F1/10 Autonomous Racing

ROS - Transformations and Frames

Lab Session CS4501

Madhur Behl madhur.behl@virginia.edu

Course Website: https://linklab-uva.github.io/autonomousracing/

Git repo for this lab: https://github.com/linklab-uva/f1tenth-course-labs

1. Introduction to `tf` :

Description:

This tutorial will give you a good idea of what tf can do for you. It shows off some of the tf power in a multi-robot example using turtlesim. This also introduces using `tf_echo` , `view_frames` , and `rviz` .

Keywords:

transforms, coordinate frames

2. Compiling the Demo

Let's start by getting any dependencies and compiling the demo package.

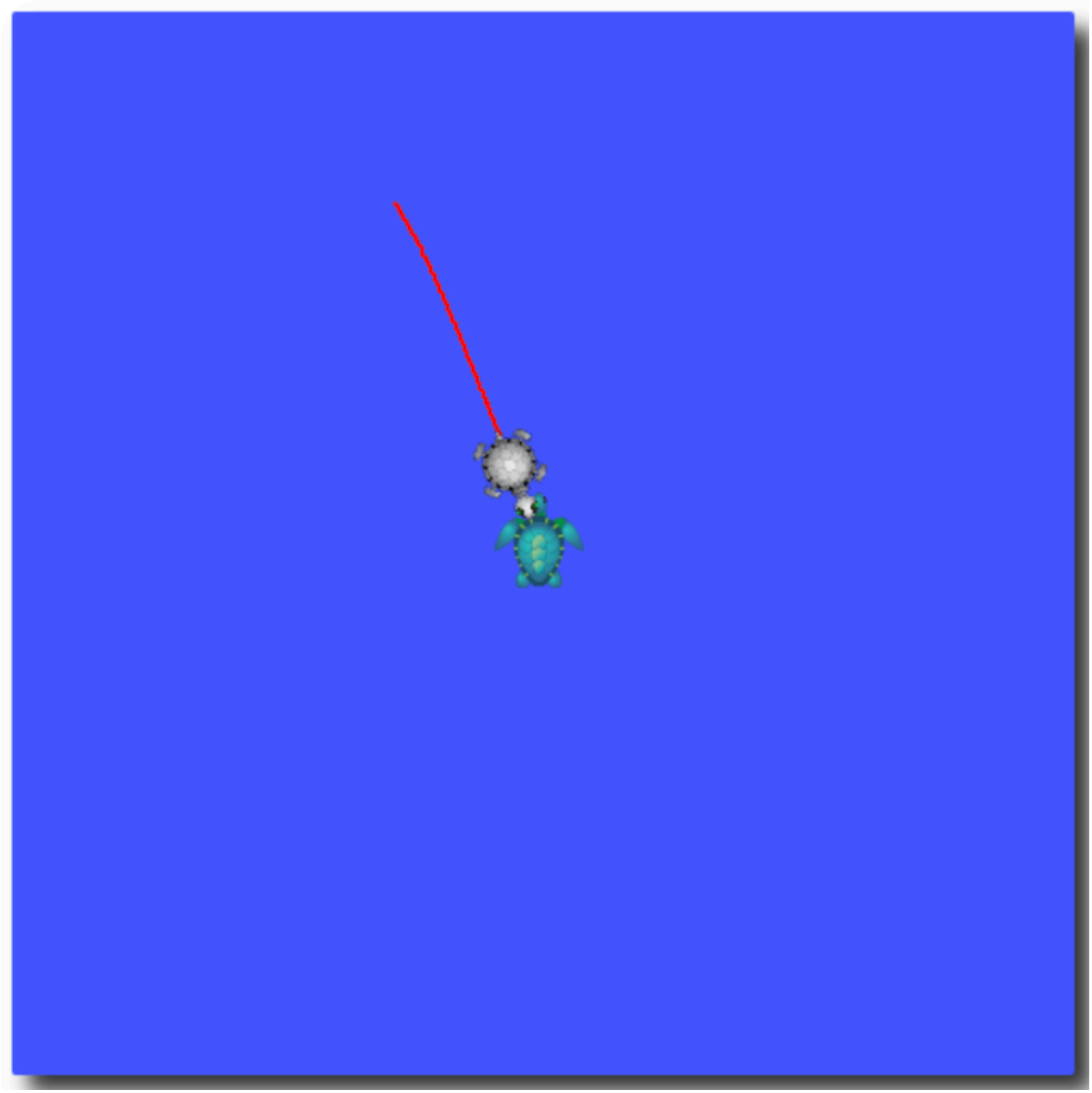
```
$ rosdep install turtle_tf rviz
$ rosmake turtle_tf rviz
```

3. Running the Demo

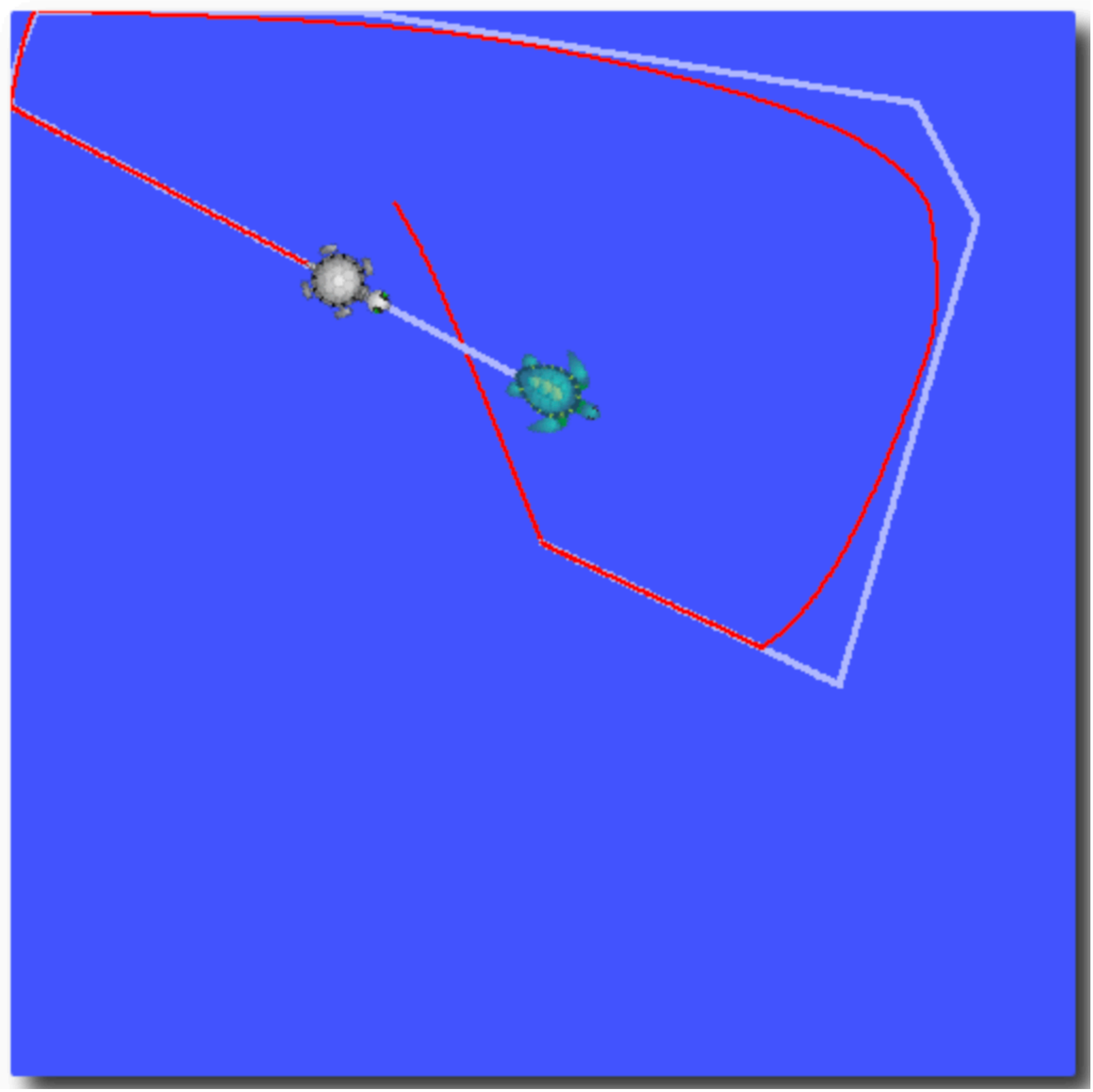
Now that we're done compiling the turtle_tf tutorial package let's run the demo.

```
$ roslaunch turtle_tf turtle_tf_demo.launch
```

You will see the turtlesim start with two turtles.



Once the turtlesim is started you can drive the center turtle around in the turtlesim using the keyboard arrow keys, select the roslaunch terminal window so that your keystrokes will be captured to drive the turtle.



As you can see that one turtle will continuously move to follow the turtle you are driving around.

4. What is Happening

This demo is using the tf library to create three coordinate frames: a world frame, a turtle1 frame, and a turtle2 frame. This tutorial uses a **tf broadcaster** to publish the turtle coordinate frames and a **tf listener** to compute the difference in the turtle frames and move one turtle to follow the other.

5. Tf tools

Now let’s look at how tf is being used to create this demo. We can use tf tools to look at what tf is doing behind the scenes.

5.1 Using view_frames

`view_frames` creates a diagram of the frames being broadcast by tf over ROS.

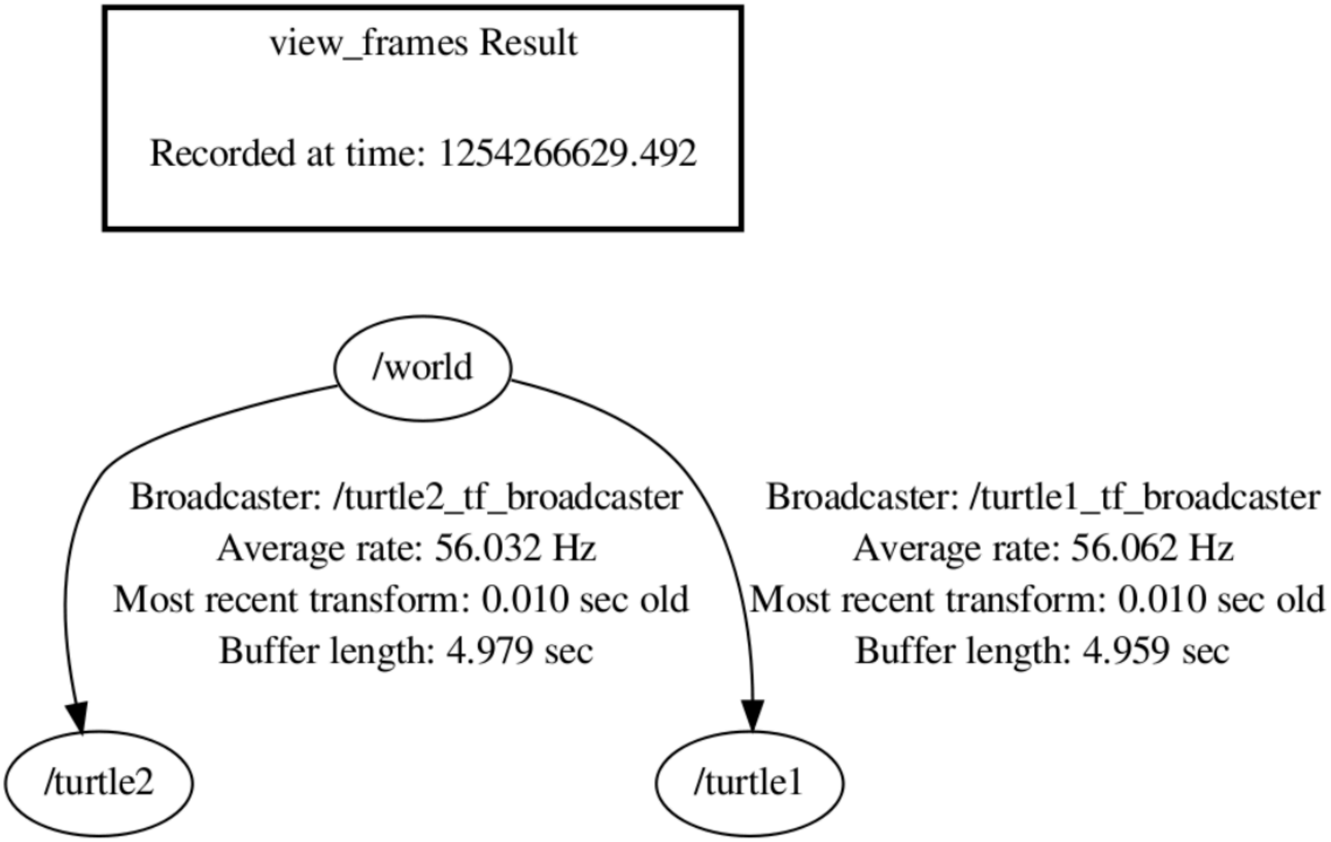
```
$ rosrun tf view_frames
```

You will see something similar to:

```
Transform Listener initing
Listening to /tf for 5.000000 seconds
Done Listening
dot - Graphviz version 2.16 (Fri Feb  8 12:52:03 UTC 2008)

Detected dot version 2.16
frames.pdf generated
```

Here a tf listener is listening to the frames that are being broadcast over ROS and drawing a tree of how the frames are connected. To view the tree:

```
$ evince frames.pdf
```


view_frames Result
Recorded at time: 1254266629.492
/world
Broadcaster: /turtle2_tf_broadcaster
Average rate: 56.032 Hz
Most recent transform: 0.010 sec old
Buffer length: 4.979 sec
Broadcaster: /turtle1_tf_broadcaster
Average rate: 56.062 Hz
Most recent transform: 0.010 sec old
Buffer length: 4.959 sec
/turtle2
/turtle1

Here we can see the three frames that are broadcast by `tf` the `world`, `turtle1`, and `turtle2` and that world is the parent of the turtle1 and turtle2 frames.
`view_frames` also report some diagnostic information about when the oldest and most recent frame transforms were received and how fast the tf frame is published to tf for debugging purposes.

5.1 Using tf_echo

`tf_echo` reports the transform between any two frames broadcast over ROS.

Usage:

```
rosrun tf tf_echo [reference_frame] [target_frame]
```

Let’s look at the transform of the turtle2 frame with respect to turtle1 frame which is equivalent to

```
$ rosrun tf tf_echo turtle1 turtle2
```

You will see the transform displayed as the tf_echo listener receives the frames broadcast over ROS.

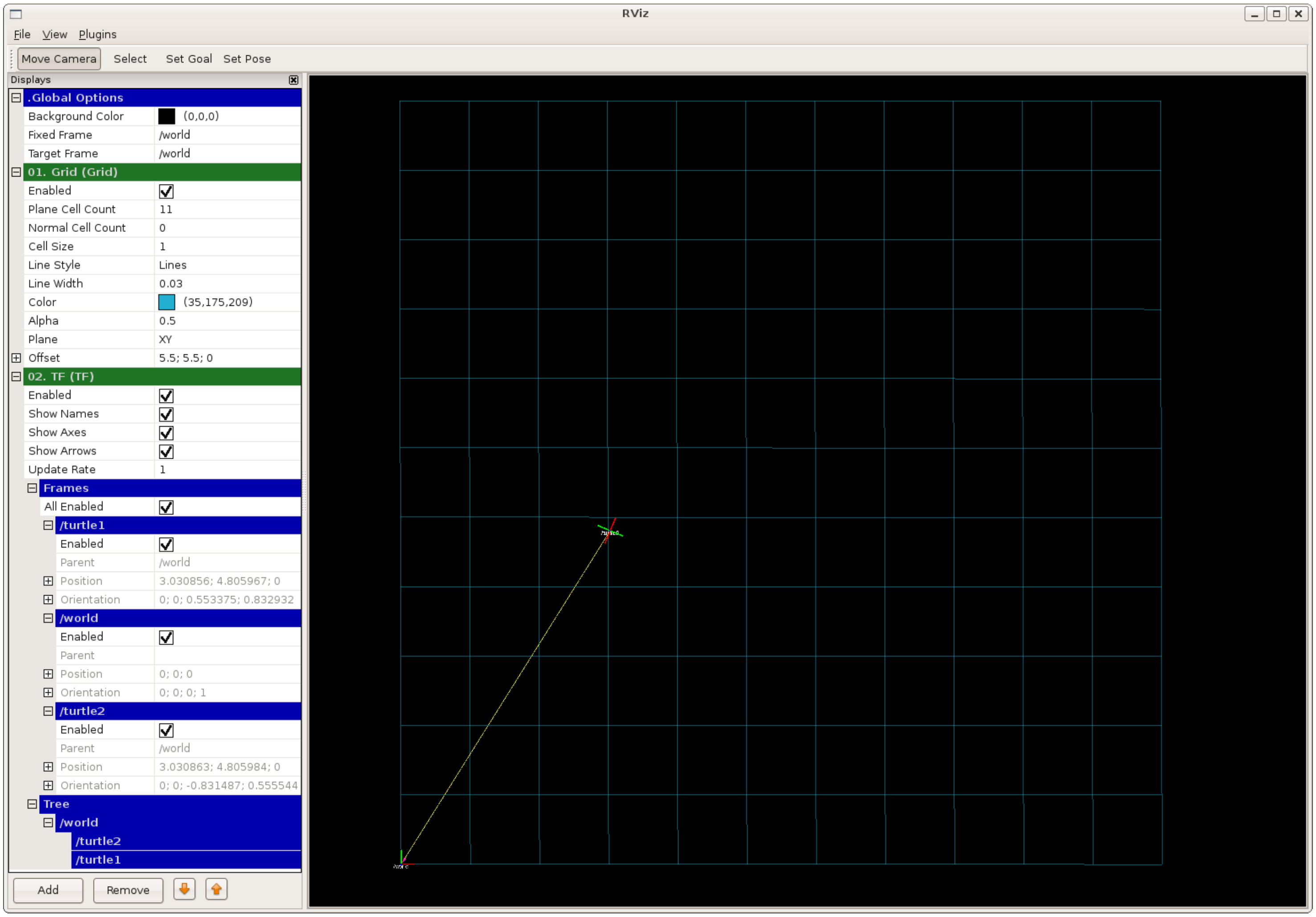
```
[ INFO] 1253585683.529245000: Started node [/tf_echo_1253585683508144000], pid [24418],
Exception thrown:Frame id /turtle1 does not exist! When trying to transform between /tu
The current list of frames is:

Success at 1253585684.557003974
[0.000000 0.000000 0.140754 0.990045] Euler(0.282446 -0.000000 0.000000)
Translation: [-0.000036 -0.000010 0.000000]
Success at 1253585685.544698953
[0.000000 0.000000 0.140754 0.990045] Euler(0.282446 -0.000000 0.000000)
Translation: [-0.000036 -0.000010 0.000000]
Success at 1253585686.557049989
[0.000000 0.000000 0.140754 0.990045] Euler(0.282446 -0.000000 0.000000)
Translation: [-0.000036 -0.000010 0.000000]
Success at 1253585687.552628993
[0.000000 0.000000 0.140754 0.990045] Euler(0.282446 -0.000000 0.000000)
Translation: [-0.000036 -0.000010 0.000000]
Success at 1253585688.553683042
[0.000000 0.000000 0.140754 0.990045] Euler(0.282446 -0.000000 0.000000)
Translation: [-0.000036 -0.000010 0.000000]
Success at 1253585688.910640001
[0.000000 0.000000 0.140754 0.990045] Euler(0.282446 -0.000000 0.000000)
Translation: [-0.000036 -0.000010 0.000000]
```

As you drive your turtle around you will see the transform change as the two turtles move relative to each other.

6. rviz and tf

rviz is a visualization tool that is useful for examining tf frames. Let’s look at our turtle frames using rviz. Let’s start rviz with the turtle_tf configuration file using the -d option for rviz:

```
$ rosrun rviz rviz -d `rospack find turtle_tf`/rviz/turtle_rviz.vcg
```



.Global Options	
Background Color	■ (0,0,0)
Fixed Frame	/world
Target Frame	/world
01. Grid (Grid)	
Enabled	☑
Plane Cell Count	11
Normal Cell Count	0
Cell Size	1
Line Style	Lines
Line Width	0.03
Color	■ (35,175,209)
Alpha	0.5
Plane	XY
Offset	5.5; 5.5; 0
02. TF (TF)	
Enabled	☑
Show Names	☑
Show Axes	☑
Show Arrows	☑
Update Rate	1
Frames	
All Enabled	☑
/turtle1	
Enabled	☑
Parent	/world
Position	3.030856; 4.805967; 0
Orientation	0; 0; 0.553375; 0.832932
/world	
Enabled	☑
Parent	
Position	0; 0; 0
Orientation	0; 0; 0; 1
/turtle2	
Enabled	☑
Parent	/world
Position	3.030863; 4.805984; 0
Orientation	0; 0; -0.831487; 0.555544
Tree	
/world	
/turtle2	
/turtle1	

In the side bar you will see the frames broadcast by tf. As you drive the turtle around you will see the frames move in rviz.

To Do:

1. Launch the ROS F1/10 simulator

```
$ roslaunch f1tenth_sim simulator.launch run_gazebo:=true
```

2. Use view_frames to visualize the tf tree for the F1/10 racecar

3. Use tf*echo tool to echo the transform between the base*link and the laser on the F1/10 car

4. Visualize the TF frames from the F1/10 car in `rviz`