Thrower and Goalie Robot Arms

Machine Learning and Robotics Lab,
University of Stuttgart

Mohammed Muddasser
Raphael Brösamle

Objective

The project topic is to build a scenario consisting of a Thrower and a Goalie robot arms. The thrower moves to the desired position and throws the ball while the task of the goalie is to intercept and block the incoming ball from making a goal.

The scenario is simple and synonymous to a goalkeeper in hockey, football or imitation of a block by a defender in basketball. The scenario is executed in an in-house simulation environment called Robotic AI (RAI) [https://github.com/MarcToussaint/rai] which uses the NVIDIA based PhysX simulation engine. RAI implements inverse kinematics using k-order Markov Path Optimization (KOMO)). The mesh model of a standard panda arm is is used for the thrower and goalie. The thrower is equipped with a gripper and the goalie is equipped with a flat paddle to its last joint. A suitable region behind the goalie is demarcated (goal) which the robot should prevent the ball from entering.

The ball is thrown towards the goal by the thrower from random positions as set by the user. The trajectories are planned/directed towards random points on the goal plane and within the goal dimensions.The thrower and goalie have a field of view of 180 degrees in front of them. A ball is removed from simulation once it has stopped moving for a certain amount of duration. The balls are shot periodically with certain interval of time from different positions continuously, until the simulation end time is reached. The perception pipeline is skipped for the task implementation such that the current ball position is directly queried through the API available in the current RAI simulation environment for trajectory planning and ball intercept position calculations.

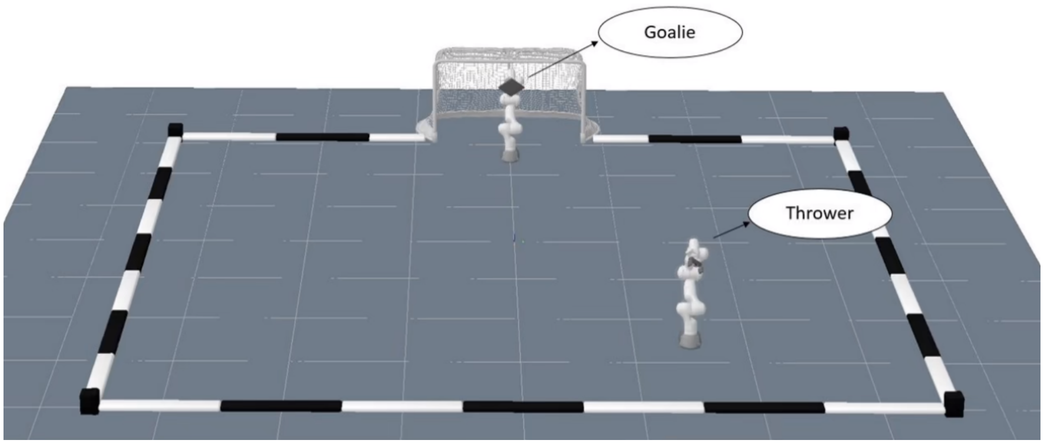


Figure 1: Environment setting

This real time implementation of the setup has applications in crowd pulling, fun and entertainment activities. Also, it can be used for spot training in football, hockey or basketball practise and training. With a ring or cup shaped attachment it can also be extended for the game of throw and catch.

Work Packages and File Structure

The code can be structured into following work packages.

Work Package 1: Scene creation

The scenario as shown in Figure 1. Robot arm modifications to include the paddle and gripper accordingly. Add a thrower, goalie, goal and set the field area in the simulation environment.

Folder:
Project\meshes - contains goal mesh.
Project\scenarios - contains the graph data structure and .g files for the thrower and goalie.
Files::
environment.py - main file for the project.
state.py - state machine.

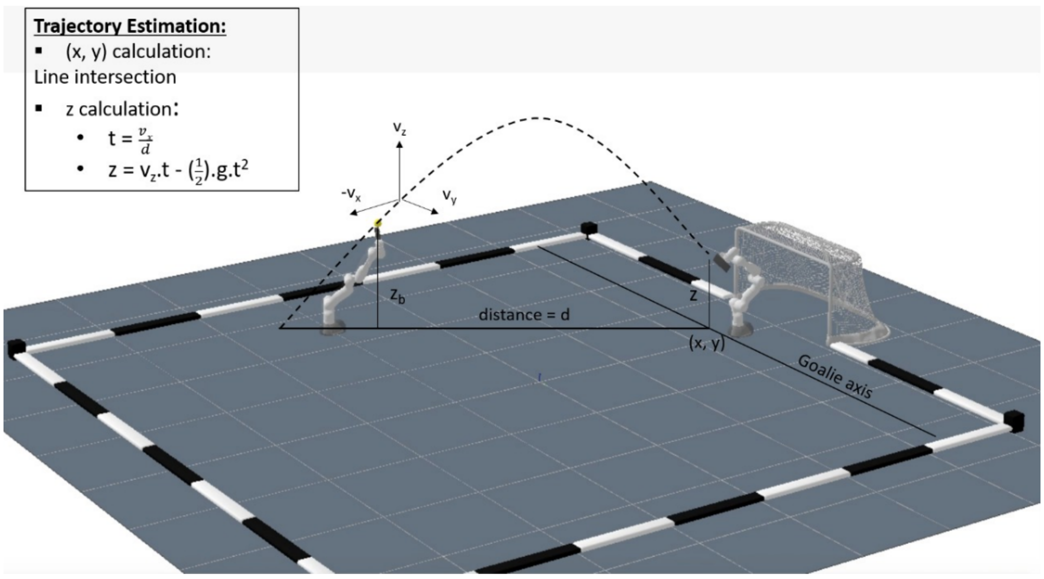


Figure 2: Trajectory planning

Work Package 2: Thrower Trajectory Planning

A robot arm is used to throw balls at the target (as a ball cannon). Here, as a first step the robot arm is moved towards the ball, the ball is grasped, and lift is initiated. Inverse Kinematics is implemented using the standard RAI library functions. The force required is obtained by moving the robot arm to attain certain velocity and acceleration at the end effector and to open the gripper at the right moment to give it a desired trajectory. This setting is calibrated by trial and error and for desired minimal accuracy, as the ball does not need to hit the target at a specific point. Additional calibration to handle the ball slips from the robot gripper during the arm motion. The trajectory direction is estimated by calculating the line intersection between the thrower and goalie 2D positions. Refer Figure 2.
Files: In the folder 'Project'
trajectory.py - trajectory planning
thrower.py - thrower functionality

Work Package 3: Ball Management

There are two modules for ball management. At the thrower side it is used to keep track of the ball initiation and termination in the simulation environment. On the Goalie side it is used to calculate the trajectory and estimate

the point of interception between the ball and the robot arm.
Files: In the folder 'Project'
ballmanagement.py - Ball management

Work Package 4: Goalie and Ball position estimation

Implemented robot manipulation using Inverse kinematics to implement a block. The position of the ball is estimated at every timestep of the simulation. The following two algorithms have been used and analysed (Refer Figure 2):
Trajectory based estimation:
- Line intersection between the goal direction and the axis of the goal to get x and y co-ordinates of the intersection point.
- 1D projectile motion to estimate the height of intersection(z).
Approximate estimation:
- Line intersection between the goal direction and the axis of the goal to get x and y co-ordinates of the intersection point.
- Track the height of the ball at every time step to get the height(z).

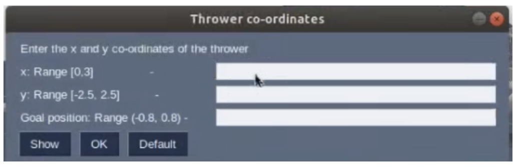
Files: In the folder 'Project'
goalie.py - Goalie related functionality

Work Package 5: GUI

The GUI for setting the position of the thrower and position of the goal as shown in 3 and in Figure 4.
Files: In the folder 'Project'
gui.py - GUI related functionality

Work Package 5: Example Jupyter Notebooks

A set of Jupyter Notebooks are added for the users to play around with the project. The Project.ipynb can be used as a starting point. The notebooks are self explanatory for the scenarios they demo.



Thrower co-ordinates

Enter the x and y co-ordinates of the thrower

x: Range [0,3] -

y: Range [-2.5, 2.5] -

Goal position: Range (-0.8, 0.8) -

Figure 3: GUI for position settings

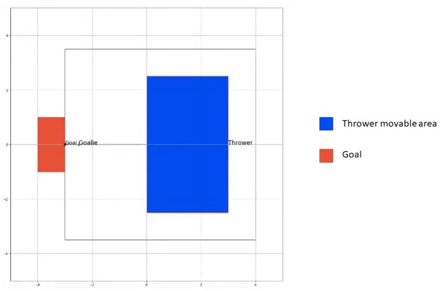


Figure 4: Field position setting of the thrower and goalie

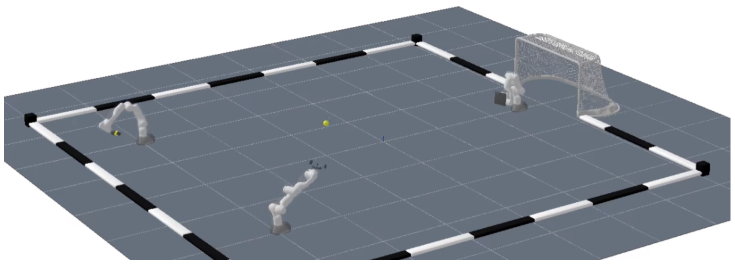


Figure 5: Trajectory planning

Setup:

This setup assumes that Ubuntu 18.04 is used. Please first follow the instruction in the *Setup for Robotics Practical in Simulation* section of the robotics-course (https://github.com/MarcToussaint/robotics-coursesetup-for-robotics-practical-in-simulation) repository.

Install the **gui** and **tkinter** dependencies:
* ‘pip3 install gui‘
* ‘sudo apt-get install python3-tk‘

- Please move all files and folders in the ’scenarios’ folder into RAI folder ’../robotics_course/scenarios’ folder.
- Please also move the ’meshes’ folder into RAI folder ’../robotics_course/scenarios’ folder.
- Please move all files in this repository into the ‘../robotics_course/SOME_FOLDER/SOME_OTHER_FOLDER‘ folder.
- You can now run the jupyter notebook in this folder.
- If you want more than one thrower in the simulation, you need to make some

changes to the code as shown in the Figure 5. Please open the '../robotics_course/scenarios/setup.g
file and uncomment the lines which are marked with MARK. Also in the '../robotic_course/SOME_FOLDER/SOME_OTHER_FOLDER/environment.py' file you need to change the 'thrower_identifiers' variable, which is also marked with MARK, to '[1, 2]'. If you want to have just one thrower again, just revert all these steps.