing this project, it is with hindsight that we can appreciate the complexity behind robotic arm control. Throughout this report, we have highlighted the steps taken towards achieving a working controller, and most importantly, the ways in which we adapted our approach with each challenge encountered. This project has thus provided us with a deeper understanding of the methodology behind implementing a successful controller, and it was hugely beneficial to take theoretical concepts, such as different trajectory generation methods, and be able to physically implement them. In addition, the use of sai2 libraries allowed us to experiment with ready-made controllers, thus granting us more time to think about the intended motion rather than the mathematics behind controller implementation.

Working on theoretical assignments in simulation, with concrete results that we expect to see, is a completely different challenge to moving a robotic arm with a particular motion. Therefore, arguably one of the most important aspects of this project has been learning to think creatively to find solutions to problems that we have not encountered before. When a particular approach does not yield the intended results, it can be, and often was, the case that a lot of time was spent trying to fix that approach instead of rethinking the solution to the problem. For example, the initial Cartesian trajectory demonstrated many issues during its physical implementation, even if it worked in simulation. This forced us to really understand the specific task at hand, and to produce a plethora of different possible solutions, both in Cartesian as well as joint space, in order to narrow down our approach towards a unique solution. Moreover, a vital skill we developed was to spend less time in simulation and more time on the actual robot - what worked in simulation often violated some hardware limitations of the robotic arm, and we quickly learned that it is far more important to get a rough trajectory in simulation and adjust it physically instead of perfecting the simulation whereby certain limitations are not considered. By working harmoniously between the robot and the workspace, making and testing changes, and honing the adjustments, this project allowed us a glimpse into how robotic arms can be controlled, and this practical component of the course has been a very rewarding experience.

Included is the link to the video of our final penalty kick: https://drive.google.com/a/stanford.edu/file/d/1-rPdyMOMCgDtXxs9frYuwh7FIeiXjcOH/view?usp=sharing