

After taking an in-depth look at the squabbling robots last video lecture, we will now dive deeper into the `lookup_transform` API.

### 5.3.5 tf2\_ros API lookup\_transform

```

55 rate_pub = rospy.Rate(0.5)
56 rate_listener = rospy.Rate(10)
57 string_idx = 0
58
59 # Continuously display the transform between the base links of the two robots and the turtlebot.
60 while not rospy.is_shutdown():
61     # This try-except block makes sure any errors we might run into due to non-availability of TF frames
62     # is correctly handled via exception handling.
63     try:
64         # Lookup translation information between robot1_base_link and turtlebot base link.
65         trans_r1 = tfBuffer.lookup_transform('robot1_base_link', 'base_link', rospy.Time())
66         # Update and print string for Robot1.
67         robot1_words = 'Robot1: ' + _robot1_strings[string_idx]
68         print ('%s (x=%4.2f, y=%4.2f).'%(robot1_words, trans_r1.transform.translation.x,
69         trans_r1.transform.translation.y))
70         # Lookup translation information between robot2_base_link and turtlebot base link.
71         trans_r2 = tfBuffer.lookup_transform('robot2_base_link', 'base_link', rospy.Time())
72         # Update and print string for Robot1.
73         robot2_words = 'Robot2: ' + _robot2_strings[string_idx]
74         print ('%s (x=%4.2f, y=%4.2f).'%(robot2_words, trans_r2.transform.translation.x,
75         trans_r2.transform.translation.y))
76         print '-----'
77         string_idx += 1
78         if(string_idx == 3):
79             string_idx = 0
80         rate_pub.sleep()
81     except (tf2_ros.LookupException, tf2_ros.ConnectivityException, tf2_ros.ExtrapolationException):
82         # In case the requested TF is not available, try again after a short duration.
83         rate_listener.sleep()
84         continue
85
86 if __name__ == '__main__':
87     try:
88         # Call the robot_squabblers() function.
89         robot_squabblers()
90     except rospy.ROSInterruptException:
91         pass

```

Video

Subtitles

Subtitles (captions) in other languages than provided can be viewed at [YouTube](https://www.youtube.com/watch?v=...). Select your language in the CC-button of YouTube

### Script

This time, we will be looking at the `transform_object_pose.pt` script, you can also find it on the scripts folder of `hrwros_week5` package

- Again, we see a list of necessary imports, which includes `rospy`, the `tf2_ros` module, and a bunch of necessary message modules.
- We also create a new `tf` buffer and listener, like in the previous video.

**Note: On the files you will get, the tf Buffer and TransformListener are created after the initialization of the ROS Node (Line 77)**

- In the callback function for the logical camera, we can see that the logical camera provides us with a lot of information about the pose of an object in the reference frame of the logical camera itself.
- The API transform can only work with poses that have a timestamp, because the transform might be influenced by a moving robot part. You can't always use the current pose of a link, sometimes you have to look at the past when the pose was generated!
- Thus, the script creates a new header and time stamp information.
- After that, we update the pose information using information from the logical camera.
- Now, we have everything we need for the transform API. We will transform the pose of the object from the camera reference frame to the world reference frame.

---

### Prepare everything needed for the script to work

Make sure the factory environment simulation is running. Don't forget to verify that all robots have shown up!

```
$ roslaunch hrwros_gazebo hrwros_environment.launch
```

- Robot 1 is currently in the FOV of the camera. We don't want this, so let's use MoveIt Commander in a new CCS to move it out of the way.

```
$ rosrun hrwros_week4 hrwros_moveit_commander_cmdline
```

- R1Home is a suitable position for it.

```
> use robot1  
> go R1Home
```

Now the arm is out of the way, let's spawn in an object on the conveyor belt!

- In a new, sourced, CCS shell, execute the following command:

```
$ rosservice call /spawn_object_once
```

- Now you can see a white box object on the conveyor belt near robot 1 in Gazebo. Let's find out what the logical camera can see:

```
$ rostopic echo /hrwros/logical_camera
```

- Search for a model of the type "object" in the output in the console.
  - It will tell you the position and rotation of the white box with respect to the camera frame.

---

### Run the transform\_object\_pose script

- Switch to a new, sourced CCS shell, and run the script:

```
$ rosrn hrwros_week5 transform_object_pose.py
```

- You can now see the output of the script in the terminal.
  - The first pose is the pose of the object in the world reference frame.
  - The second pose is the pose of the object in the camera reference frame.
  - Using the right-hand rule, you can verify the poses are correct!
  - Again, during this course, we will take care of the rotations for you.

---

## Question 1

1 point possible (ungraded)

For the transform API to function correctly, there is a package/python module dependency. Which one is it?

☐ tf2\_geometry\_msgs

☐ geometry\_msgs

☐ gazebo\_msgs

Submit