KUKA Cricket Star

Project Weekly Report 04
EN.503.707 Robot System Programming
Spring 2020

Jiawen Hu, Kejia Ren, Qihao Liu, Shimin Pan

April 26th, 2020

1 Ball Throwing

progress:

1. Added gazebo plugin in ball and pad urdf/sdf files. Set surface parameters in ode, bounce, contact tags to make ball bouncy.
2. Added function (input: initial position, target position, region size, duration time) to control the initial ball velocity randomized within a proper range (to guarantee to ball is reachable by the robot arm).

future work:

1. Might need to further tune related parameters to buy more time for the robot to react.

2 Multi-View System

progress: The implementation of the multi-view system is all done.

1. Fixed the latency by turning off the camera images shown in Gazebo and downsizing the image to a smaller resolution. Currently, the image frequency can reach about 40 Hz, which is usable. (The image rendering rate is bottlenecked by Gazebo, particularly, the high load for Gazebo to simulate image rendering restricts FPS, thus we could do nothing to completely solve this problem unless we use a real camera.)
2. Added Kalman filter to smooth the 3D tracking process. In Kalman filter, we use a constant acceleration (gravity) motion model to estimate a 6-dim state (ball position and velocity in 3D), and the 3-dim observation is the tracked 3D ball position directly obtained from triangulation. Since the environment in Gazebo is not complicated (very little noise), the tracking result is already very accurate, thus Kalman filter does not give noticeable improvement. But it is believed that the Kalman filter will help if we run our project in the much more complicated real world.
3. Added another topic (geometry_msgs/PointStamped) for visualizing the smoothing performance in Rviz.

3 Trajectory Prediction

progress:

1. Implemented the traj_pred package for running a node to fit the 2nd order polynomial curve (three curves w.r.t. time, which respectively correspond to X, Y, Z directions) with least square formulation, and then predict the ball position and velocity (by computing the derivatives of the polynomials) after 0.5/1 second, and this information will be sent to the robot for preparing hitting the ball.

2. Added related topic for visualizing the trajectory prediction performance in Rviz.

future work:

1. It still needs to tune the parameters to get better performance. The whole process, from throwing the ball to tracking and estimating the ball trajectory to making a decision for robot to interact, lasts no more than 2-3 seconds. Thus we need to later consider how to allocate the time for each task.

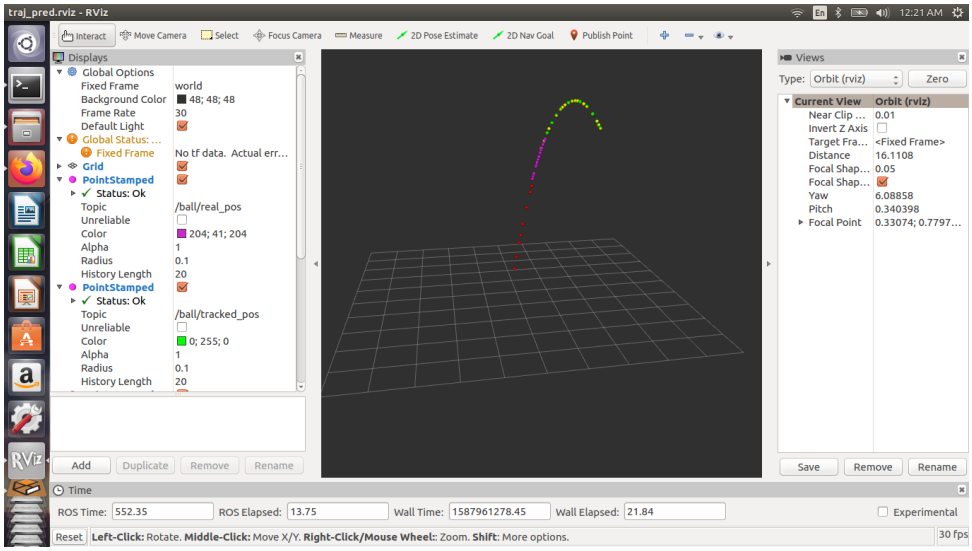

Figure 1: Rviz visualization of the results from multi-view tracking and trajectory prediction. The purple point is the real ball position directly obtained from Gazebo, the green point is the tracked ball position by multi-view system, the yellow point is the estimated ball position after Kalman filter smoothing, and the red point is the predicted ball position in 0.5 second from our estimated polynomial trajectory.

4 KUKA LWR Control

progress:

1. Implemented a process control package which controls the robot arm in the cricket-hitting task (subscribe prediction messages and send command to the control pipeline). This package also acts as an top-layered interface between the user and the robot arm, communicating through keyboard. User can send 'ready gesture' command to the robot arm and receive its running state info.

2. Debugged the motion planning package for lwr's fast and smooth motion.

future work:

1. Integrate parameter fine tuning for robot arm controlling pipeline.

5 Summary

We need to combine all tasks together and run the whole project pipeline in the coming last week. We plan to run as many as experiments to tune parameters and fix bugs to get expected performance.