KUKA Cricket Star

Project Weekly Report 03
EN.503.707 Robot System Programming
Spring 2020

Jiawen Hu, Kejia Ren, Qihao Liu, Shimin Pan

April 19th, 2020

1 Multi-View System

progress: Completed implementation of the Multi_View_Sys class and tested on it.

1. Used message_filters package to synchronize the 2D tracking topics (type of the topics is geometry_msgs/PointStamped) from multiple cameras.

2. Completed the 3D ball tracking with triangulation method provided by OpenCV.

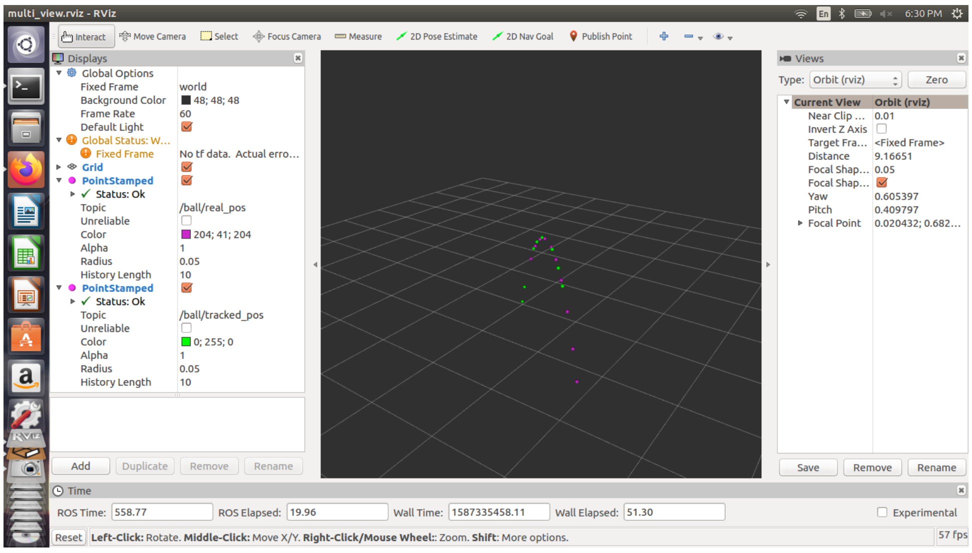
   More details involed:

   1). The camera projection matrices are manually computed once at the beginning using the camera intrinsics and poses where the poses are obtained directly from Gazebo.

   2). We also use KDL for helping calculate the camera rotation matrices since OpenCV does not provide good methods to deal with quaternion.

   3). If the ball is currently invisible to some cameras, we will drop the images from them and just use the remaining images for estimation; if the ball is currently visible to less than one camera, we will not do estimation for this stamp.

3. Tested the 3D tracking performance by visualizing in Rviz to compare the real position and tracked position of the ball. The tracking accuracy is good, however, the latency of receiving message is high, which is due to the heavy workload of running Gazebo. (e.g. We set the update rate of the camera plugin to 144Hz to simulation high-speed camera but the real rate is bottlenecked to merely 10Hz on my computer.)



```
Global Options
    Fixed Frame          world
    Background Color     48; 48; 48
    Frame Rate           60
    Default Light        ✓
Global Status: W...
    Fixed Frame          No tf data.  Actual erro...
Grid                     ✓
PointStamped             ✓
    Status: Ok
    Topic                /ball/real_pos
    Unreliable           ☐
    Color                204; 41; 204
    Alpha                1
    Radius               0.05
    History Length       10
PointStamped             ✓
    Status: Ok
    Topic                /ball/tracked_pos
    Unreliable           ☐
    Color                0; 255; 0
    Alpha                1
    Radius               0.05
    History Length       10
```

```
Type: Orbit (rviz)    Zero
Current View          Orbit (rviz)
    Near Clip ...     0.01
    Invert Z Axis     ☐
    Target Fra...     <Fixed Frame>
    Distance          9.16651
    Focal Shap...     0.05
    Focal Shap...     ✓
    Yaw               0.605397
    Pitch             0.409797
    Focal Point       0.020432; 0.682...
```

```
ROS Time: 558.77    ROS Elapsed: 19.96    Wall Time: 1587335458.11    Wall Elapsed: 51.30    ☐ Experimental
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.    57 fps
```

Figure 1: Rviz visualization of 3D tracking. The purple point is the real position from Gazebo, the green point is the tracked position by our program.

summary: The 3D tracking part is almost completed and works well, except for a little high latency, which might need to be solved by running Gazebo on a better PC.

future work:

1. In terms of the latency, will test to run the program on a better PC to see whether the camera frame rate could achieve our expected frequency.
2. If the above test fails, need to find an alternative way to reduce the latency, or let the ball fly for a longer while to win more time for the robot to react.

2 Trajectory Prediction

progress:

1. Build a package to predict the position of the ball within 100 time steps. The input is the 3D position of the ball from the Multi-view system. And currently the output is the 3D position and the 3D velocity (i.e. a 6×1 vector) of the ball within 100 time steps
2. Add Kalman Filter to the package to estimate the current state. The state is a 7×1 vector, which include the 3D position, 3d velocity and the acceleration along z-axis.

future work:

1. It still needs to test more hyper-parameters for better estimation.
2. The form of the output still needs to modified.

3 Robot Arm Control

progress:

1. Implemented LWR trajectory planning module "cricket_motion_planning" package using Reflexxes (Type II) Library. The input of this module is the target 6D pose of the end-effector and the output of this module is intermediate pose (command), which will be sent to inverse kinematics module.

future work:

1. Implement a cricket hitting process control module, whose task is to control the whole process of the robot arm in this task and coordinate with Trajectory prediction module. This involves deciding the hitting point & hitting time, judging whether the ball has been in the work range of the robot arm, sending command at specific point of time, etc.