Baxter Catching a Flying Ball

Yichi Zhang[1], Jingjun Liu[2] and Jiarong Li[2]

***Abstract*— We build a robotic ball-catching system from Rethink Robotics Baxter Robot arm. Unlike previous work which used physical way to solve the problem, we develop both physical model and machine learning model to decide where the robot arm should move to. 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**

I. INTRODUCTION

Let robots perform a sports activity is an excellent realtime benchmark for perception of dynamic scenes, for motion planning, control and for action planning. Also, robot doing human activities can help us judge robot's human-like performance, study human robot interaction and develop a more human-like robot.

Catching a thrown ball is not easy neither for humans nor for robots, which demands highly for a tight interaction of skills in mechanics, control, planning and visual sensing to reach the necessary precision in space and time. In general, a stereo vision system tracks the ball and predicts the balls trajectory, then the point and time, where and in which orientation the robot should intercept the ball on its trajectory, is determined. Then, the robot configuration to reach the catch point is computed and nally a path is generated, which brings the robot from its start conguration to the desired catch conguration.

A. Related Work

In previous work [1], [2], the 4 DOF WAM arm with a gripper to grasp the ball and an active vision system is used. The heuristic catch point selection chooses the closest point of the ball trajectory to the robot base and orients the gripper perpendicular to the trajectory.

In paper [3], a system with a 5 DOF arm on a humanoid robot, which only the arm is moving, with a cooking basket at the end effector for catching the ball and an active vision system is presented. To lead to a human like movement behavior, the inverse kinematics is solved by a neural network.

*This work was supported by Prof. Ruzena Bajcsy and VODAFONE teaching lab @ UC Berkeley.

[1]Yichi Author is a student at the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA zhangych1996@berkeley.edu

[1]Jingjun Author is a student at the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA liujingjun@berkeley.edu

[1]Jiarong Author is a student at the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA jiarongli@berkeley.edu

A 7 DOF DLR-LWR-II arm with a small basket at the end effector and a stationary stereo camera and image processing system built from PAL cameras with 50Hz and standard PC is illustrated in paper [4]. They create an heuristic to determined the catch point by balancing: 1) choose a point far away from the robot to prohibit the robot folding up and running into the joint limits. 2) choose a point near the robot to move fast and reach the catch point in time. Also, they take the orientation of the catching basket opening be perpendicular to the ball trajectory into account when calculating the inverse kinematics. Finally, an interpolator generates the joint paths with a trapezoidal velocity prole (in joint space).

In [5] the focus is on investigating motion primitives for human like path generation. The robot is equipped with a base ball glove to take the end effector orientation into account. The catch point is chosen to be the intersection of the ball trajectory with a horizontal plane at a given height. The main focus of the work [6] is on teleoperated ball catching.

B. Contributions

We use a humanoid robot Baxter with a basket (only the arm is moving), a usb HD camera and a standard PC with Ubuntu. In our work, we not only implement the physical model to predict the ball trajectory but also using machine learning model. Then, we compare the results of these two models and analysis their advantages and disadvantages. We find out that the machine learning model can get a lower error rate and a better result.

II. PRELIMINARIES & SYSTEM ARCHITECTURE

We set a separated camera to detect the flying tennis ball. However, due to the low frequency of our 2D camera (25Hz), we cannot read the radius of tennis ball and then get the distance accurately, so we decided to make the ball fly on a 2D plane. Also, we can easily extend it into 3D space. The camera captures a series of positions and send it to computer. Following different models, a predict point is calculated and sent to a fast controller. Finally the robot arm will move to that position and catch the ball. The paper [7] develop a distributed system to visual ball tracking trajectory by using online kinematically real-time optimization. According to this, [8] presents a real-time perception system for catching flying balls and [9] describes the result of catching flying balls.

III. CIRCLE DETECTION

From the image we received from the camera, we extract the ball trajectory by pushing the RGB image into the HSV

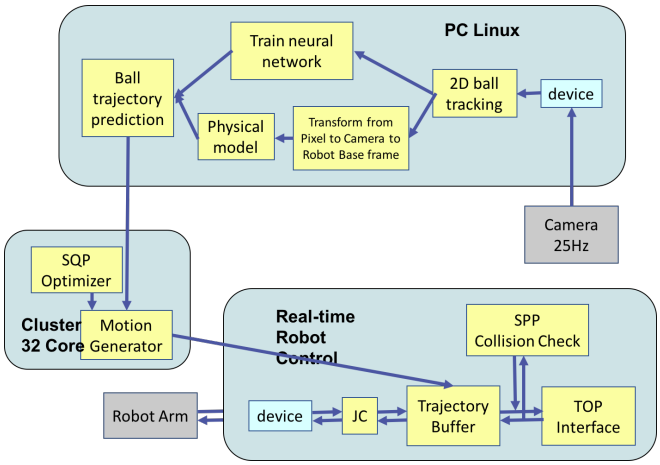

Fig. 1. System architecture

color space and filter out all non-green colors. With the help of OpenCV, we can read the pixel position of ball in camera frame.

In physical model, we use the first 3 points as well as their time intervals for prediction.

In machine learning model, we choose the first 5 points as a set of training data.

IV. PREDICTION MODEL

A. Physical Model

Physical model is a more practical model as we can apply it to any system and any condition.

1) Step I: System calibration: In this model, the transformation between coordinates is important. There are two frames in total: camera frame and robot base frame. Also there are two representations of the ball position: one in unit pixel and the other in unit meter.

Firstly, we change the position of pixel unit into meter unit. To do this, we manually put the ball at several positions with constant distance in pixels and measure the corresponding distances in meter in camera frame by hand. Then, the relationship between pixel representation and meter representation matches.

Secondly, we use AR tags to determine the transformation from camera frame to robot base frame. We attach an AR tag on the camera and use *tf* package. After finding the transformation matrix, we store it for further calculation.

2) Step II: Calculate physical parameters: After we obtain the pixel coordinates of first three data points and the time intervals of tennis ball, firstly we transform it into Cartesian coordinate of camera, then we fit the trajectory of the ball by a parabola model. We define the following pipeline to find the prediction point:

The velocity of the ball can be calculated by

$$v_x = \frac{\Delta x_1}{\Delta t_1} \tag{1}$$

$$v_y = \frac{1}{2}\left(\frac{\Delta y_1}{\Delta t_1} + \frac{\Delta y_2}{\Delta t_2}\right) \tag{2}$$

Total flying time:

$$t_{total} = \frac{2 \cdot \|v_x\|}{g} \tag{3}$$

Target position:

$$x_p = x_1,\ y_p = y_1 + v_y * t_{total} \tag{4}$$

Finally get the target position by coordinate transformation:

$$tar\_pos = g_i \cdot \begin{bmatrix} x_p & y_p & z_p & 1 \end{bmatrix}^T \tag{5}$$

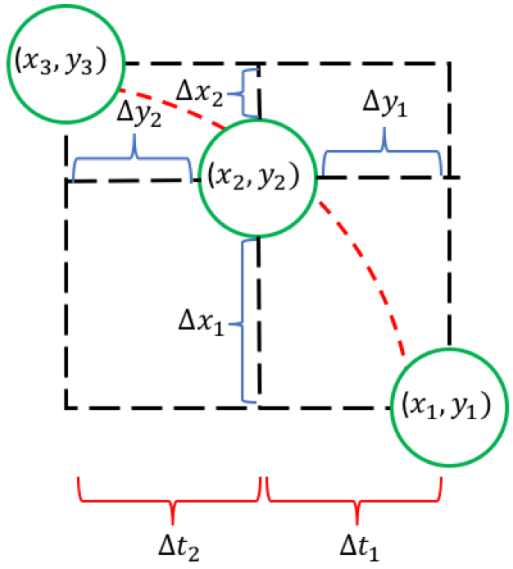

Fig. 2. Physical model

3) Step III: Rules of prediction: Because we need enough time for robot arm to move ti desired configuration, the flying time of ball need to be maximized. So we choose the symmetric point of the first point the camera detected as the catching position.

4) Step IV: inverse kinematics: With the help of ROS package *pykdl*, we can calculate the inverse kinematics of predicted position and send that to controller.

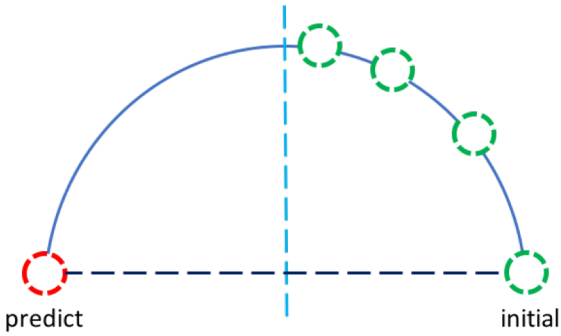

Fig. 3. Prediction point

The physical model can be extended to any movement in a fixed conservative force field. With several sample points we can calculate some physical characteristics of the object, such as velocity, acceleration. Hence, we can deduce its trajectory and a prediction can be made.

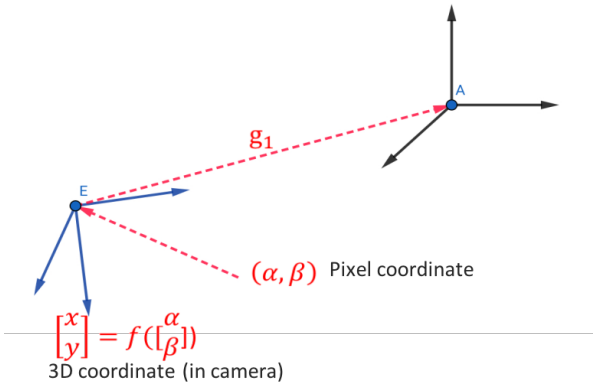

Fig. 4. Inverse kinematics

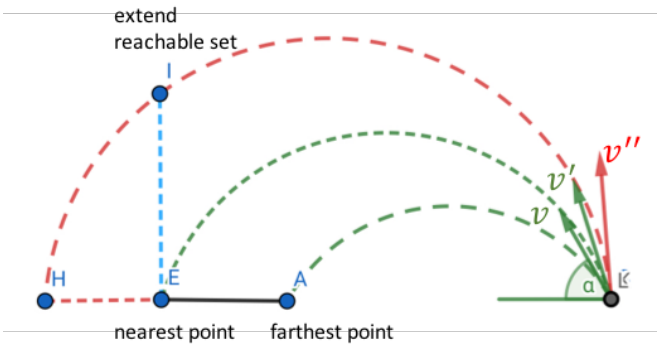

Fig. 5. The Reachable Set of Baxter Arm

B. Machine Learning Model

Theoretically, the physical model can be easily generalized to any possible parabolic trajectories of the ball because its predictions depend on the calculation of velocities along axes. For the same reason, however, it has higher requirements on the hardware. High resolution and high speed camera is required to obtain accurate velocities by computing the average velocity in a short time interval, and to obtain the depth of the ball by examining the radius of the circle fitted in captured images. What's more, the camera also needs to be calibrated for its surface is curved due to manufacturing issues. All these problems take a long time for engineers to solve. Therefore, this report comes up with the idea that using machine learning model to complete the task.

This particular case can be modeled into a supervised learning problem. A general machine learning process includes collecting data, fitting a model on training data and then predicting results on new data. The collected data should contain both feature matrix, which appends every data point as its rows and each of its column is call a feature, and its corresponding labels. 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.

1) Step 1: Determine the Reachable Set of the Baxter Arm: 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 at a certain level. Those intersections are the positions where the Baxter arm can catch the ball. Therefore, the reachable set of the Baxter arm can be defined as a 1 dimensional hyper plane $\mathscr{H}_1$, which is denoted as the blue solid line in the figure.

However, in that case two degrees of freedom of the end effector in a 2D space can not be taken full advantage of, thus the range of shooting angles of the ball is overlimited. To solve this problem, we add another 1 dimensional hyperplane $\mathscr{H}_2$, which is denoted as a blue dashed line that is perpendicular to the original reachable set $\mathscr{H}_1$ to extend the allowable shooting angles. The total reachable set then will be $\mathscr{H} = \mathscr{H}_1 \cup \mathscr{H}_2$.

Intuitively understanding, the Baxter arm always moves in a line to catch the flying ball, if possible. When the shooting angle θ is too large such that the drop point of the ball is estimated to be exceeding the most backward point that the arm can reach, Baxter will lift its arm to try to catch the ball at an earlier position on the parabola trajectory.

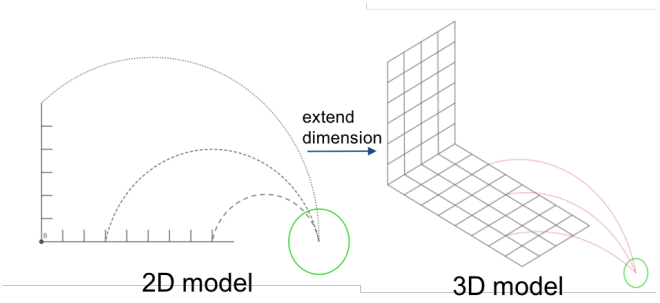

Fig. 6. My model

2) Step 2: Grid the Reachable Set & Collect Data: To guarantee the training data occupies the whole reachable set, we can mesh $\mathscr{H}$ by a certain step size. The step size is determined by the size of the ball. As long as it is smaller than the radius of the ball, the Baxter arm can still catch the ball with that error. In this case we choose 4 cm.

While collecting the data, the end effector of 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 parabola 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. In our case, each feature in the feature matrix is a position in the pixel coordinate system of the camera, each label is a position in the Baxter base frame.

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. Here is some example data points in the feature

matrix and label matrix.

TABLE I
SOME EXAMPLE DATA POINTS IN THE FEATURE MATRIX

576	173	517	155	456	160	396	176	332	193
585	117	543	118	480	123	420	134	367	160
560	278	510	238	456	198	410	170.5	357.5	163

TABLE II
SOME EXAMPLE LABELS IN THE LABEL MATRIX

0.436	-0.654	-0.135
0.395	-0.651	-0.135
0.398	-0.651	0.061

3) Step 3: Fit kNN (k Nearest neighbors) Model: 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 k nearest neighbors algorithm is designed for regression problems. It basically computes the Euclidean distances between the test data point and all data points in the feature matrix, and pick out top k training data points that have the shortest distances to the test data point. The label of the test data point is then the average of the labels of those k training data points. The value of k is determined by exhaustive search. In this case $k = 5$.

The algorithm below shows the procedure of kNN. Denote d as the number of features, q as the dimension of output labels, x_test as the test data point and x_i^T as the i^{th} row of the feature matrix.

Input: Data matrix $X \in \mathbb{R}^{n\times d}$;Label matrix $y \in \mathbb{R}^{n\times q}$; Number of neighbors k; Test data point $x_{test} \in \mathbb{R}^{1\times d}$
Output: A target drop position in Baxter base frame $p \in \mathbb{R}^{1\times q}$
function X, y, k, x_{test}**:**
  Initialize $l \leftarrow$ blank set $\{\}$
  while $step < k$ **do**
    $index \leftarrow \underset{i\in\{1,2,\ldots,n\}-l}{argmin} \|x_{test} - x_i\|_2^2$
    $l \leftarrow l + \{index\}$
  end
  $p \leftarrow \frac{1}{k}\sum_{j\in l} y_j$
  return p
end function

Algorithm 1: KNN

We have collected 284 training data in total. After applying kNN algorithm, the final training error is approximately 0.02, which is 2 centimeters. The validation error is approximately 0.016 (1.6 centimeters). As mentioned before, as long as the error is smaller than the radius of the tennis ball, which is between 6.541 centimeters to 6.858 centimeters, the Baxter arm can catch the ball. Therefore, 1.6 cm validation error reaches the minimum requirement.

It can also be inferred from the values of two errors that this is still an underfitting model. Usually training error is less than or approximately equal to validation error. When the training error is much smaller than validation error, it means that the model suffers an overfitting problem. On the opposite side, when training error is larger, it means the model is underfitting. Its performance can still be improved by collecting more data.

V. FAST CONTROLLER IMPLEMENTATION

In order to reach the prediction point in time, a fast and stable controller is needed for robot arm. After several test, we decide to use a PD controller with a decaying parameter K_p on the proportional element. In our design, K_p is initially large so that the arm can move quickly. It declines to make the arm stay stable at the prediction point.

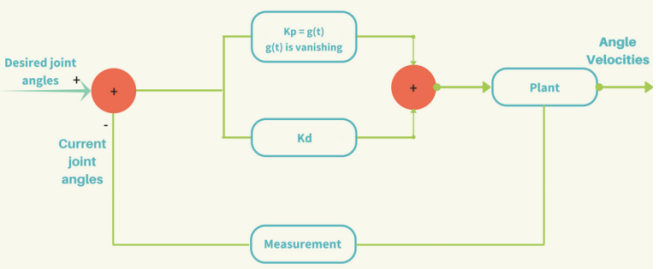

Fig. 7. Controller

VI. EXPERIMENTAL EVALUATION

In testing our models, we find that both model have their advantages and disadvantages.

Physical Model Evaluation

The physical model has some advantages compared with machine learning model.

- 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

However, the behavior of physical model prediction is relatively worse than machine learning model. The success rate is roughly 3 in 20. We believe the reasons are as follows:

- 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.

Fig. 8. Image sequence for physical model (from upper left to lower right)

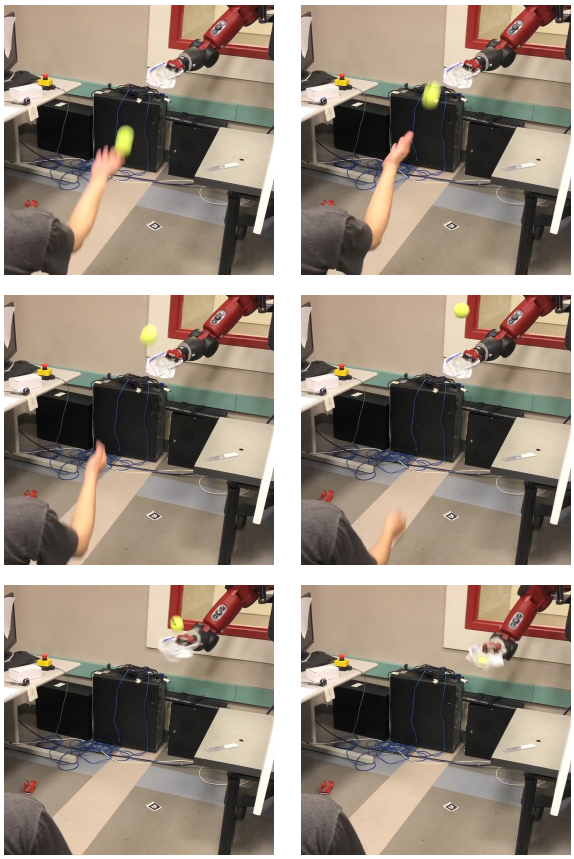

Fig. 9. Image sequence for machine learning model (from upper left to lower right)

- It has a high requirement on devices as we need to determine position of tennis ball between a very small time interval.

Machine Learning Model Evaluation

Compared with physical model, machine learning model shows a better accuracy and stability. The success rate is roughly 7 in 20. Here are its advantages:

- 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.

However, it still has some shortcomings:

- **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.
- It is hard to implement again on other ball-catching system as the type of robot arm and position of camera may change.

VII. CONCLUSIONS

The robot can use either physical way or machine learning way to catch the flying ball thrown by human. However, the success rate and stability of machine learning model is better.

ACKNOWLEDGMENT

This project was supported by the lab of course EECS C106B Robotic Manipulation and Interaction, at UC Berkeley. We would like to show our great appreciation to Prof. Ruzena Bajcsy for all of her support on theories of robotics and sharing her experience and pearls of wisdom with us during the course and the project. What's more, we'd like to thank Chris Correa and Valmik Prabhu for their assistance on some technical problems. Thank the lab manager and engineers at VODAFONE teaching lab for their kind help on hardware issues.

REFERENCES

[1] B. Hove and J.-J. E. Slotine, "Experiments in robotic catching," in *American Control Conference, 1991*. IEEE, 1991, pp. 380–386.

[2] W. Hong and J.-J. E. Slotine, "Experiments in hand-eye coordination using active vision," in *Experimental Robotics IV*. Springer, 1997, pp. 130–139.

[3] K. Nishiwaki, A. Ionno, K. Nagashima, M. Inaba, and H. Inoue, "The humanoid saika that catches a thrown ball," in *Robot and Human Communication, 1997. RO-MAN'97. Proceedings., 6th IEEE International Workshop on*. IEEE, 1997, pp. 94–99.

[4] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schäfer, M. Hahnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2001, pp. 1623–1629.

[5] M. Riley and C. G. Atkeson, "Robot catching: Towards engaging human-humanoid interaction," *Autonomous Robots*, vol. 12, no. 1, pp. 119–128, 2002.

[6] C. Smith and H. I. Christensen, "Using cots to construct a high performance robot arm," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 4056–4063.

[7] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 2592–2599.

[8] O. Birbach, U. Frese, and B. Bäuml, "Realtime perception for catching a flying ball with a mobile humanoid," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5955–5962.

[9] B. Bäuml, F. Schmidt, T. Wimböck, O. Birbach, A. Dietrich, M. Fuchs, W. Friedl, U. Frese, C. Borst, M. 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*. IEEE, 2011, pp. 3443–3444.