Humanoid Robot: Throw Ball at a target

Bhavishya Mittal
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur, India
Email: bhavishy@iitk.ac.in

Pratibha Prajapati
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur, India
Email: pratibap@iitk.ac.in

Advisor: Dr. Amitabha Mukerjee
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur, India
Email: amit@cse.iitk.ac.in

***Abstract*—Our project aims to identify known target through ALDEBARAN NAO and then to throw an object towards it. In our project we first detect the known target using Image Processing techniques, then we compute the position of that object and finally we perform the throwing action. The throwing object ability make a humanoid to manipulate the target or object outside the movable reach of the robot. In the throwing action the robot motion is quick and dynamic. There are many ways to perform this action and this makes the inverse kinematics very non-trivial. Also throwing action requires precision between the timings of different part. This project proposes the linear approximation model and feedback learning based approach to throw objects towards target.**

***Index Terms*—Humanoid Robot; Feedback Learning; Aldebaran NAO; Image Processing; Robotics.**

I. Introduction

The interest in humanoid robotics has increased manifold especially in creating intelligent and autonomous bots. Humanoids are expected to perform day to day task and actions so that they become more interactive with humans.

Performing throwing action is one such task which fulfills a whole lot of needs. It enable one to manipulate places out of ones direct reach. This procedure has many applications in army, humanoid game playing and several other fields.

In our project, we detect the known target, estimate the depth and direction of the target using prior knowledge of the target size and specifications and perform throwing action. Instead of solving non-trivial inverse kinematics involved in performing throwing for Aldebaran NAO we introduce a Learning based Linear Approximation Model approach. To incorporate errors involved with this method we try to capture the movement of thrown object and process its path. Based on the error involved we do feedback learning and perform the throwing action again to reduce errors.

II. Previous Work

There have been several attempts on developing a humanoid or a robot to perform throwing actions at a target..

- In [1] authors discuss the control for throwing manipulation by one joint robot. The paper proposes the control strategy based on the iteration optimization learning to perform the throwing action effectively.
- In [2] authors formulises the idea of task-level learning to make a robot throw a ball at a target. Task-level learning can compensate for the structural modelling error of the robots lower control system.
- In [3] author shows that reinforcement learning can be employed to refine the hitting skill acquired by imitation learning according to a cost function.
- Object recognition and depth estimation is proposed in [4].
- In Aldebaran Nao, there has been a project [5] on kicking a ball to a particular direction, which involved bezier curve interpolation between keyframes.

III. Methodology

We have implemented the ball throwing task using learning based Linear Approximation Model and feedback learning. The Nao is fixed at a position and the target can be placed anywhere subjected to the conditions that it should be kept at same ground level as NAO. The ball and the target are of different colors. The whole methodology can be described in four parts fig:1.

- Target detection
- Throw action
- Video capture and error estimation
- Feedback

A. Target Detection

We set the *HeadYaw* angle (ϕ_1) of the nao head so that it can view the entire target object if kept outside the field of view of nao which is $60.9\,^{\circ}$. Once the image is ready the following processing is done to detect the target and it's coordinates:

- Convert the RGB image fetched from the camera at the chin of the nao to HSV image.
- HSV image produces better results for creating binary image and it also makes the computing faster.
- Get the binary image from HSV in which the area of the image which has HSV values in range of the given color(we used green color in our experiments) of the target is white and the rest of the image is black.
- Apply erosion and dilation, which gives clearer object boundaries and removes noises in the image.

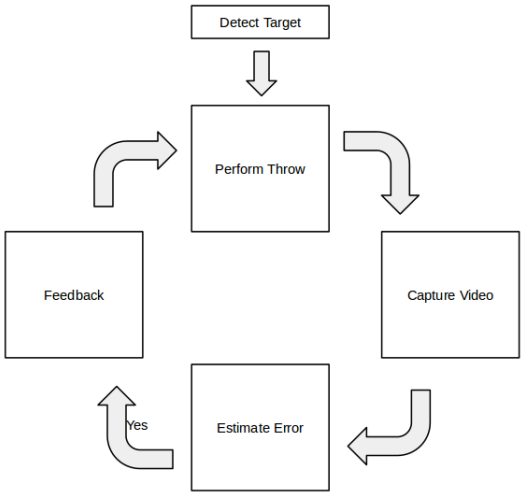

Fig. 1: Flowchart showing the working of our methodology

- From binary dilated image, get the x and y coordinates of the target and thus the angle from camera line of view(ϕ_2).
- Angle of target in naos frame $\theta = \phi_1 + \phi_2$

Fig. 2: RGB Image

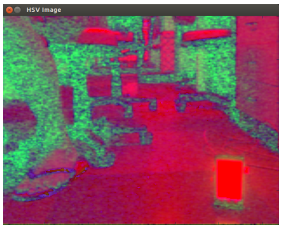

Fig. 3: HSV Image

Fig. 4: Erosion Image

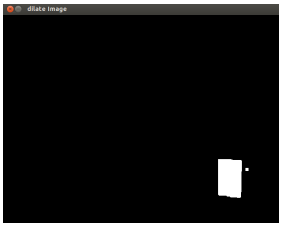

Fig. 5: Dilated Image

HSV Color Model: HSV color model has these three parameters as described in [7].

- **Hue** The hue (H) of a color refers to which pure color it resembles. All tints, tones and shades of red have the same hue.
- **Saturation** The saturation (S) of a color describes how white the color is. A pure red is fully saturated, with a saturation of 1; tints of red have saturations less than 1; and white has a saturation of 0.
- **Value** The value (V) of a color, also called its lightness, describes how dark the color is. A value of 0 is black, with increasing lightness moving away from black.

Erosion and Dilation: Erosion and dilation are the morphological operations as mentioned in [8]

- **Erosion** The erosion of the binary image A by the structuring element B is defined by: $A \ominus B = \{z \in E | B_z \subseteq A\}$ where B_z is the translation of B by the vector z, i.e., $B_z = \{b + z | b \in B\}, \forall z \in E$.
- **Dilation** The dilation of A by the structuring element B is defined by: $A \oplus B = \bigcup_{b \in B} A_b$.

Determining Coordinates of the Target: From the binary dilated image, we get the width(W) and height of the target in the image. From the prior knowledge of original height and width(W_{real}) of the target, we calculate the x and y coordinates in nao torso frame as the following.

$$X_{nao} = \left(X_{img} - \frac{imgWidth}{2}\right) * W_{real}/W$$
$$Y_{nao} = (imgHeight - Y_{img}) * a$$

Here, a is a constant calibrated by experiment, $imgWidth$ and $imgHeight$ are the frame size and X_{img} and Y_{img} are the coordinates of the center of the detected target in the image. Angle of throw in camera frame

$$tan(\phi_2) = X_{nao}/Y_{nao}$$

Therefore, Angle of throw in nao frame

$$= HeadYaw + \text{Angle in camera frame}$$
$$\Rightarrow \theta = \phi_1 + \phi_2$$

B. Throw action

For performing throwing action after the angle of throw has been determined, there are two approaches namely Cartesian control approach and Joint control approach.

Cartesian Control: In Cartesian control, one can directly control the effectors of Nao, that is the 3D points of the Nao, but to throw a ball, we not only need to control the effectors of Nao, but also the trajectory on which Nao arm moves, as this directs the trajectory of the ball after throwing. Therefore, we used Joint control to move robot arm.

Joint Control: These API can control the position of Nao joints directly. And hence can be used to give desired trajectory to robot arm.

The **joints used** for the throwing actions are: RShoulderPitch, RShoulderRoll, RElbowYaw. Table:I gives the motion range of these joints.

TABLE I: Joints and their ranges

Joint	Motion	Range(in degrees)
RShoulderPitch	Right shoulder joint front and back	-119.5 to 119.5
RShoulderRoll	Right shoulder joint right and left	-76 to 18
RElbowYaw	Right shoulder joint twist	-119.5 to 119.5

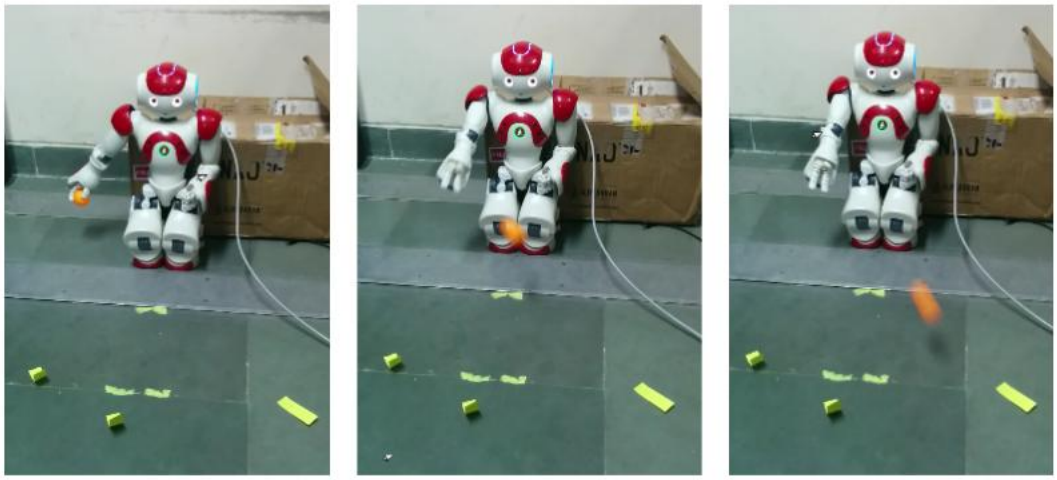

Fig. 6: Nao performing throwing action while training

Initial Attempts: First, we tried to solve the inverse kinematics for the naos right arm. But this resulted in a much complex problem. As there are many ways to throw a ball in a specific direction using three joints used, the problem clearly didn't have a unique solution. Also this is not some trivial end-effector problem as nao need to move its arm in a defined direction or projection to make the throw successful. This can be understood as suppose somehow we have the coordinates where the nao should leave the ball but the trajectory of the ball after leaving naos hand has temporal dependency. Also, the time of release matters a lot in out case because delay in releasing the ball will lead to just dropping of the ball. Therefore, we moved to a relatively new approach **Learning Based Approach**

Learning Based Approach: In this model we used 8 parametes.

- 3 initial joint angles
- 3 final joint angles
- 1 for speed of motion
- 1 for time to release the ball

Now we assign random values to these 8 parameters repetitively and observe where the nao throws the ball. During this learning phase following **observation** were made:

- Many combination resulted in nearly the same angle of throw.
- Many combinations failed to throw the ball. Main reasons for it was leaving the ball too soon or late then desired. Leaving sooner results in slipping of ball from hand and leaving late results in mere dropping of ball.
- Even for the same parameters, outcomes may very due to hardware volatility.

From this nao learns parameters for performing throw in specific angles. Fig-6 shows the throw action being performed. Yellow markers in these image refers to the learned angles i.e. parameters for these angle of throw are known apriori.

Now, for any test angle (any random angle, may not be available in training set), we used **Linear Approximation Model**.

Linear Approximation Model: This model assumes that parameters for angles between any two nearest learned angles can be approximated as a linear combination of their parameters. Each of the 8 parameters are calculated this way and perform the throwing action. The following procedure is followed:

- Using binary search and find the containing learned angles.
- Assuming linear model, estimate parameters for the given theta from the parameters of the containing learned angles.
- Perform throw actions following these parameters.

C. Video Capture and Error Estimation

As nao throws the ball, it simultaneously captures the video of the trajectory of the ball for feedback learning. From this video, we track the motion of the the ball. This gives the angle in which the ball actually moves when thrown with the given parameters. To track the ball in the video, first we tried the following strategy,

Initial Attempt: As in our video only ball is dynamic and the environment is static we tried to get the direction of the motion of the ball by using the optical flow of ball pixels in the video. But this didn't work very well, as the quality of the video obtained from nao camera was not of good quality and the image of the ball was blurred in most of the video frames. Another problem with this approach is that the ball bounces several times on the ground, and hence the final direction may be different from the initial angle at which ball was thrown. So we implemented an alternate approach to track the ball.

Fig. 7: Detecting where ball bounced. It is visible here that the ball must have bounced first time between frame 2 and 3.

Alternate Approach: In this approach fig:7, we detected the coordinates of ball center in each video frame. Whenever, in three consecutive images, the ball center moves from a higher point to lower point and then again moves to a higher point, this means the ball has bounced on the ground at this particular point. We compute the coordinates of the ball in nao's torso frame at this point and compute the angle of the ball(ϕ) from nao. In case the ball doesn't bounce, the last image of the ball is considered for computing final angle from nao. When we try to detect the ball in the video frame, a blurred image of ball is captured. So to deal with this a substantial HSV range has to be given to detect the ball. But this also captures much noise from the environment. To handle this noise, we used the curve nature of the ball as the determining feature and also have thresholds for minimum and maximum size for filtering noise.

The error e in the throw action is:

$$\begin{aligned} e &= \text{target angle} - \text{angle of ball motion} \\ &= \theta - \phi \end{aligned}$$

D. Feedback Learning

If error in throw angle is very small(less than $5°$), we insert the current used parameters for θ in the learning database. If the error in throw angle is more than $10°$, we perform the throw action again with corrected value of θ, which will be $\theta + e$. We keep repeating this action until the error is reduced to a very small threshold.

IV. Results

The method was tested with random positions and there is an average error of 5-10 degrees. The throw movement was more effective when nao was in standing position rather than couch. One reason for this is when standing, nao is more free to move its arm in all direction. When in couch position and the throw is to be made in cross direction, nao tends to hit its knees with the arm.

Target was kept within a maximum distance of 1 meter from nao. Depending upon the timing of releasing the ball different trajectories are possible keeping other parameters constant.

V. Conclusions

Nao performs better when in standing posture then in crouch. Linear Approximation Model has proved to be an effective model. Based on just 5 initial learned angles varying from ($45°$ to $-30°$), nao is able to hit it's target kept between this range effectively. Instead of solving the quite complex and non-trivial inverse-kinematics involved here, this model provides a simple alternative.

Fig. 8: Sequence of images showing the working of the ball throwing humanoid robot

Linear Approximation Model and feedback learning can together be used to increase the database and add valid entries of parameters for the test set. This will eventually lead to rich database of learned points and the efficiency for further throws will improve.

VI. Future Work

Future work includes :

- Able to throw at targets kept at a height.
- Detecting target of various geometry and shape.
- Able to throw directly at the target.

References

[1] Miyashita, Hideyuki, Tasuku Yamawaki, and Masahito Yashima. "Control for throwing manipulation by one joint robot." *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on IEEE*, 2009.

[2] Aboaf, Eric W., C. G. Atkeson, and D. J. Reinkensmeyer. "Task- level robot learning." *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on IEEE*, 1988.

[3] Kober, Jens, Katharina Muelling, and Jan Peters. "Learning throwing and catching skills." *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on IEEE,* 2012.

[4] Masters Thesis of Toms Gonzlez Snchez,Dep. of Computer Science and Mathematics, Universitat Rovira I Virgili, September 2009, 64-82

[5] Lovish, Rahul IIT Kanpur, "Detection(Partial)/Kicking the Ball with Aldebran Nao", 2013

[6] http://en.wikipedia.org/wiki/HSL_and_HSV

[7] http://infohost.nmt.edu/tcc/help/pubs/colortheory/web/hsv.html

[8] http://en.wikipedia.org/wiki/Mathematical_morphology