KUKA Cricket Star

Final Project Proposal
EN.503.707 Robot System Programming
Spring 2020

Jiawen Hu, Kejia Ren, Qihao Liu, Shimin Pan

March 29th, 2020

1 Description

Our project aims to make the KUKA robot arm imitate a cricket player in Gazebo simulation environment. More particularly, we throw a ball within an selected range of initial velocities and use a multi-view ball tracking system (consisting of 3-4 cameras) to track and predict the 3D polynomial ball trajectory in real time. And then we control the KUKA LWR robot arm, which is holding our self-designed cricket bat, to hit back the ball.

minimum deliverable
–able to throw a ball with random initial velocity (within a certain range).
–able to control the KUKA robot arm with a self-designed bat model attached onto it.
–build the multi-view tracking system and able to inspect and process the images from it.

expected deliverable
–able to detect the ball and track its 3D trajectory in real time.
–able to predict the 3D ball trajectory by fitting polynomials.
–directly move the KUKA robot arm at proper time to the target position to touch the ball.

maximum deliverable
–able to visualize the ball trajectory prediction in Rviz.
–more precisely predict the polynomial trajectories with Kalman filter.
–take more desirable planning on robot arm to make more professional hit (e.g. forcefully hit back the ball with good angle).

2 Software

Existing ROS Packages/Libraries & Testing:
1. computer vision related packages: opencv, cv_bridge, image_geometry, pcl_ros.
–testing: able to successfully interface ROS with OpenCV; built a package for ROS-based 2D and 3D object recognition to test.

2. KUKA related packages: lwr_controllers, lwr_description, lwr_hw.
–testing: loaded KUKA LWR model into Gazebo with corresponding plugins; sent motion command to the controller and virtual robot arm in Gazebo moved accordingly.

3. linear algebra package: Eigen (used to solve polynomial prediction).

4. camera sensor: libgazebo_ros_camera.so;
–testing: added gazebo camera sensor plugin to an arbitrary link with image showed correctly in rviz; with specific topic set inside plugin tag, outside nodes could subscribe image file from this topic and do further process.

5. Parrot AR Drone related packages (If at last time remains, we might want to additionally implement a drone ball picker to recycle the ball for making the project more interesting): hector_*, ardrone_autonomy, tum_simulator.
–testing: successfully installed the tum_simulator package with ROS Kinetic version; able to hover,land, move and rotate the drone in Gazebo by publishing to related topics.

New Packages to Implement:
1. ball_throwing: a simple ros package for spawning and throwing balls with initial velocities in Gazebo by interfacing via terminal.

2. multi-view system plugin: a Gazebo plugin for tracking 3D positions of the ball with multi-camera in real time.

3. trajectory_prediction: a package subscribes to information sent from the multi-view system for predicting trajectory and visualizing in Rviz.

4. kuka_cricket_control: for automatically controlling the ball hitting process.

3 Timeline

Week 1
-development: implement the interface for throwing ball in Gazebo; design the cricket bat model in URDF and attach it onto the robot arm; start to build the multi-view system (manually tune the pose for each camera so that the whole experiment area can be seen by each camera).
–testing: test whether the robot arm can still be controlled to move as expected after adding the bat model onto it; inspect images from the multi-view system to make sure it works properly.

Week 2
–development: implement ball detection and tracking on 2D image with computer vision method; implement the 3D ball tracking with triangulation; implement polynomial fitting method for ball trajectory prediction
–testing: visualize in Rviz for testing performance: a. the real ball trajectory by directly read the pose of the ball from Gazebo; b. the tracked ball trajectory by multi-view system; c. the predicted ball trajectory by fitting the curve.

Week 3
–development: implement the KUKA control package/plugin for automatic motion.
–testing: by directly using the real ball trajectory, test whether the KUKA robot arm can automatically touch the ball by the cricket bat.

Week 4
–development: improve the trajectory prediction task by adding smoothing method (e.g. Kalman filter); improve the KUKA planning method to take more professional automatic hit (e.g. forceful hit with good angle, or even double-hit).
–testing: visualize in Rviz to see whether the trajectory prediction becomes more accurate; test with the new KUKA control/planning strategy.

Week 5
–development: integrate all the parts together and fix bugs to wrap up the final project; write the corresponding documentation; write the final project report and poster.
–testing: take as many experiments of the whole project to find underlying problems and fix them.

References

[1] https://github.com/CentroEPiaggio/kuka-lwr.
[2] http://wiki.ros.org/vision_opencv.
[3] https://github.com/angelsantamaria/tum_simulator.