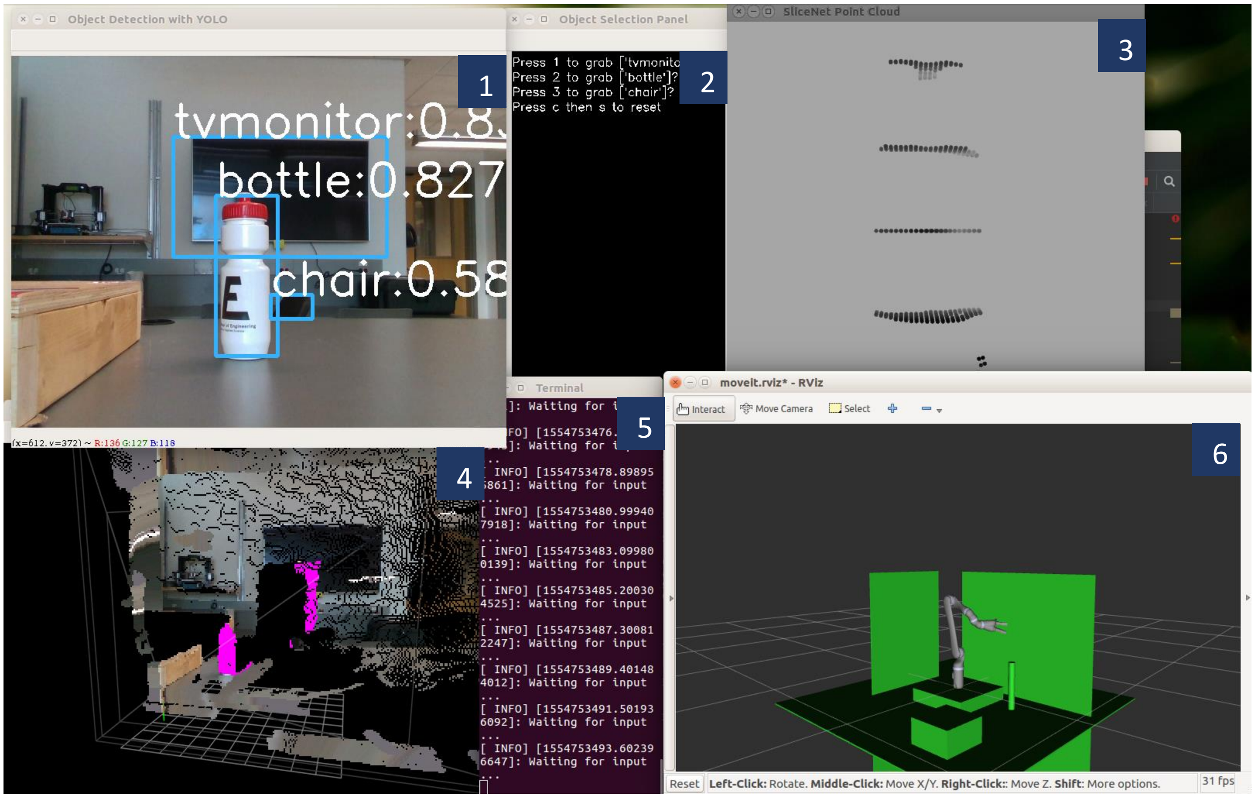
JACO ARM GUI Instructions

The following instructions have been developed for autonomous grasping of simple cylindrical objects that fit within the gripper width. They have been developed off of a set-up shown in Figure 1.

WARNING: THE ROBOT MAY PERFORM **UNEXPECTED MOTIONS** IN ACHIEVING THE OBJECTIVE GRASPING FUNCTION. **THE ROBOT IS UNAWARE OF ADDITIONAL COLLISION OBJECTS. KEEP CLEAR BY 1 M.**

Dependencies

Processor	Operating System	ROS	Python
AMD64 Processor	Ubuntu 16.04	Kinetic	Python2.7
	Gnome Terminal	Gazebo	Python3.5
		MoveIt!	NumPy
		RViz	SciPy
		GraspIt!	Open-CV 4.0.1.


Object Detection with YOLO
1
tvmonitor:0.8
bottle:0.827
chair:0.58
(x=612, y=372) ~ R:136 G:127 B:118
Object Selection Panel
2
SliceNet Point Cloud
3
4
Terminal
5
moveit.rviz* - RViz
Interact
Move Camera
Select
6
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Move Z. Shift: More options.
31 fps

```
Press 1 to grab ['tvmonito
Press 2 to grab ['bottle']?
Press 3 to grab ['chair']?
Press c then s to reset
```

```
]: Waiting for i
FO] [1554753476.
]: Waiting for i
...
 INFO] [1554753478.89895
5861]: Waiting for input
...
[ INFO] [1554753480.99940
7918]: Waiting for input
...
[ INFO] [1554753483.09980
0139]: Waiting for input
...
[ INFO] [1554753485.20030
4525]: Waiting for input
...
[ INFO] [1554753487.30081
2247]: Waiting for input
...
[ INFO] [1554753489.40148
4012]: Waiting for input
...
[ INFO] [1554753491.50193
6092]: Waiting for input
...
[ INFO] [1554753493.60239
6647]: Waiting for input
...
```

1	**Object Detection with YOLO**	Objects in bounding boxes according to confidence
2	**Object Selection Panel**	Objects numbered for key-press selection
3	**SliceNet Point Cloud**	Sliced point cloud of object used for its location
4	**RS-Viewer Window**	Viewing window showing stereographic point cloud
5	**Terminal**	Terminal showing execution stages of JACO ARM
6	**RViz**	Terminal showing virtual simulation of JACO

Running the Program

1. Turn on JACO from the control console
2. Open gnome terminal
3. Wait 5-6 seconds
4. Execute **bash c.sh** in the gnome terminal
5. Minimize the first gnome terminal
6. Open PyCharm, Spyder or another Python IDE for opening the project JACOAuto
7. From JACOAuto, run rs_viewer.py
8. Minimize the Python IDE
9. **Left-Click** on the **RS-Viewer Window** 4 shown in Figure 2.
10. **Left-Click** on the **Object Detection with YOLO Window** 1 shown in Figure 2.
11. Press any key out of either 1,2,3,4 ... as selected grasp object. Duplicate labels are listed in order of descending confidence.
12. **Left-Click** on the **Object Detection with YOLO Window** 1.
13. **Press "o"** to send **gripper to object**
14. **Press "g"** to **grasp** and return **home**
15. **Left-Click** on the **Object Detection with YOLO Window** 1.
16. **Press "s" for selection OR**
17. **Press "c" for new scene**

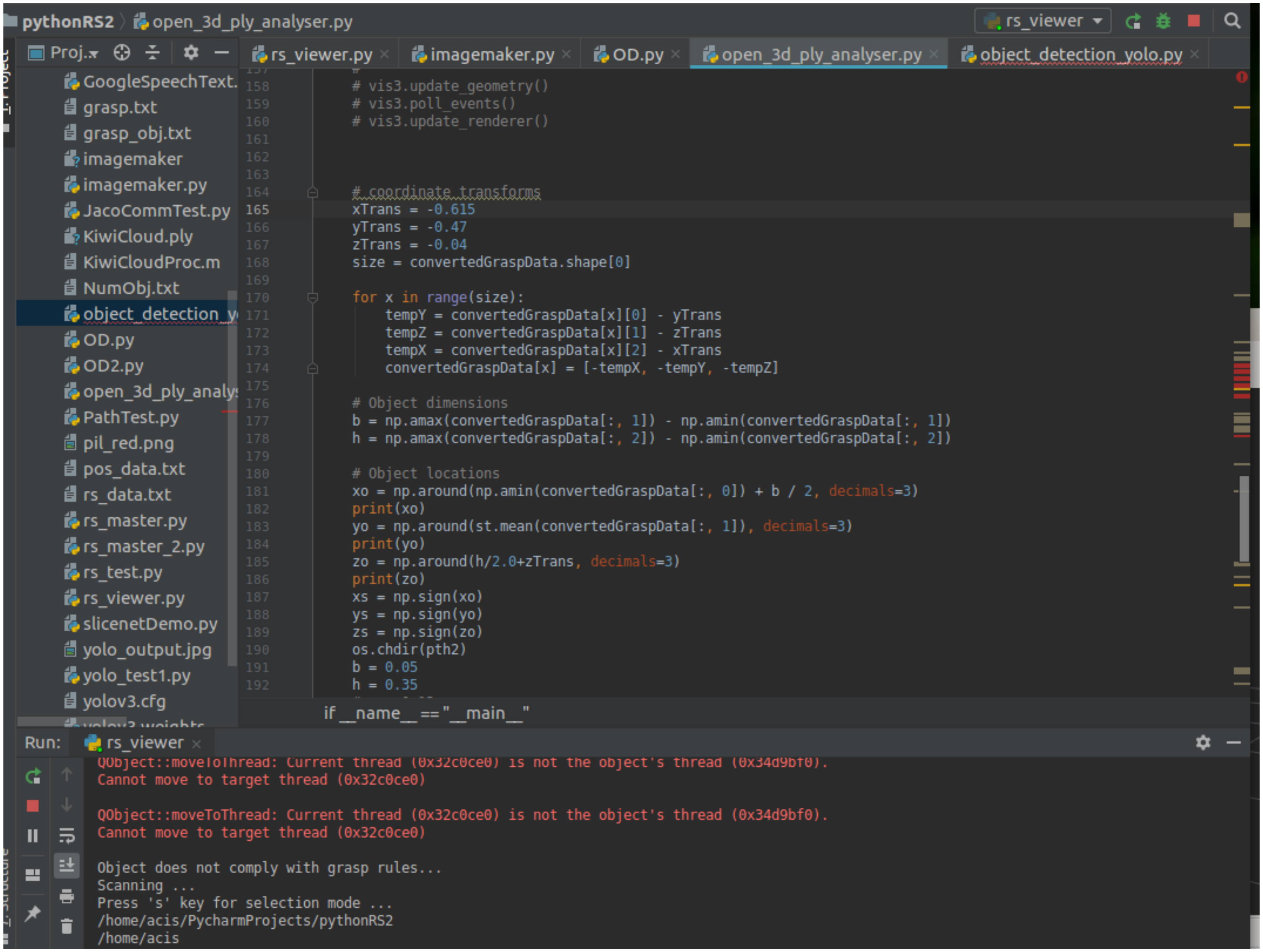
Closing Procedure

18. **Left-Click RS-Viewer Window** 4,
19. **Press "1" or "e"** to exit the program.

Calibration Procedure

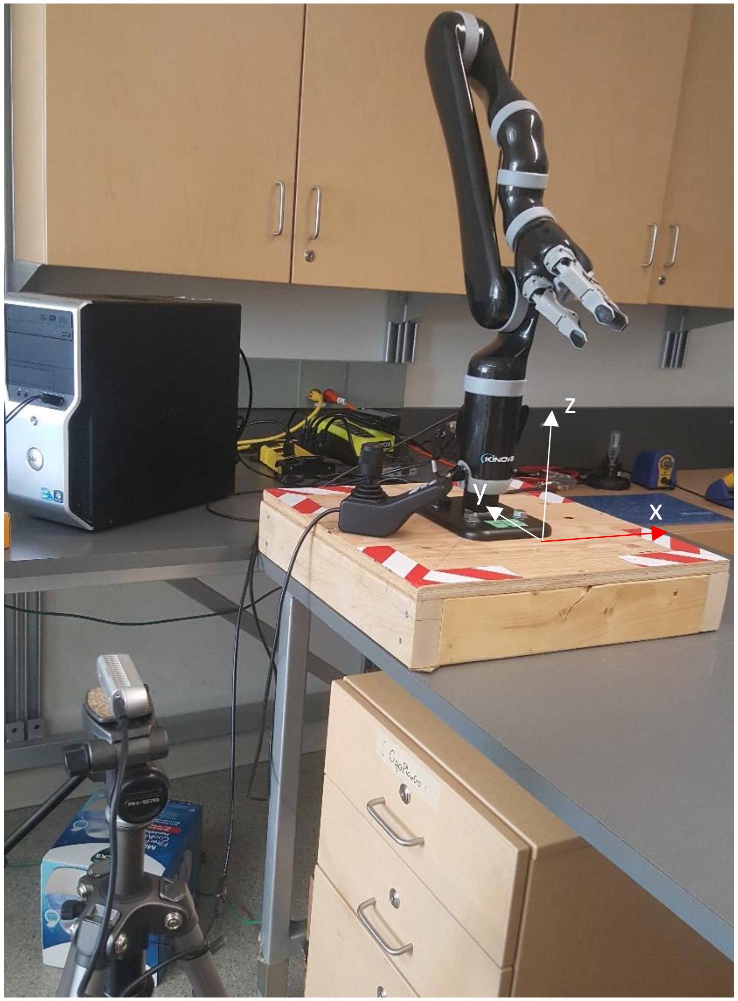
WARNING: LOCATION OF THE CAMERA IS SIGNIFICANT TO THE FUNCTION OF THIS METHOD. DO NOT RELOCATE THE CAMERA POSITION AS AUTOCALIBRATION IS NOT IN PLACE. USE THE FOLLOWING PROCEDURE TO RECALIBRATE EACH TIME THE LOCATION OF THE CAMERA WITH RESPECT TO THE ROBOT IS CHANGED.

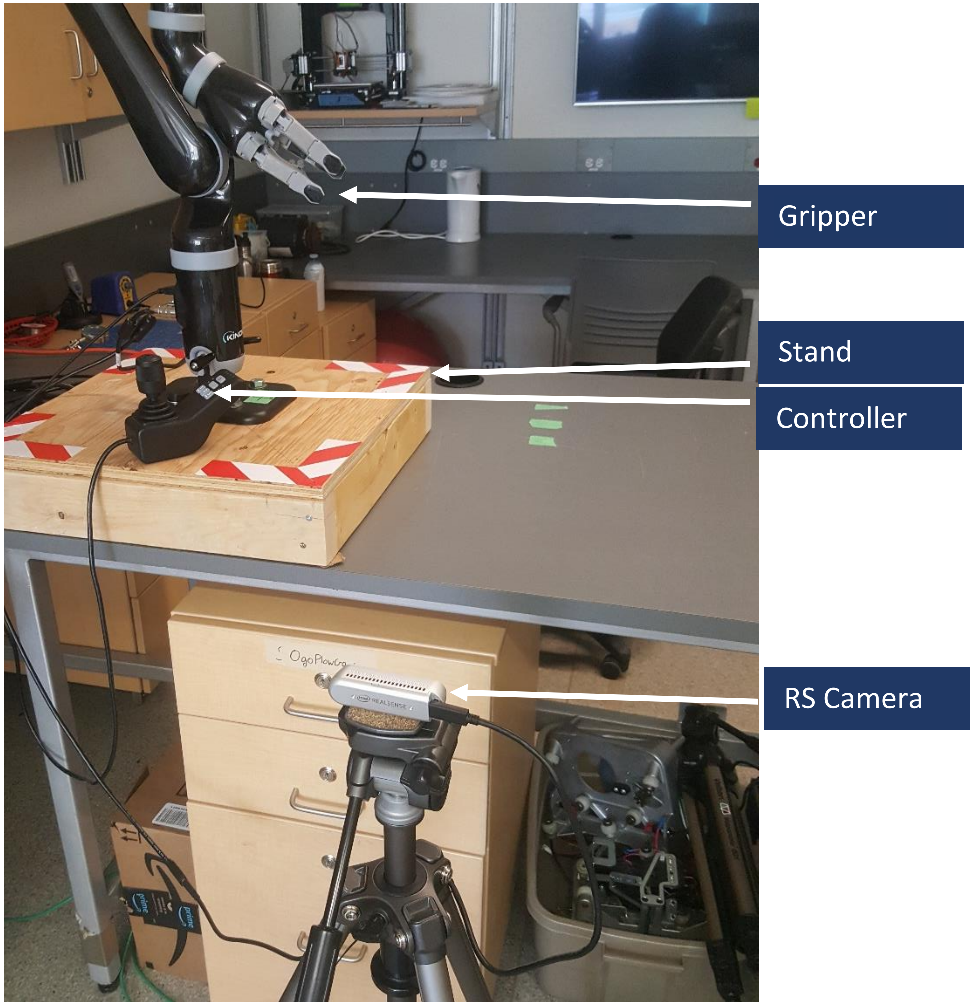
1. Place the camera next to the box.
2. Measure to center of camera to origin of robot in x, y and z coordinates.
3. Enter these distances in the script object_detection_ply.py.
4. Import measured surrounding objects by editing the build_workscene() function in pick_place.cpp for additional or changed geometries, these geometries may be imported as point clouds with additional file reads (additional files required).
5. Run the program, or allow it to continue running to check the result

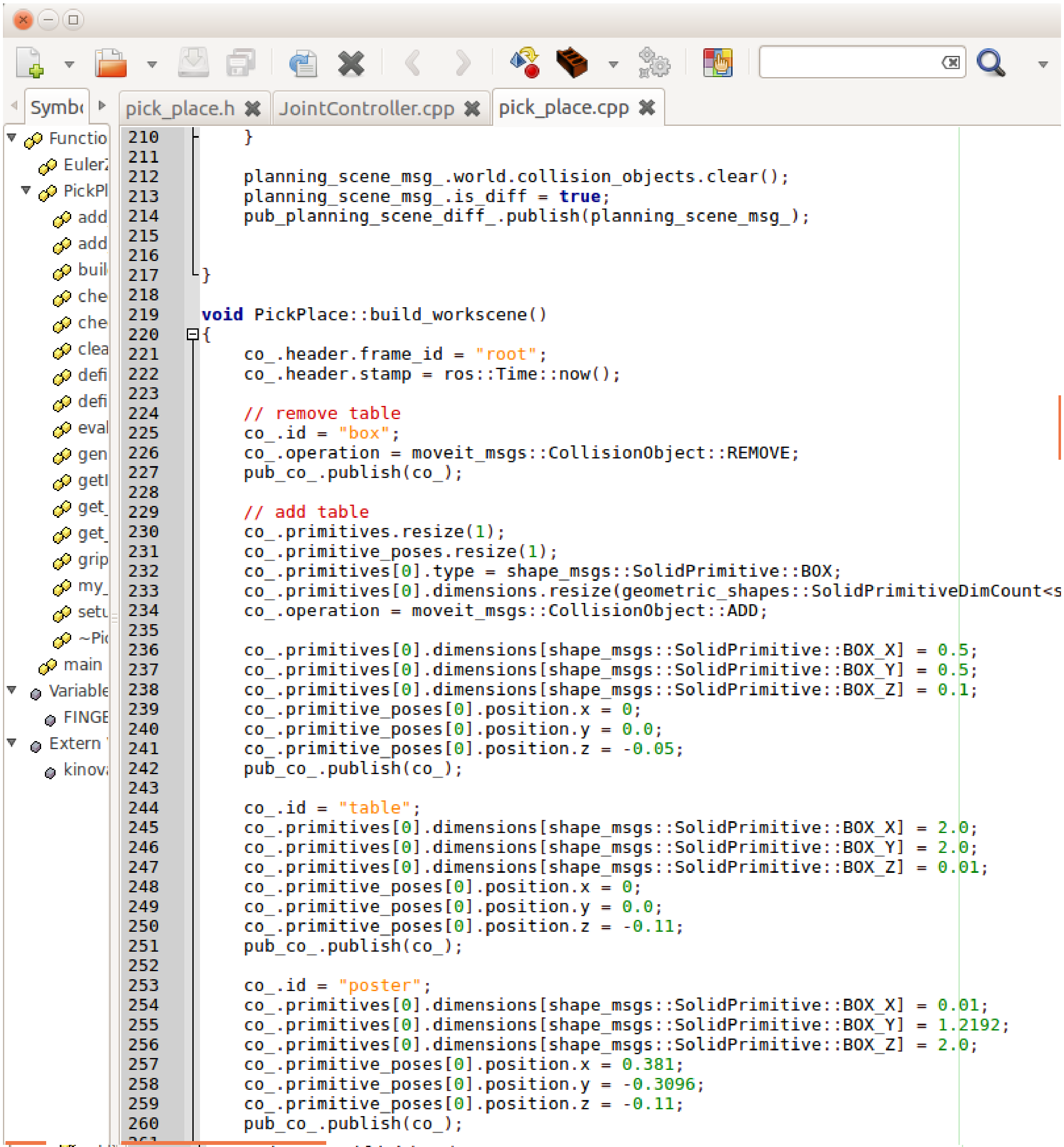


```python
    # vis3.update_geometry()
    # vis3.poll_events()
    # vis3.update_renderer()



    # coordinate transforms
    xTrans = -0.615
    yTrans = -0.47
    zTrans = -0.04
    size = convertedGraspData.shape[0]

    for x in range(size):
        tempY = convertedGraspData[x][0] - yTrans
        tempZ = convertedGraspData[x][1] - zTrans
        tempX = convertedGraspData[x][2] - xTrans
        convertedGraspData[x] = [-tempX, -tempY, -tempZ]

    # Object dimensions
    b = np.amax(convertedGraspData[:, 1]) - np.amin(convertedGraspData[:, 1])
    h = np.amax(convertedGraspData[:, 2]) - np.amin(convertedGraspData[:, 2])

    # Object locations
    xo = np.around(np.amin(convertedGraspData[:, 0]) + b / 2, decimals=3)
    print(xo)
    yo = np.around(st.mean(convertedGraspData[:, 1]), decimals=3)
    print(yo)
    zo = np.around(h/2.0+zTrans, decimals=3)
    print(zo)
    xs = np.sign(xo)
    ys = np.sign(yo)
    zs = np.sign(zo)
    os.chdir(pth2)
    b = 0.05
    h = 0.35
```

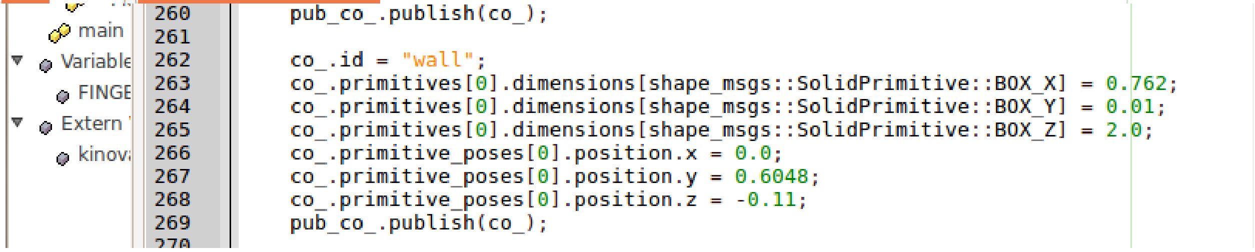
```
QObject::moveToThread: Current thread (0x32c0ce0) is not the object's thread (0x34d9bf0).
Cannot move to target thread (0x32c0ce0)

QObject::moveToThread: Current thread (0x32c0ce0) is not the object's thread (0x34d9bf0).
Cannot move to target thread (0x32c0ce0)

Object does not comply with grasp rules...
Scanning ...
Press 's' key for selection mode ...
/home/acis/PycharmProjects/pythonRS2
/home/acis
```


z
y
x


Gripper
Stand
Controller
RS Camera



```
        }

        planning_scene_msg_.world.collision_objects.clear();
        planning_scene_msg_.is_diff = true;
        pub_planning_scene_diff_.publish(planning_scene_msg_);


}

void PickPlace::build_workscene()
{
    co_.header.frame_id = "root";
    co_.header.stamp = ros::Time::now();

    // remove table
    co_.id = "box";
    co_.operation = moveit_msgs::CollisionObject::REMOVE;
    pub_co_.publish(co_);

    // add table
    co_.primitives.resize(1);
    co_.primitive_poses.resize(1);
    co_.primitives[0].type = shape_msgs::SolidPrimitive::BOX;
    co_.primitives[0].dimensions.resize(geometric_shapes::SolidPrimitiveDimCount<s
    co_.operation = moveit_msgs::CollisionObject::ADD;

    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_X] = 0.5;
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_Y] = 0.5;
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_Z] = 0.1;
    co_.primitive_poses[0].position.x = 0;
    co_.primitive_poses[0].position.y = 0.0;
    co_.primitive_poses[0].position.z = -0.05;
    pub_co_.publish(co_);

    co_.id = "table";
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_X] = 2.0;
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_Y] = 2.0;
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_Z] = 0.01;
    co_.primitive_poses[0].position.x = 0;
    co_.primitive_poses[0].position.y = 0.0;
    co_.primitive_poses[0].position.z = -0.11;
    pub_co_.publish(co_);

    co_.id = "poster";
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_X] = 0.01;
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_Y] = 1.2192;
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_Z] = 2.0;
    co_.primitive_poses[0].position.x = 0.381;
    co_.primitive_poses[0].position.y = -0.3096;
    co_.primitive_poses[0].position.z = -0.11;
    pub_co_.publish(co_);
```



```
    pub_co_.publish(co_);

    co_.id = "wall";
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_X] = 0.762;
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_Y] = 0.01;
    co_.primitives[0].dimensions[shape_msgs::SolidPrimitive::BOX_Z] = 2.0;
    co_.primitive_poses[0].position.x = 0.0;
    co_.primitive_poses[0].position.y = 0.6048;
    co_.primitive_poses[0].position.z = -0.11;
    pub_co_.publish(co_);
```