

## Problem 1. PID controller node for line following

Question 1.1.a. `__init__()` method (no tuning)

```

14 class ControllerLine(Node):
15     """Simple PID controller that allows robot to follow line while moving forward."""
16     def __init__(self):
17         """Initialize node per assignment instructions, specifics commented below."""
18         super().__init__('controller_line')
19         #1. Initialize attributes
20         self.lin_speed = 0.0
21         self.gain_kp = 0.0
22         self.gain_kd = 0.0
23         self.gain_ki = 0.0
24         #2. Initialize subscriber and two publishers
25         self.subscriber = self.create_subscription(PointStamped,
26                                                     '/image/centroid',
27                                                     self.subscriber_callback, 10)
28         self.twist_publisher = self.create_publisher(Twist,
29                                                     'robot_twist',
30                                                     10)
31         self.error_publisher = self.create_publisher(Float64,
32                                                     'control_error',
33                                                     10)
34         #3. Initialize controller PID object
35         self.pid = controller.PID(gain_kp=self.gain_kp,
36                                   gain_kd=self.gain_kd,
37                                   gain_ki=self.gain_ki)
38         #4. Initialize robot_model.StampedMsgRegister object
39         self.stamped_msg_register = robot_model.StampedMsgRegister()

```

Question 1.1.b. `__init__()` method (with tuning)

```


14 class ControllerLine(Node):
15     """Simple PID controller that allows robot to follow line while moving forward."""
16     def __init__(self):
17         """Initialize node per assignment instructions, specifics commented below."""
18         super().__init__('controller_line')
19         #1. Initialize attributes
20         self.lin_speed = 0.055
21         self.gain_kp = 0.004
22         self.gain_kd = 0.003
23         self.gain_ki = 0.003
24         #2. Initialize subscriber and two publishers
25         self.subscriber = self.create_subscription(PointStamped,
26                                                     '/image/centroid',
27                                                     self.subscriber_callback, 10)
28         self.twist_publisher = self.create_publisher(Twist,
29                                                     'robot_twist',
30                                                     10)
31         self.error_publisher = self.create_publisher(Float64,
32                                                     'control_error',
33                                                     10)
34         #3. Initialize controller PID object
35         self.pid = controller.PID(gain_kp=self.gain_kp,
36                                   gain_kd=self.gain_kd,
37                                   gain_ki=self.gain_ki)
38         #4. Initialize robot_model.StampedMsgRegister object
39         self.stamped_msg_register = robot_model.StampedMsgRegister()

```

### Question 1.2. subscriber\_callback()

```
41 ... def subscriber_callback(self, point_stamped_msg):
42 ...     """Perform assignment specified operations, commented below."""
43 ...     #1. Compute error_signal
44 ...     IMG_WIDTH = 320
45 ...     error_signal = Float64()
46 ...     error_signal.data = point_stamped_msg.point.x - (0.5 * IMG_WIDTH)
47 ...     #2. Publish error_signal to topic control_error
48 ...     self.error_publisher.publish(error_signal)
49 ...     #3. Obtain time_delay via self.stamped_msg_register
50 ...     time_delay = self.stamped_msg_register.replace_and_compute_delay(point_stamped_msg)[0]
51 ...     #4. Initialize Twist msg
52 ...     msg = Twist()
53 ...     #5. Set msg.linear.x
54 ...     msg.linear.x = self.lin_speed
55 ...     #6. Set msg.angular.z
56 ...     msg.angular.z = self.pid.proportional(error_signal=error_signal.data) \
57 ...     + self.pid.derivative(error_signal=error_signal.data, time_delay=time_delay) \
58 ...     + self.pid.integral(error_signal=error_signal.data, time_delay=time_delay)
59 ...     #7. Publish msg (to topic robot_twist)
60 ...     self.twist_publisher.publish(msg)
```

### Optional Question 1.1. controller\_line\_launch.py

 controller\_line\_launch.py U X

src > me416\_lab > launch >  controller\_line\_launch.py > ...

```
1 from launch_ros.actions import Node
2 from launch import LaunchDescription
3
4
5 def generate_launch_description():
6     """
7     Launches motor_command, image_segment, and controller_line
8     """
9     ldesc = LaunchDescription()
10    motor_command = Node(package="me416_lab", executable="motor_command")
11    image_segment = Node(package="me416_lab", executable="image_segment")
12    controller_line = Node(package="me416_lab", executable="controller_line")
13    ldesc.add_action(motor_command)
14    ldesc.add_action(image_segment)
15    ldesc.add_action(controller_line)
16    return ldesc
17
```