ROS Navigation in 5 Days

Unit 3: Solutions

Index

Solution Exercise 3.3

- Exercise 3.3 -

- Launch File: change_map.launch -

In [ ]:

```
<?xml version="1.0"?>
<launch>

  <arg name="map_file" default="$(find husky_navigation)/maps/playpen_map.yaml
  <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_f

  <arg name="use_map_topic" default="true"/>
  <arg name="scan_topic" default="scan" />

  <node pkg="amcl" type="amcl" name="amcl">
    <param name="use_map_topic" value="$(arg use_map_topic)"/>
    <!-- Publish scans from best pose at a max of 10 Hz -->
    <param name="odom_model_type" value="diff"/>
    <param name="odom_alpha5" value="0.1"/>
    <param name="gui_publish_rate" value="10.0"/>
    <param name="laser_max_beams" value="60"/>
    <param name="laser_max_range" value="12.0"/>
    <param name="min_particles" value="500"/>
    <param name="max_particles" value="2000"/>
    <param name="kld_err" value="0.05"/>
    <param name="kld_z" value="0.99"/>
    <param name="odom_alpha1" value="0.2"/>
    <param name="odom_alpha2" value="0.2"/>
    <!-- translation std dev, m -->
    <param name="odom_alpha3" value="0.2"/>
    <param name="odom_alpha4" value="0.2"/>
    <param name="laser_z_hit" value="0.5"/>
    <param name="laser_z_short" value="0.05"/>
    <param name="laser_z_max" value="0.05"/>
    <param name="laser_z_rand" value="0.5"/>
    <param name="laser_sigma_hit" value="0.2"/>
    <param name="laser_lambda_short" value="0.1"/>
    <param name="laser_model_type" value="likelihood_field"/>
    <!-- <param name="laser_model_type" value="beam"/> -->
    <param name="laser_likelihood_max_dist" value="2.0"/>
    <param name="update_min_d" value="0.25"/>
    <param name="update_min_a" value="0.2"/>
    <param name="odom_frame_id" value="odom"/>
    <param name="resample_interval" value="1"/>
    <!-- Increase tolerance because the computer can get quite busy -->
    <param name="transform_tolerance" value="1.0"/>
    <param name="recovery_alpha_slow" value="0.0"/>
    <param name="recovery_alpha_fast" value="0.0"/>
    <remap from="scan" to="$(arg scan_topic)"/>
  </node>
```

```
</launch>
```

Solution Exercise 3.5

- Exercise 3.5 -

- Launch File: get_pose_service.launch -

In [ ]:

```
<launch>
    <node pkg="get_pose" type="get_pose_service.py" name="service_server" outp
    </node>
</launch>
```

- Python File: get_pose_service.py -

In [ ]:

```
#! /usr/bin/env python

import rospy
from std_srvs.srv import Empty, EmptyResponse # Import the service message pyt
from geometry_msgs.msg import PoseWithCovarianceStamped, Pose

robot_pose = Pose()

def service_callback(request):
    print("Robot Pose:")
    print(robot_pose)
    return EmptyResponse() # the service Response class, in this case EmptyRes

def sub_callback(msg):
    global robot_pose
    robot_pose = msg.pose.pose


rospy.init_node('service_server')
my_service = rospy.Service('/get_pose_service', Empty , service_callback) # cr
sub_pose = rospy.Subscriber('/amcl_pose', PoseWithCovarianceStamped, sub_callb
rospy.spin() # mantain the service open.
```

Solution Exercise 3.8

- Exercise 3.8 -

- Launch File: my_amcl_launch.launch -

In [ ]:

```
<launch>

  <arg name="map_file" default="$(find husky_navigation)/maps/my_map.yaml"/>
  <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_f

  <arg name="use_map_topic" default="true"/>
  <arg name="scan_topic" default="scan" />

  <node pkg="amcl" type="amcl" name="amcl">
    <param name="use_map_topic" value="$(arg use_map_topic)"/>
    <!-- Publish scans from best pose at a max of 10 Hz -->
    <param name="odom_model_type" value="diff"/>
    <param name="odom_alpha5" value="0.1"/>
    <param name="gui_publish_rate" value="10.0"/>
    <param name="laser_max_beams" value="60"/>
    <param name="laser_max_range" value="12.0"/>
    <param name="min_particles" value="1"/>
    <param name="max_particles" value="5"/>
    <param name="kld_err" value="0.05"/>
    <param name="kld_z" value="0.99"/>
    <param name="odom_alpha1" value="0.2"/>
    <param name="odom_alpha2" value="0.2"/>
    <!-- translation std dev, m -->
    <param name="odom_alpha3" value="0.2"/>
    <param name="odom_alpha4" value="0.2"/>
    <param name="laser_z_hit" value="0.5"/>
    <param name="laser_z_short" value="0.05"/>
    <param name="laser_z_max" value="0.05"/>
    <param name="laser_z_rand" value="0.5"/>
    <param name="laser_sigma_hit" value="0.2"/>
    <param name="laser_lambda_short" value="0.1"/>
    <param name="laser_model_type" value="likelihood_field"/>
    <!-- <param name="laser_model_type" value="beam"/> -->
    <param name="laser_likelihood_max_dist" value="2.0"/>
    <param name="update_min_d" value="0.25"/>
    <param name="update_min_a" value="0.2"/>
    <param name="odom_frame_id" value="odom"/>
    <param name="resample_interval" value="1"/>
    <!-- Increase tolerance because the computer can get quite busy -->
    <param name="transform_tolerance" value="1.0"/>
    <param name="recovery_alpha_slow" value="0.0"/>
    <param name="recovery_alpha_fast" value="0.0"/>
    <remap from="scan" to="$(arg scan_topic)"/>
  </node>
```

```
</launch>
```

Solution Exercise 3.9

- Exercise 3.9 -

- Launch File: my_amcl_launch.launch -

In [ ]:

```
<launch>

  <arg name="map_file" default="$(find husky_navigation)/maps/my_map.yaml"/>
  <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_f

  <arg name="use_map_topic" default="true"/>
  <arg name="scan_topic" default="scan" />

  <node pkg="amcl" type="amcl" name="amcl">
    <param name="use_map_topic" value="$(arg use_map_topic)"/>
    <!-- Publish scans from best pose at a max of 10 Hz -->
    <param name="odom_model_type" value="diff"/>
    <param name="odom_alpha5" value="0.1"/>
    <param name="gui_publish_rate" value="10.0"/>
    <param name="laser_max_beams" value="60"/>
    <param name="laser_max_range" value="1.0"/>
    <param name="min_particles" value="500"/>
    <param name="max_particles" value="2000"/>
    <param name="kld_err" value="0.05"/>
    <param name="kld_z" value="0.99"/>
    <param name="odom_alpha1" value="0.2"/>
    <param name="odom_alpha2" value="0.2"/>
    <!-- translation std dev, m -->
    <param name="odom_alpha3" value="0.2"/>
    <param name="odom_alpha4" value="0.2"/>
    <param name="laser_z_hit" value="0.5"/>
    <param name="laser_z_short" value="0.05"/>
    <param name="laser_z_max" value="0.05"/>
    <param name="laser_z_rand" value="0.5"/>
    <param name="laser_sigma_hit" value="0.2"/>
    <param name="laser_lambda_short" value="0.1"/>
    <param name="laser_model_type" value="likelihood_field"/>
    <!-- <param name="laser_model_type" value="beam"/> -->
    <param name="laser_likelihood_max_dist" value="2.0"/>
    <param name="update_min_d" value="0.25"/>
    <param name="update_min_a" value="0.2"/>
    <param name="odom_frame_id" value="odom"/>
    <param name="resample_interval" value="1"/>
    <!-- Increase tolerance because the computer can get quite busy -->
    <param name="transform_tolerance" value="1.0"/>
    <param name="recovery_alpha_slow" value="0.0"/>
    <param name="recovery_alpha_fast" value="0.0"/>
    <remap from="scan" to="$(arg scan_topic)"/>
  </node>
```

```
</launch>
```

Solution Exercise 3.10

- Exercise 3.10 -

- Launch File: my_amcl_launch.launch -

In [ ]:

```
<?xml version="1.0"?>
<launch>

  <arg name="scan_topic" default="scan" />
  <arg name="map_file" default="$(find husky_navigation)/maps/my_map.yaml"/>

  <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_f

  <node pkg="amcl" type="amcl" name="amcl">
    <rosparam file="$(find my_amcl_launcher)/params/my_amcl_params.yaml" comma
    <remap from="scan" to="$(arg scan_topic)"/>
  </node>

</launch>
```

- Params File: my_amcl_params.yaml -

In [ ]:
```
use_map_topic: true
odom_model_type: diff
odom_frame_id: odom

gui_publish_rate: 10.0
min_particles: 500
max_particles: 2000
kld_err: 0.05
update_min_d: 0.25
update_min_a: 0.2
resample_interval: 1
transform_tolerance: 1.0

laser_max_beams: 60
laser_max_range: 12.0
laser_z_hit: 0.5
laser_z_short: 0.05
laser_z_max: 0.05
laser_z_rand: 0.5
```

Solution Exercise 3.11

- Exercise 3.11 -

- Launch File: init_particles_caller.launch -

In [ ]:
```
<launch>
    <node pkg="initialize_particles" type="init_particles_caller.py" name="ser

    </node>
</launch>
```

- Python File: init_particles_caller.py -

In [ ]:

```
#! /usr/bin/env python

import rospy
from std_srvs.srv import Empty, EmptyRequest
import sys

rospy.init_node('service_client')
rospy.wait_for_service('/global_localization')
disperse_particles_service = rospy.ServiceProxy('/global_localization', Empty)
msg = EmptyRequest()
result = disperse_particles_service(msg)
print(result)
```

Solution Exercise 3.12

- Exercise 3.12 -

- Python File: square_move.py -

In [ ]:

```
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist, PoseWithCovarianceStamped
from std_srvs.srv import Empty, EmptyRequest
import time
import math

class MoveHusky():

    def __init__(self):

        # Init Publisher
        self.husky_vel_publisher = rospy.Publisher('/cmd_vel', Twist, queue_si
        self.cmd = Twist()
        # Init Subscriber
        self.amcl_pose_sub = rospy.Subscriber('/amcl_pose', PoseWithCovariance
        self.sub_msg = PoseWithCovarianceStamped()
        # Initialize Service Client
        rospy.wait_for_service('/global_localization')
        self.disperse_particles_service = rospy.ServiceProxy('/global_localiza
        self.srv_request = EmptyRequest()
        # Other stuff
        self.ctrl_c = False
        rospy.on_shutdown(self.shutdownhook)
        self.rate = rospy.Rate(10)

    def shutdownhook(self):

        # works better than the rospy.is_shut_down()
        self.stop_husky()
        self.ctrl_c = True

    def stop_husky(self):

        rospy.loginfo("Shutdown time! Stop the robot")
        self.cmd.linear.x = 0.0
        self.cmd.angular.z = 0.0
        i = 0

        while i < 20:
            self.husky_vel_publisher.publish(self.cmd)
            self.rate.sleep()
            i += 1

    def move_forward(self, linear_speed=0.5, angular_speed=0.0):
```

```
        self.cmd.linear.x = linear_speed
        self.cmd.angular.z = angular_speed
        i = 0

        while i < 50:
            self.husky_vel_publisher.publish(self.cmd)
            self.rate.sleep()
            i += 1

    def turn(self, linear_speed=0.0, angular_speed=0.8):

        self.cmd.linear.x = linear_speed
        self.cmd.angular.z = angular_speed
        i = 0

        while i < 25:
            self.husky_vel_publisher.publish(self.cmd)
            self.rate.sleep()
            i += 1

    def move_square(self):

        i = 0

        while not self.ctrl_c and i < 4:
            # Move Forwards
            rospy.loginfo("######## Going Forwards...")
            self.move_forward()
            self.stop_husky()
            # Turn
            rospy.loginfo("######## Turning...")
            self.turn()
            self.stop_husky()
            i += 1

        self.stop_husky()
        rospy.loginfo("######## Finished Moving in a Square")

    def call_service(self):

        rospy.loginfo("######## Calling Service...")
        result = self.disperse_particles_service(self.srv_request)

    def sub_callback(self, msg):
```

```
        self.sub_msg = msg

    def calculate_covariance(self):

        rospy.loginfo("######## Calculating Covariance...")
        cov_x = self.sub_msg.pose.covariance[0]
        cov_y = self.sub_msg.pose.covariance[7]
        cov_z = self.sub_msg.pose.covariance[35]
        rospy.loginfo("## Cov X: " + str(cov_x) + " ## Cov Y: " + str(cov_y) +
        cov = (cov_x+cov_y+cov_z)/3

        return cov


if __name__ == '__main__':
    rospy.init_node('move_husky_node', anonymous=True)
    MoveHusky_object = MoveHusky()

    cov = 1

    while cov > 0.65:
        MoveHusky_object.call_service()
        MoveHusky_object.move_square()
        cov = MoveHusky_object.calculate_covariance()
        rospy.loginfo("######## Total Covariance: " + str(cov))
        if cov > 0.65:
            rospy.loginfo("######## Total Covariance is greater than 0.65. Rep
        else:
            rospy.loginfo("######## Total Covariance is lower than 0.65. Robot
            rospy.loginfo("######## Exiting...")
```