ECE 470 Final Project Report (Fall 2019)

Anthony Li, Qihang Zhao, Haoran Tang, Quanrui Bai
Netid: tianyul2, qihangz2, haorant3, quanrui2
Team Name: Ping Pong
https://github-dev.cs.illinois.edu/tianyul2/ECE470_FA2019
https://youtu.be/hjoLDNa-Cf8

Ping Pong, or table tennis, is a very popular sport in Asia. Usually, two or four players are involved in the game that hit a ball back and forth across a table divided by a net. Many people love playing Ping Pong, but not everyone can find a mate to play within any time, which usually depends on their friends' schedules. In order to solve this problem, a Ping Pong machine will be designed using a robot arm to enable single-player mode for this sport, so a person can play Ping Pong. Due to time constraints, a basket will be used instead of a racket to collect the ball. The robot arm can successfully reach and collect the Ping Pong ball with an accuracy of 73.3%. The knowledge of inverse kinematics and supervised learning is applied in this project. Further improvement will emphasize hitting the Ping Pong ball back to the player with a racket.

1. Introduction

The goal of this project is to collect Ping Pong ball from a shooting machine with the UR-3 robot arm holding a basket. The entire project is implemented in the V-rep simulator with a built-in Lua script and Python remote API. The problem will need to be simplified and narrowed to a scope that meets the time constraint of this class. There will be no Ping Pong table or net in this simulated environment. Similar to the real Ping Pong rule, the ball will hit the flat surface once, but before the second bounce, the robot arm will need to move to the correct location which catches the Ping Pong ball in the air. The shooting angle will always be within a specific angle range in order to keep the speed slow and the range reachable for the robot arm. The shooting angle is defined by the initial velocity vectors, which contains x, y, and z-direction component. To achieve this goal, the most important parts are inverse kinematics for the robot arm and prediction of the bouncing trajectory for the Ping Pong ball. By applying newton's method to inverse kinematics, given the target point, the correct angles for each joint can be calculated which enable fast and accurate response to catch the ball. The trajectory of the Ping Pong ball with a bouncing path is hard to predict due to friction, restitution, and elasticity. A pattern for the bouncing action is discovered which if the initial velocity is large (mostly x and z component), then the distance that the ball travels will also be large. This corresponding to an approximately linear relation. Therefore, multiple linear regression is used to tackle this problem which predicts the second bounce location of the ball. This technique enables prediction of the catch location of the Ping Pong ball. A more detailed explanation and analysis of these methods will be shown in the upcoming sections. It is worth to mention that in our presentation video, the initial velocity of the ball was originally detected by the camera, but the detection was not accurate enough, so it was removed.

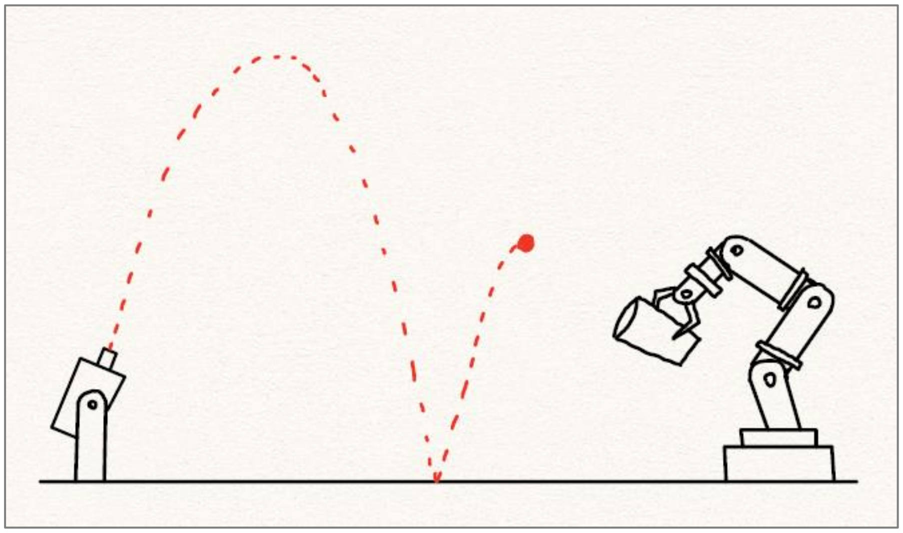

Figure 1. Project Objective

2. Method/Experimental Setup

There are three key steps for completing this project:

I. Predict Ping Pong ball catch position in x component (Linear Regression)
II. Predict Ping Pong ball catch position in y component (Geometry)
III. Move end-effector to the predicted position (Inverse Kinematics)

In order to provide better visualization of the operation process of this project, a block diagram for the project is shown in Figure 2.

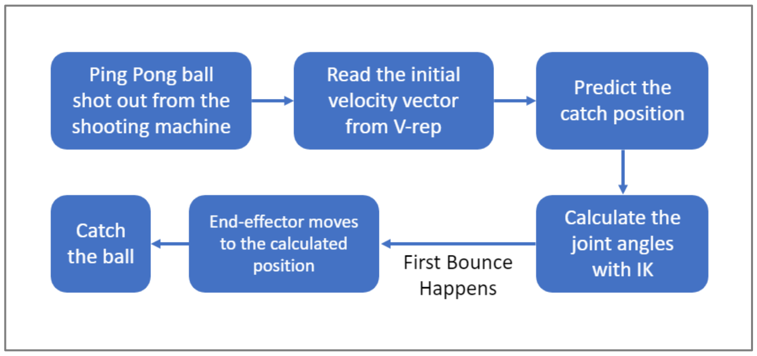
Ping Pong ball shot out from the shooting machine
Read the initial velocity vector from V-rep
Predict the catch position
Calculate the joint angles with IK
First Bounce Happens
End-effector moves to the calculated position
Catch the ball

Figure 2. Block Diagram of the Operation Process

The entire operation process runs in the V-rep simulation environment. The environmental setup and reference world frame are shown in Figure 3. The simulated experiment contains a base floor, a UR-3 robot arm with a basket in the middle of the floor, and a shooting machine at the far edge of the base floor. The basket and the shooing machine are made through 3D computer-aided-design modeling on NX 10.0 software. The UR-3 robot arm and the Ping Pong ball are the V-rep built-in model. The diameter of the ball is 0.12 m. The initial shooting force on the center of mass of the ball is random within a range where

$$F_x = [70,85],\ F_y = [-20,20],\ F_z = [660,700]$$

Each run begins with the start of the simulator in V-rep and running of the python script.

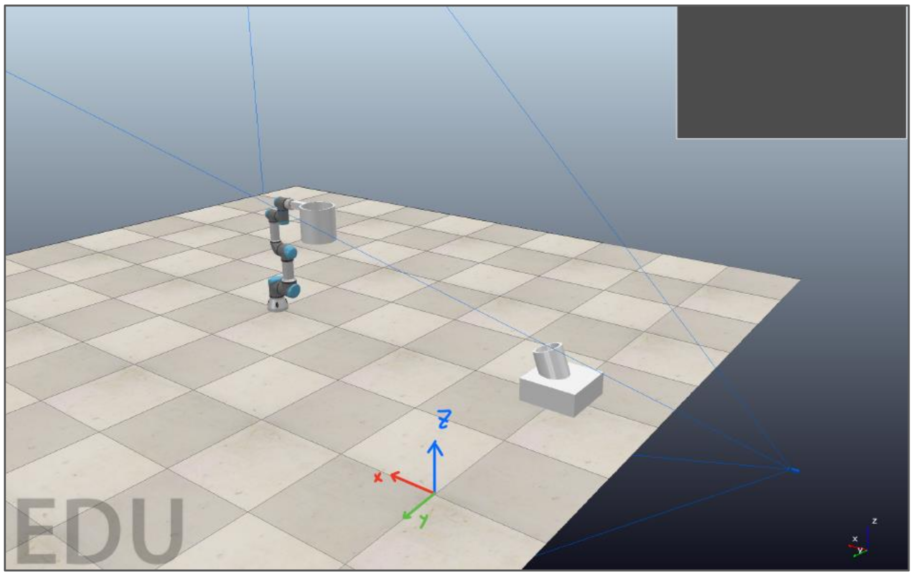

Figure 3. Environment Setup

2.1 Linear Regression

As mentioned in the introduction section, the approximately linear relation is discovered in the initial velocity and distance of the ball travels. Therefore, linear regression is utilized for the prediction of the catching position. More specifically, a pattern for the bouncing action is discovered which if the initial velocity is large, then the travel distance of the ball will also be large. For simplicity, the distance of the ball's second bounce location is described only by the x-location. The y-location will be simply calculated using the initial shooting angle from the initial velocity vector. Since the basket will need to catch the ball in the air before the second bounce, the catching position is at an offset position in the x- and z-direction from the second bounce location as described in Figure 4. The catching z-position will always be a constant. The initial velocity is described by the x and z component. The y-component of the initial velocity won't affect too much of the distance since the range of the horizontal angle that the shooting machine can shoot from is limited by the project constraints.

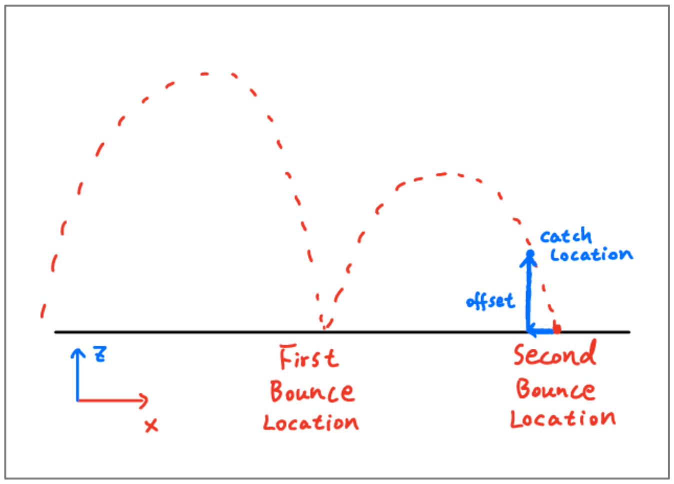

Figure 4. Bouncing Trajectory

The input of the linear regression is the x and z component of the initial velocity and the output of the linear regression is the x-position of the ball second bounce location. In order to do linear regression, it is important to have as many data as possible. For the time constraint, 500 data are collected by 500 times of simulation. The data collection process is described as the following: The UR-3 robot is removed. For each trial of simulation, shooting the ball and record the x and z component of the initial velocity (obtained from V-rep). Then wait to let the ball bounces and record the second bounce x-position of the ball. These corresponding pairs of data are collected through Lua script, and Lua script passes the data to Python which outputs the data to .csv file. In the data collecting process, the y-component of the initial velocity is always zero. With 500 pairs of data, post-processing and multivariable linear regression is performed using MATLAB. The result then generates coefficients that describe the linear relation. The result and analysis will be shown in the Data and Result section.

2.2 Catch Location in y-direction

As mentioned in section 2.1, the actual catching location is not the second bounce location but the second bounce location with an offset in x- and z-direction. The offsets will be determined using trial and error. The y location also needs to be determined. The horizontal shooting angle will be calculated using a trigonometric relation between x and y components of the initial shooting velocity as shown in Figure 5. The distance between the shooting machine and the UR-3 robot arm is 1.975 m in the simulator. By having the distance and the angle, the y location will be able to calculate using tangent relation. This experience is familiar since it is similar to the analytical inverse kinematics lab.

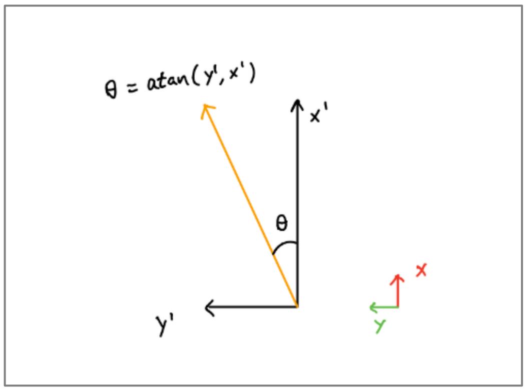

Figure 5. Determining the Horizontal Shooting Angle

2.3 Inverse Kinematics

After obtaining the prediction of the position of the catching location, the UR-3 robot arm with the basket will need to move to the corresponding location without changing the orientation of the end-effector in order to catch the ball. Numerical Inverse kinematics (IK) is applied to solve this problem. Before performing any IK, it is critical to obtain the properties of the UR-3 robot arm including the zero configuration of the end-effector and screw axis in the space frame. After having these properties calculated, IK will be the next step. Given a position coordinate in space, IK will return the corresponding joint angles of the UR-3 robot arm to reach the position. Specifically, the Newton-Raphson method Inverse Kinematics is used in this project. The calculation step for the Newton-Raphson method is described below.

a) Initialize tolerance and maximum iteration count which are conditions for iterations.
b) Initialize guess joint angles θ (most of the time they are zeros)
c) Start iteration
d) Get current T for the end-effector: $T_{current}$
e) Calculate twist $[V] = \log\left(T_{target}{T_{current}}^{-1}\right)$
f) Extract $[V]$ to V
g) Calculate Jacobian from S and current joint angles θ^{i}
h) Calculate pseudoinverse for J (in order to avoid singularity, add a small value of offset to the diagonal of $J^{T}J$ when calculating pseudoinverse)
i) $\theta^{i+1} = \theta^{i} + pseudoinverse(J)V$
j) Repeat step (d) – (i) until condition met.
k) Return θ^{i+1}

After the joint angles converge to a specific value, the Python script will command each joint to move to the specific angle through remote API. The next step is just to wait for the ball to jump into the basket.

3. Data and Result:

The most important analysis in this report is the linear regression. By postprocessing the data in MATLAB, a linear regression model is generated for the prediction of the x position of the second bounce location.

Although 500 data points are collected, only 269 are useful due to inaccurate measurements during the data collection process. Sometimes the V-rep is not sensitive to the collision between the ball and the floor, so V-rep will record the third bounce instead of the second bounce in some occasions. Therefore, suspicious data points with large x distance but small initial velocity are removed. Only 269 data points are left over after removing the outliners. In the beginning, removing data points with such a simple assumption seems inaccurate. However, after careful observation of the data collection process, the inaccuracy is confirmed that it is due to bugs in V-rep detection.

In order to demonstrate the linear relation behind, multiple plots are generated to better visualize the model. Firstly, the relation between each independent variable and the only explanatory variable will be shown. Figure 6 shows the relation between the x component of the initial velocity and the x position of the second bounce location. The blue circles are the data points. The red line is the fit trend line for the data points. There is clearly a linear relationship between the two variables. As the initial velocity increases, the second bounce x-location also increases. The zero at the y-axis is the position of the base of the UR-3 robot arm.

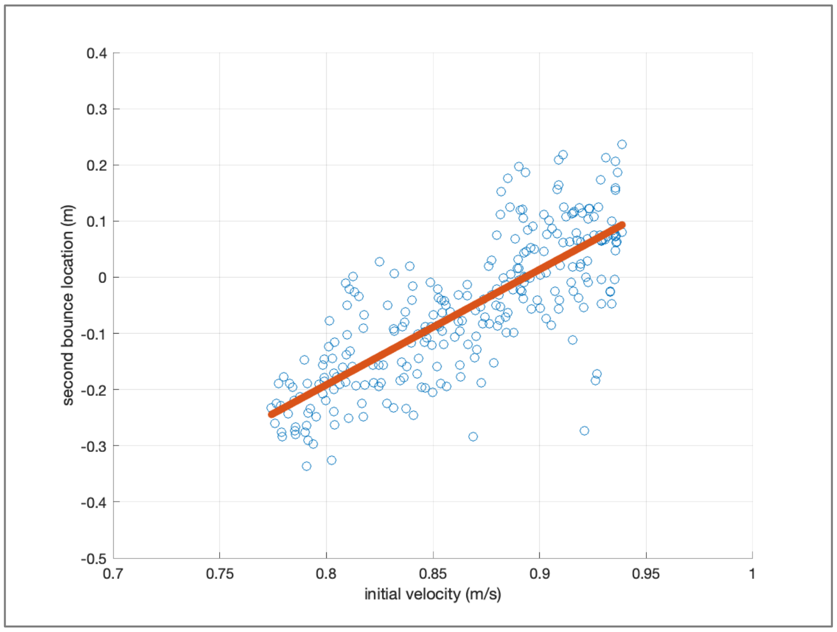

Figure 6. X_{bounce} vs. V_{Xi}

Figure 7 shows the relation between the z component of the initial velocity and the x position of the second bounce location. The linear relation between these two variables is weaker. The data points are more spread out in the figure. By comparing these two figures, one can conclude that the x component of the initial velocity has a larger impact on the second bound x location.

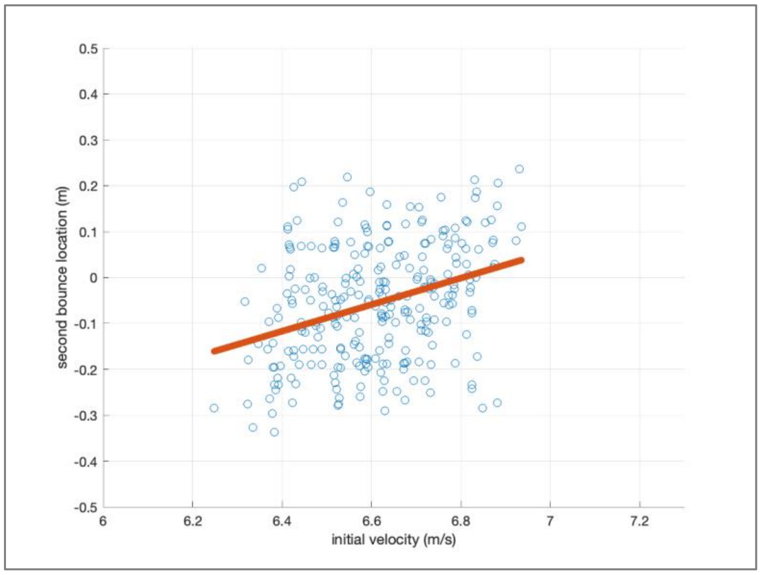

Figure 7. X$_{\text{bounce}}$ vs. V$_{\text{Zi}}$

After comparing the variables separately, a linear regression model will be generated in MATLAB to summarize the relations in a higher dimension. The model is described with the following equation:

$$x_{bounce} = 1.9804V_{xi} + 0.071423V_{zi} - 2.2433 \qquad (1)$$

The R-squared value for this model is 0.617, which indicated a weak linear relation exists. Also, the p-value for the three constant are small: x=1.757e-50, z=0.0408, c=2.81e-21, which shows that the null hypothesis can be rejected. The root-mean-square value is 0.0786. Figure 8 shows the residual plot of this model.

As mentioned before in Figure 4, there will an offset in the second bounce x and z position in order to catch the ball in the air. After multiple trial and error with this model, the best offset is 0.25m for x-direction. The best catching position for z is at 0.5m. After adding the offsets, the prediction model is finished.

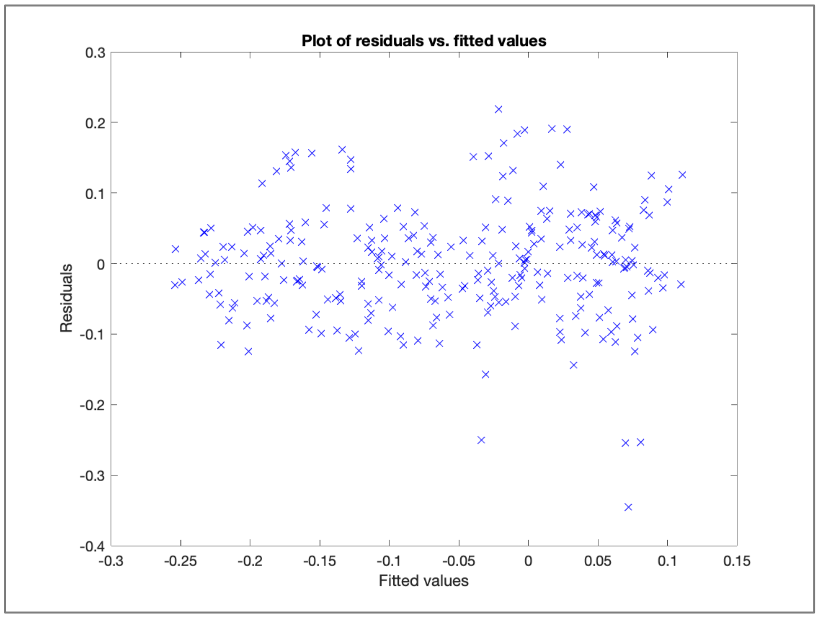

Figure 8. Residual Plot

Compare to the analysis for the linear regression, the analysis for IK is trivial. The success of the IK can be defined as by giving a target location, the end-effector will move to the target location. Since the UR-3 robot arm is always able to converge to the correct location within the constraint of this project using all-zero initial guess angles, IK has a 100% success rate in this project.

The overall success rate for the project is also tested. By operating 30 trials, in 22 of these trails, the UR-3 robot arm successfully catches the ball in the air. The success rate is about 73.3%. Since learning is not a highly accurate method due to the number of data points collected and outliners, the prediction is not always perfect. A lack of data could possibly produce failure cases. Also, failure cases happen often when the ball is shooting out in a low angle in vertical direction. The ball will get to the robot arm faster so there will be less time for the robot arm to react. Also, since the basket is open on the top, the ball will more likely to hit the edge if it does not bounce high enough. This is a design defect that can be fixed by making the open of the basket inclined to the shooting machine direction.

4. Conclusions:

By applying inverse kinematics and supervised learning, the UR-3 robot arm with the basket is able to achieve a success rate of 73.3% in the task of collecting Ping Pong ball under specific constraints. In this project, students had the opportunity to practice robot design in a simulation environment, which is an important step for validating the design before building the system in the real world. Specifically, the V-rep simulator is a popular tool for demonstrating a robot design concept which will be useful for student in future industry work. More importantly, the knowledge of inverse kinematics is applied in this project, and hands-on experience always helps students understand the concept better. Additionally, this project promotes interests in machine learning concepts. Although the class did not cover this knowledge, students recognized it as a useful tool in solving real-world problems and decided to explore the topic. This machine can still be improved in many different ways. If more time is allowed, firstly, a more complex machine learning model, or even neural network, will be used. The model will have a 3-dimensional velocity input and 3-dimensional location output for the ball. Secondly, an actual Ping Pong racket will be used instead of the basket, which is more realistic and similar to the real Ping Pong machine player. Also, the ball will be hit in a specific direction and velocity. Additionally, an actual Ping Pong table with a net will be built in the V-rep simulator.

5. Reference:

N/A