Baxter catching ball

EECS 106B FINAL PROJECT

Yichi Zhang, Jingjun Liu, Jiarong Li

27005775, 3033483513, 3033483511

Abstract

We try to let Baxter follow a random trajectory in a fixed conservative field. In this project, we choose to throw a tennis ball and make Baxter catch it. We come up with two methods: Physical Modeling and Machine Learning Modeling. In the Physical Model, we use classical mechanics along with transformation of coordinates to predict the drop point. In Machine Learning Model, we train Baxter with a bunch of training data so it can react more quickly and more accurately. Finally, we compare these two methods and analysis their advantages and disadvantages

Introduction

In order to catch a flying ball by Baxter, we decompose the task as

- Detect the tennis ball.
- Predict the dropping point of the tennis ball.
- Calculate inverse kinematics.
- Move!

Solution

Model 1: Physical Model

Firstly, according to the image we received from the camera, we extract the ball trajectory by pushing the RGB image into the HSV color space and filter out all non-green colors. Consequently, we can obtain the coordinates of first three data points as well as the time interval between each two points. Secondly, we fit the trajectory of the ball by a parabola model. The velocity of the ball can be calculated by $v_x = \frac{\Delta x_1}{\Delta t_1},\ v_y = \frac{1}{2}\left(\frac{\Delta y_1}{\Delta t_1} + \frac{\Delta y_2}{\Delta t_2}\right)$ and total flying time by $t_{total} = \frac{2 \cdot \|v_x\|}{g}$. Then, we calculate target position by $x_p = x_1,\ y_p = y_1 + v_y * t_{total}$. Finally we get the target position by coordinate transformation. The result is $tar\_pos = g_i \cdot \begin{bmatrix} x_p & y_p & z_p & 1 \end{bmatrix}^T$.

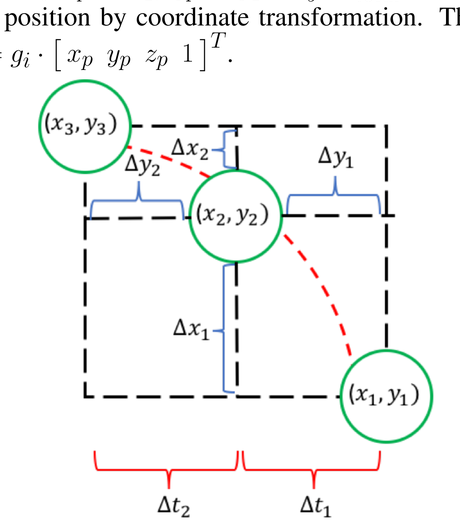

Figure 1: Physical model

To maximize the flying time of the ball, we choose the symmetric position of the first point as the Baxter catching position of the ball and calculate it inverse kinematics to get the Baxter configuration.

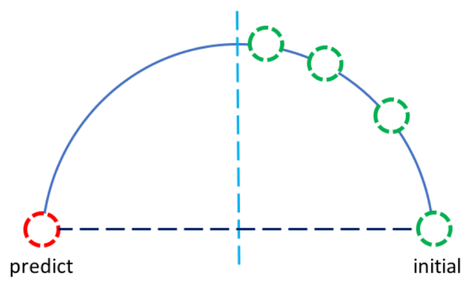

Figure 2: Prediction point

Figure 3: Inverse kinematics

Model 2: Machine Learning Model

A general machine learning process includes collecting data, fitting a model on training data and then predicting results on new data. To fit a model in our case, we decompose the training process into three steps.

- Determine the reachable set of the baxter arm.
- Grid the reachable set and collect data.
- Fit a particular model using the training data.

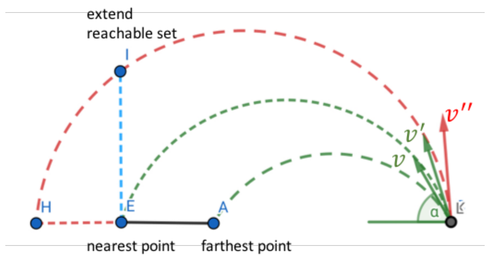

Figure 4: Training Step 1: Determine the reachable set. The angle between v and v' is the original range of initial velocities of the ball that is allowed. Then angle between v' and v'' is the extended range of velocities of the ball that can be thrown by people.

In a two dimensional space, the trajectory of a flying ball is a parabola. Whatever the direction of the initial velocity of the ball is, we can always use a 1 dimensional hyper-plane to intersect those concave parabolas. Therefore, we define the reachable set of the baxter arm to be a 1 dimensional hyper plane, which is denoted as the blue solid line in figure 4.

However, we haven't taken full advantage of the two degrees of freedom of the baxter arm in a 2D space, thus the range of shooting angles of the ball is too limited. For this reason, we add a perpendicular 1 dimensional hyper plane to the original reachable set to extend the allowable shooting angles.

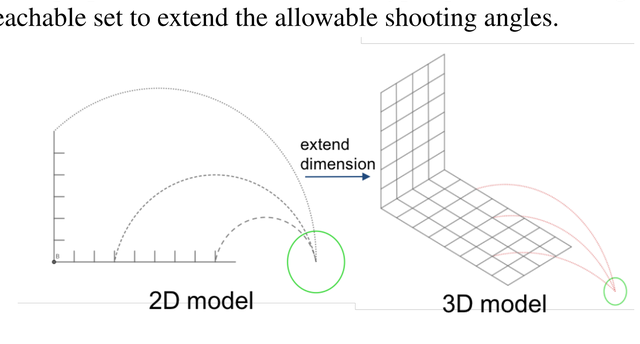

Figure 5: Training Step 2: Grid the reachable set.

The reachable set containing two hyper-planes can be meshed by a certain step size. The step size is determined by the size of the ball. In this case we choose 4 cm. While collecting the data, the baxter arm moves to a node on the grid, the coordinate of the node in the baxter base frame is recorded and appended as a row of the label matrix. The coordinates of first five points of the ball along the trajectory are recorded and appended as a row of the feature matrix. After each node and roughly every corresponding shooting angle are covered, the data collection procedure is done.

This procedure is easy to be generalized to a 3D space. Only by adding one axis and meshing the two hyper-planes in the same way as described above, we can extend the method to a 3D case.

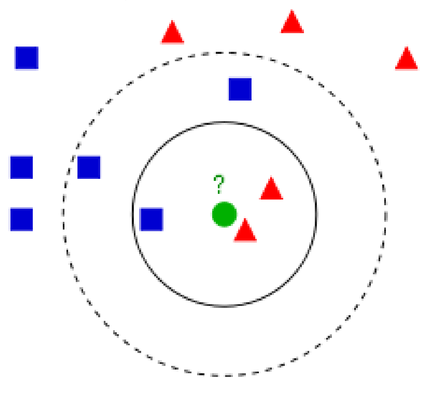

Figure 6: Training step 3: Fitting kNN (k Nearest Neighbors) model. (wikipedia)

Since the training data are collected by meshing the whole space, they evenly disperse the reachable set of the baxter arm. In this case, kNN will be a good choice for us to try. The final training error is approximately 0.02. The validation error is approximately 0.016. It can be inferred that this is still an underfitting model. We can still improve the performance by collecting more data.

Fast controller implementation

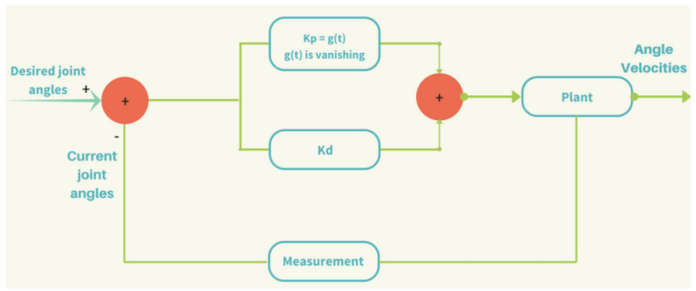

Figure 7: Controller Flow Chart.

Why we use vanishing K_P from large initial value

- Initially large because we need huge speed
- Vanishing because we need it stable

Results & Experimental Evaluation

Physical Model Evaluation

Goodness

- It has good generalization because we do not set limit on the predict point.
- Baxter arm has a larger reachable set
- Do not need to store too many parameters

Badness

- The prediction position is not very accurate
  - we neglect air resistance and assume gravity acceleration equals 9.81 ms^{-2}
  - The image we captured has some distortion and the camera need to be calibrated.
  - We measured the translation between ball and camera by hand so there are certain errors in transformation mapping
- The prediction position is hard to control. Maybe it would predict an unreachable point.
  - Inverse kinematics became unstable because of larger reachable set.
- It has a high requirement on devices as we need to determine position of tennis ball between a very small time span.

Machine Learning Model Evaluation

Goodness

- If given enough data, it can predict the dropping point faster and more accurate. **In our project we collect 284 data points**
- Because all predictions lie in the convex hull of training data. So the robot arm would not move to a unreasonable position, and it is safer and more stable.
- Because we do not need to calibrate camera or measure any mappings. The complexity of system is reduced.

Badness

- **It Costs a lot of time to collect data**
- According to the model, the robot arm would not move out of the configuration we set. So it cannot deal with the trajectory out of reachable set.

Conclusions

Baxter can catch ball by both physical and machine learing model. However, the success rate of physical model is lower and the requirement of devices is higher.

Forthcoming Research

We can extend the 2D catching model to 3D catching model and extend the flying ball model to any random moving object model in a fixed conservative field.

References

[1] Berthold Bäuml, Florian Schmidt, Thomas Wimböck, Oliver Birbach, Alexander Dietrich, Matthias Fuchs, Werner Friedl, Udo Frese, Christoph Borst, Markus Grebenstein, et al. Catching flying balls and preparing coffee: Humanoid rollin'justin performs dynamic and sensitive tasks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3443–3444. IEEE, 2011.

[2] Berthold Bäuml, Thomas Wimböck, and Gerd Hirzinger. Kinematically optimal catching a flying ball with a hand-arm-system. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2592–2599. IEEE, 2010.

[3] Oliver Birbach, Udo Frese, and Berthold Bäuml. Realtime perception for catching a flying ball with a mobile humanoid. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5955–5962. IEEE, 2011.