Activities Webots Nov 14 22:32

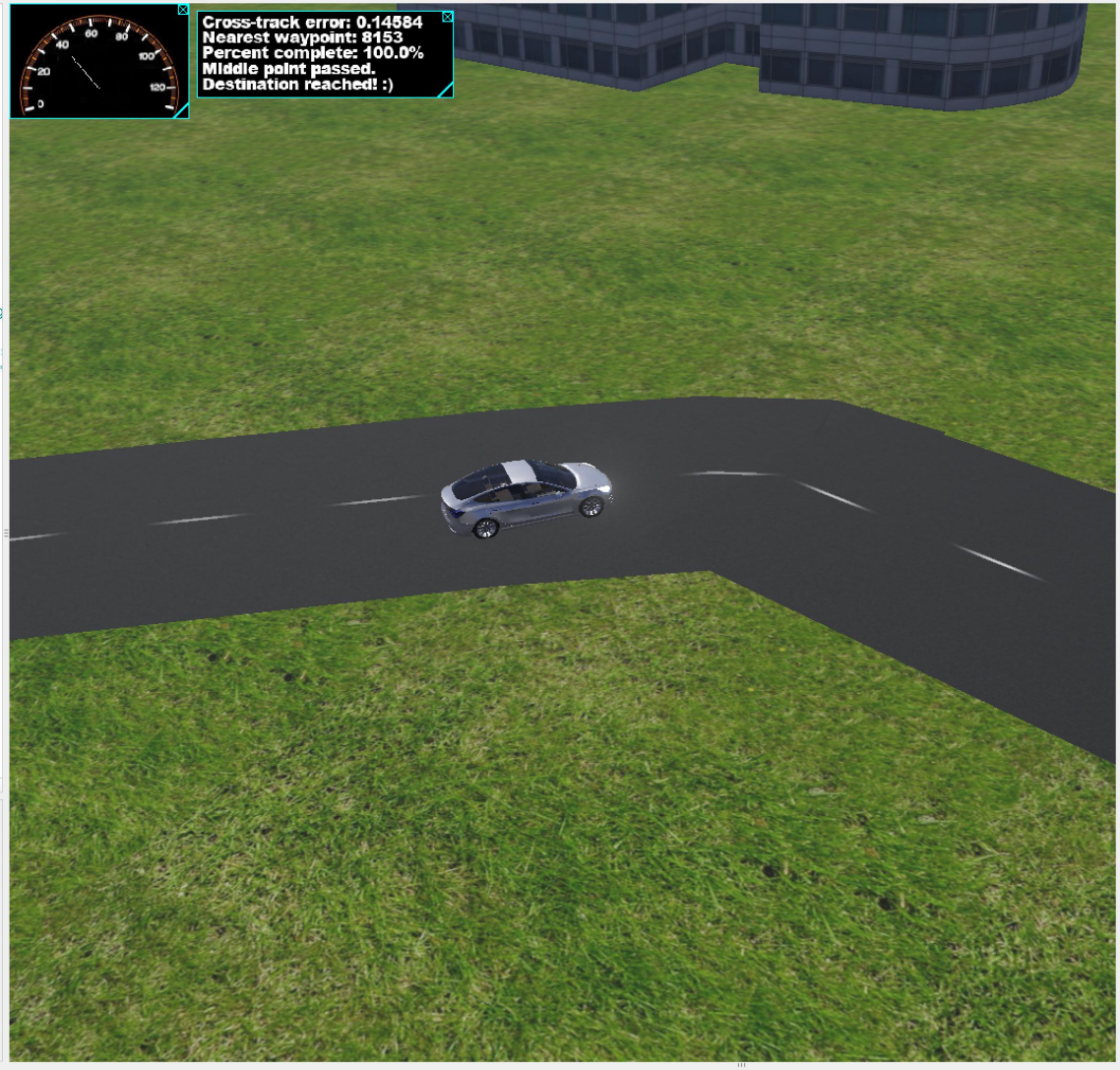
/root/Downloads/P3_student/P3_student/P3-LQR/worlds/automotive_new.wbt (P3-LQR) - Webots R2021b (on 1125836cf4b5)

File Edit View Simulation Build Overlays Tools Wizards Help

Simulation View

5:40:15:552 - 2923x

- WorldInfo
- Viewpoint
- TexturedBackground
- TexturedBackgroundLight
- DEF TESLA Robot
- Floor "floor"
- Road "road"
- Slide "slide"
- Swing "swing"
- Forest
- SimpleBuilding "building(2)"
- SimpleBuilding "Newell-Simon Hall"
- SimpleBuilding "Scott Hall"
- SimpleBuilding "Doherty Hall"
- SimpleBuilding "Wean Hall"
- SimpleBuilding "Roberts Engineering
- SimpleBuilding "Hamerschlag Hall"
- SimpleBuilding "Tepper School of Bu
- SimpleBuilding "College of Fine Arts"
- SimpleBuilding "Hunt Library"
- SimpleBuilding "Baker Hall"
- SimpleBuilding "Porter Hall"
- SimpleBuilding "Scaife Hall"



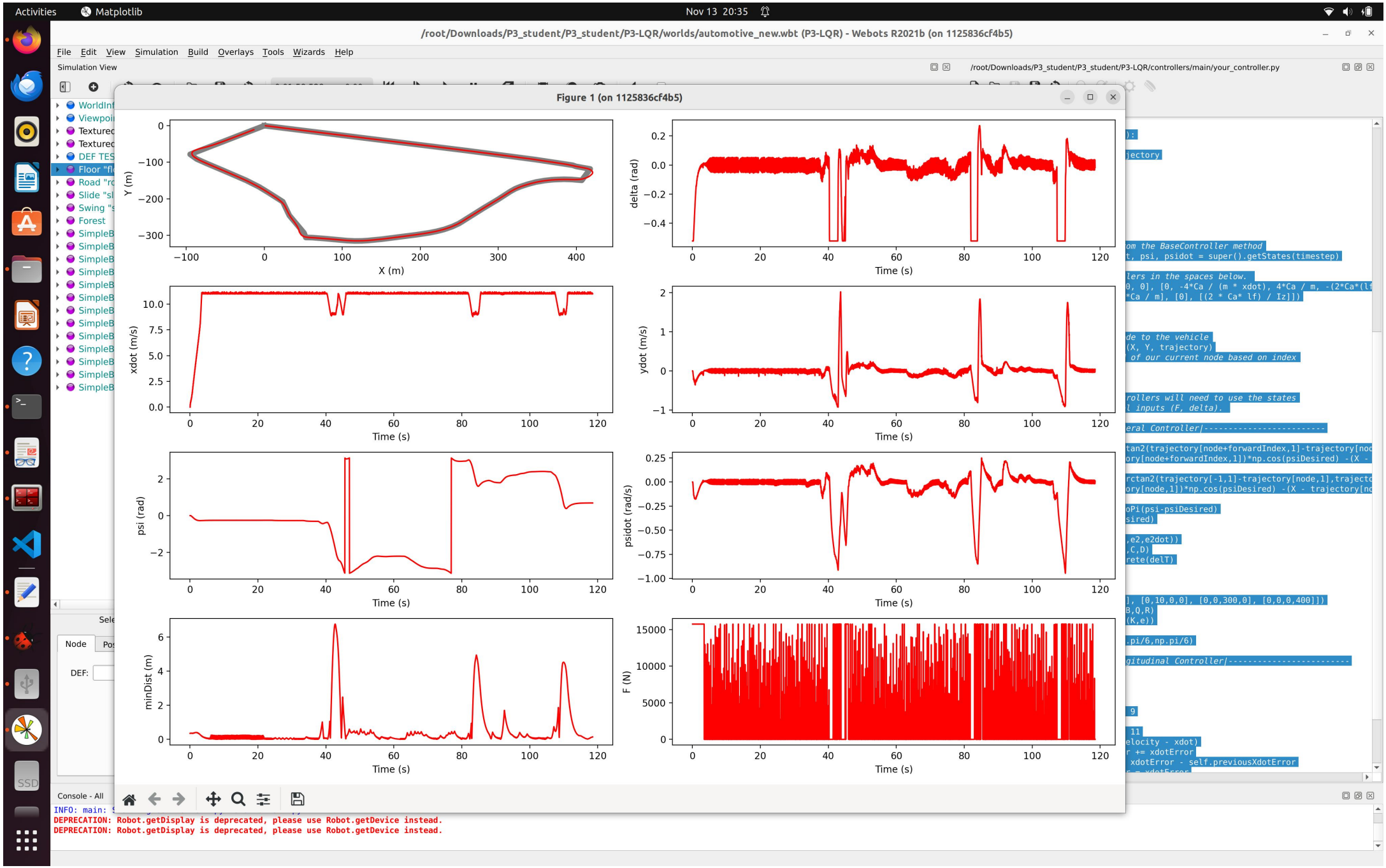
/root/Downloads/P3_student/P3_student/P3-LQR/controllers/main/your_controller.py

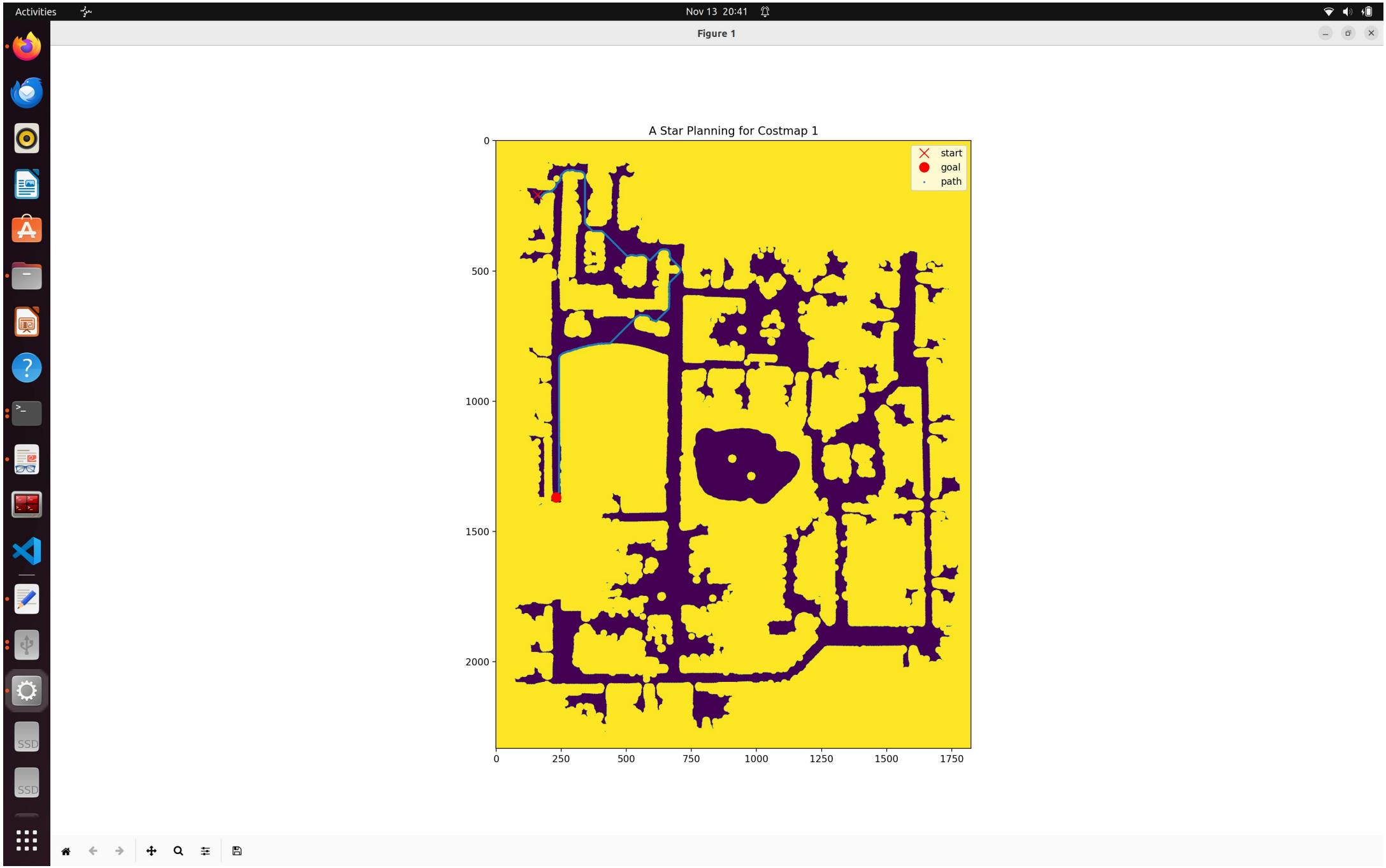
your_controller.py

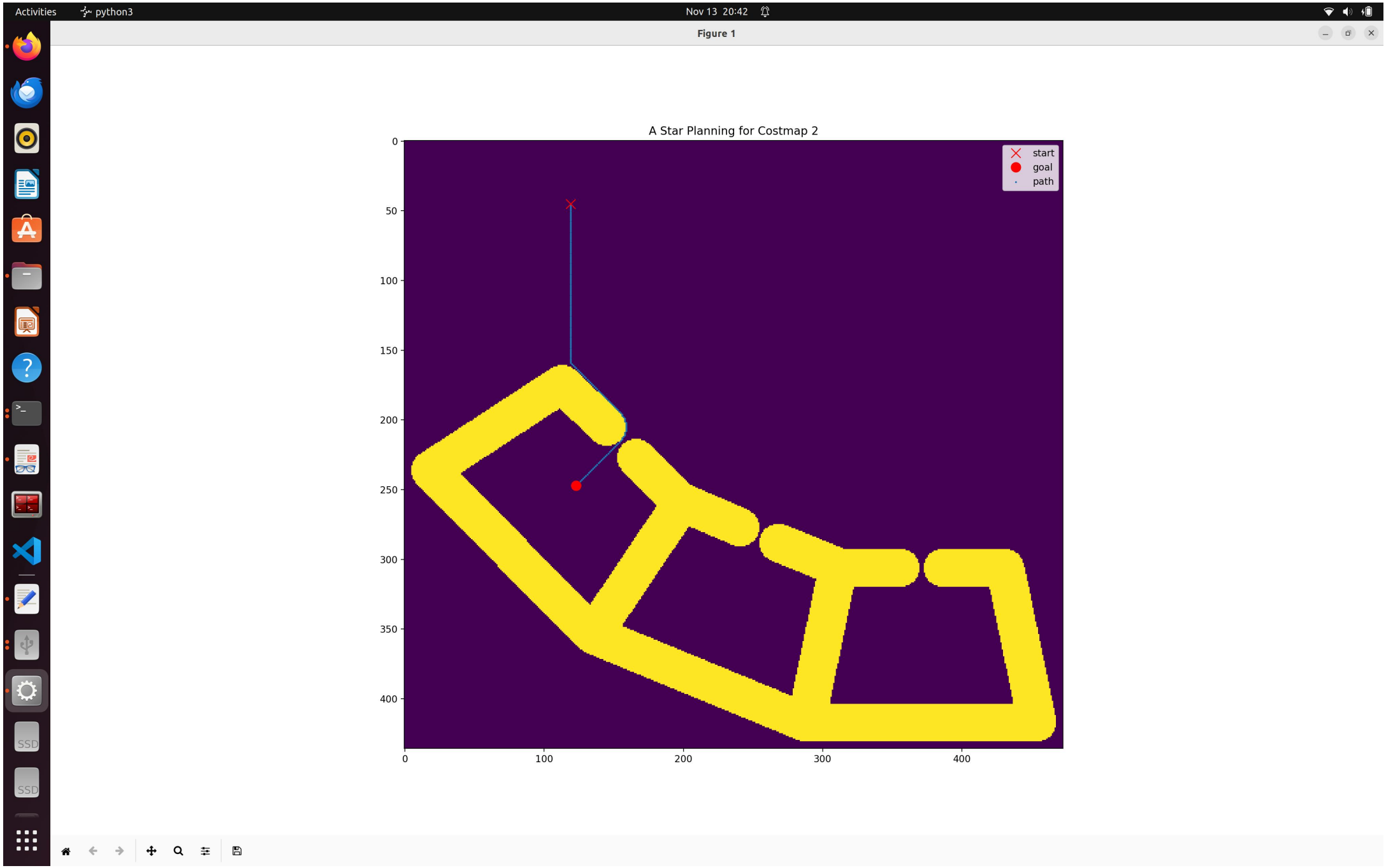
```python
# Fill in the respective functions to implement the controller

# Import libraries
import numpy as np
from base_controller import BaseController
from scipy import signal, linalg
from scipy.signal import StateSpace, lsim, dlsim
from util import *

# CustomController class (inherits from BaseController)
class CustomController(BaseController):

    def __init__(self, trajectory):

        super().__init__(trajectory)

        # Define constants
        # These can be ignored in P1
        self.lr = 1.39
        self.lf = 1.55
        self.Ca = 20000
        self.Iz = 25854
        self.m = 1888.6
        self.g = 9.81
        self.integralPsiError = 0
        self.previousPsiError = 0
        self.previousXdotError = 0

        # Add additional member variables according to your need here.

    def dlqr(self,A,B,Q,R):
        S = np.matrix(linalg.solve_discrete_are(A, B, Q, R))
        K = -np.matrix(linalg.inv(B.T@S@B+R)@(B.T@S@A))
        return K
    def update(self, timestep):

        trajectory = self.trajectory

        lr = self.lr
        lf = self.lf
        Ca = self.Ca
        Iz = self.Iz
        m = self.m
        g = self.g

        # Fetch the states from the BaseController method
        delT, X, Y, xdot, ydot, psi, psidot = super().getStates(timestep)

        # Design your controllers in the spaces below.
        A = np.array([[0, 1, 0, 0], [0, -4*Ca / (m * xdot), 4*Ca / m, -(2*Ca*(lf
        B = np.array([[0], [2*Ca / m], [0], [(2 * Ca* lf) / Iz]])
        C = np.eye(4)
        D = np.zeros((4,1))

        # Find the closest node to the vehicle
        _, node = closestNode(X, Y, trajectory)
# Choose a node that is ahead of our current node based on index
        forwardIndex = 100


        # Remember, your controllers will need to use the states
        # to calculate control inputs (F, delta).

        # ---------------|Lateral Controller|-------------------------
```

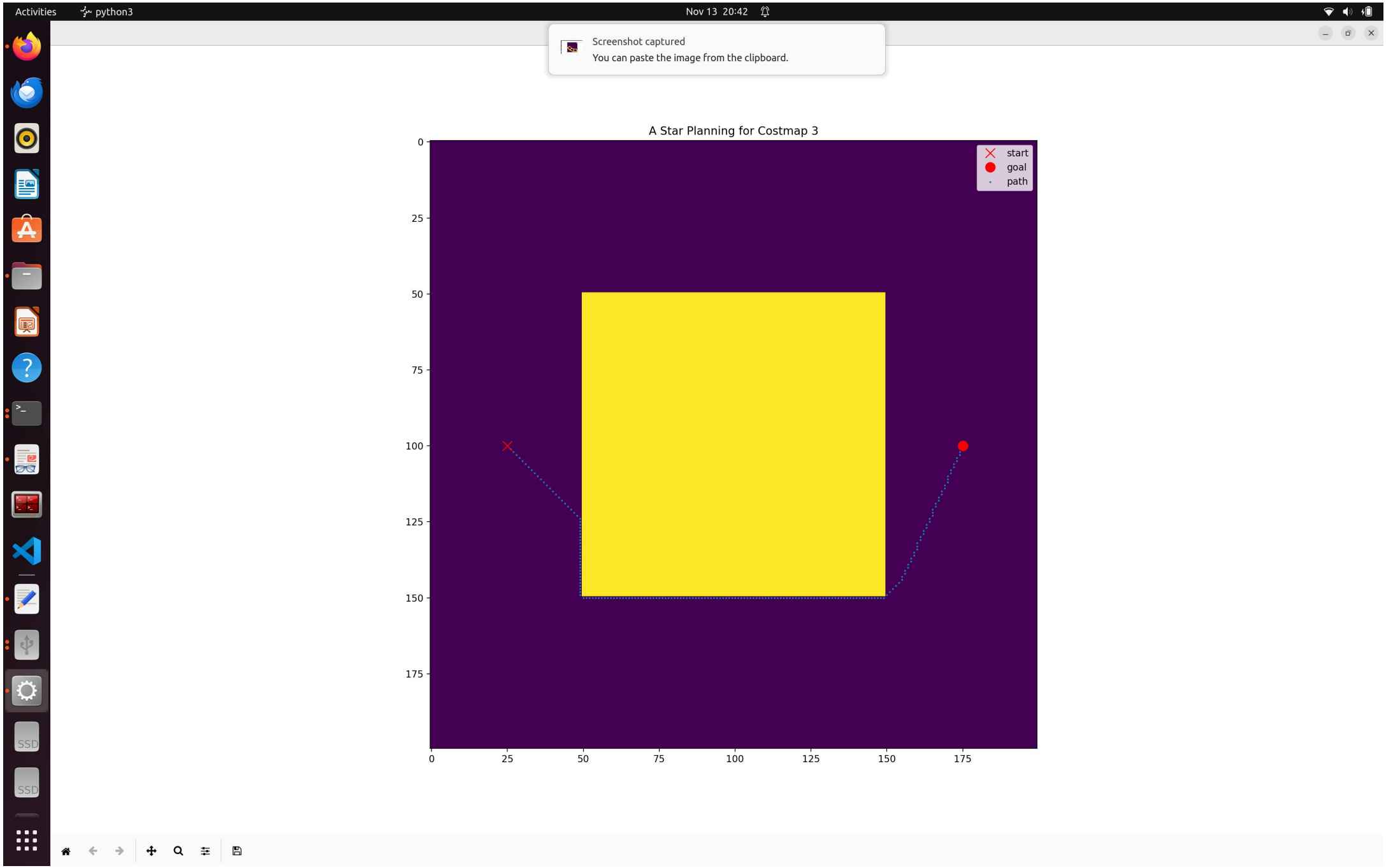
Console - All

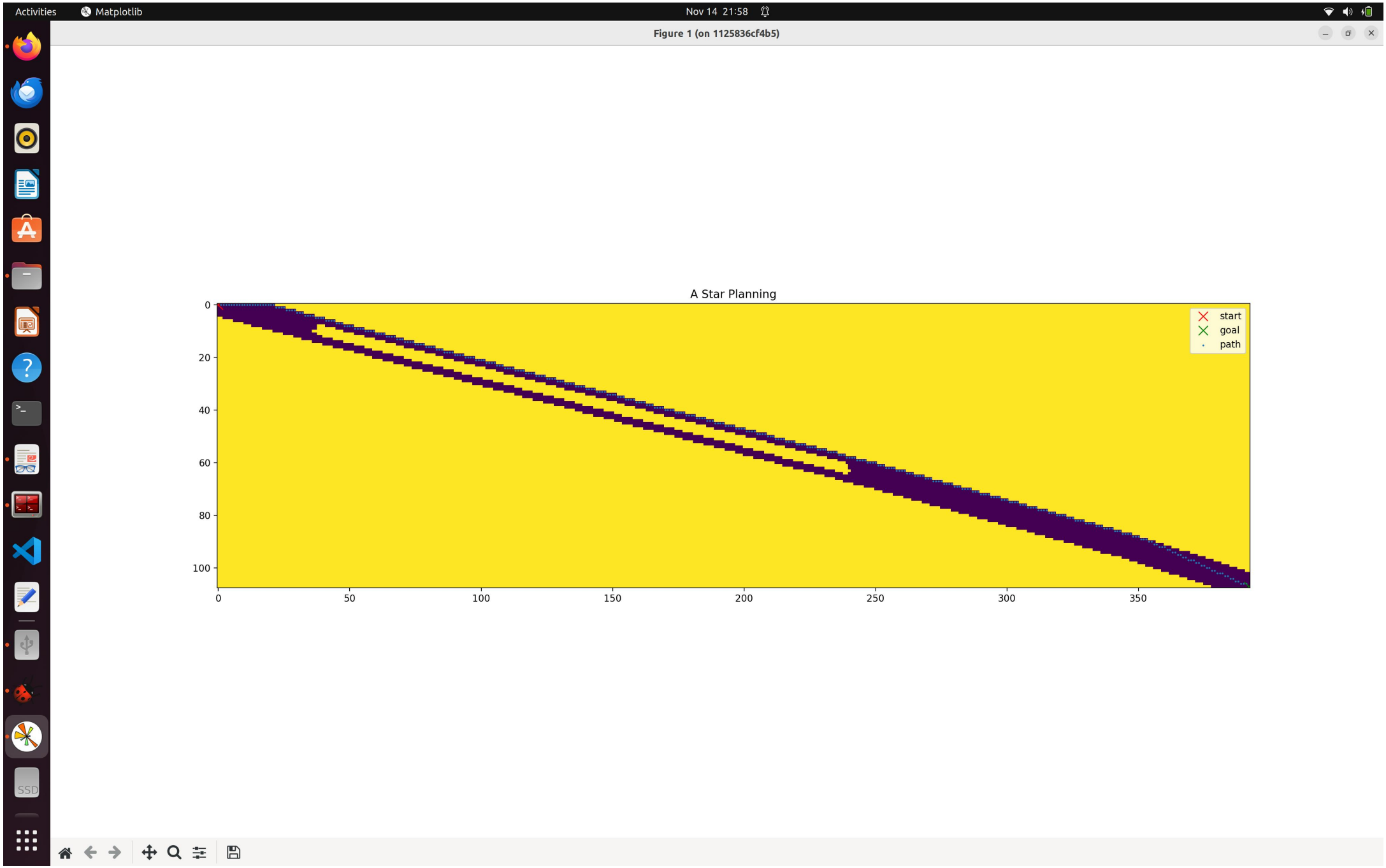
```
Your time is  118.49600000000001
Your total score is : 100.0/100.0
total steps:  118496
maxMinDist:  6.768355886049102
```


Activities
Matplotlib
Nov 13 20:35
/root/Downloads/P3_student/P3_student/P3-LQR/worlds/automotive_new.wbt (P3-LQR) - Webots R2021b (on 1125836cf4b5)
File Edit View Simulation Build Overlays Tools Wizards Help
Simulation View
/root/Downloads/P3_student/P3_student/P3-LQR/controllers/main/your_controller.py
Figure 1 (on 1125836cf4b5)
Y (m)
X (m)
delta (rad)
Time (s)
xdot (m/s)
ydot (m/s)
psi (rad)
psidot (rad/s)
minDist (m)
F (N)
Console - All
DEPRECATION: Robot.getDisplay is deprecated, please use Robot.getDevice instead.
DEPRECATION: Robot.getDisplay is deprecated, please use Robot.getDevice instead.


Activities
Nov 13 20:41
Figure 1
A Star Planning for Costmap 1
start
goal
path
0
500
1000
1500
2000
0
250
500
750
1000
1250
1500
1750


Activities
python3
Nov 13 20:42
Figure 1
A Star Planning for Costmap 2
start
goal
path
0
50
100
150
200
250
300
350
400
0
100
200
300
400


Activities
python3
Nov 13 20:42
Screenshot captured
You can paste the image from the clipboard.
A Star Planning for Costmap 3
start
goal
path
0
25
50
75
100
125
150
175


Activities
Matplotlib
Nov 14 21:58
Figure 1 (on 1125836cf4b5)
A Star Planning
start
goal
path
0
20
40
60
80
100
0
50
100
150
200
250
300
350

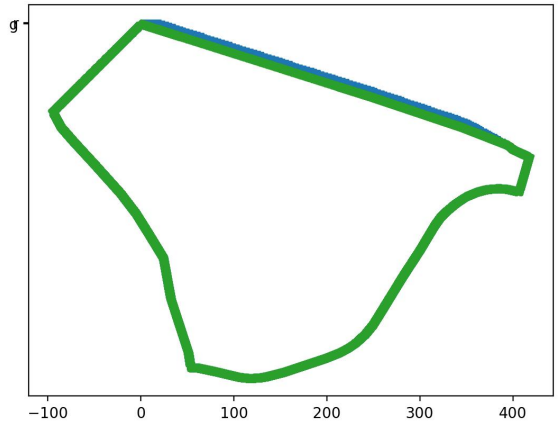
Activities Matplotlib Nov 14 21:59

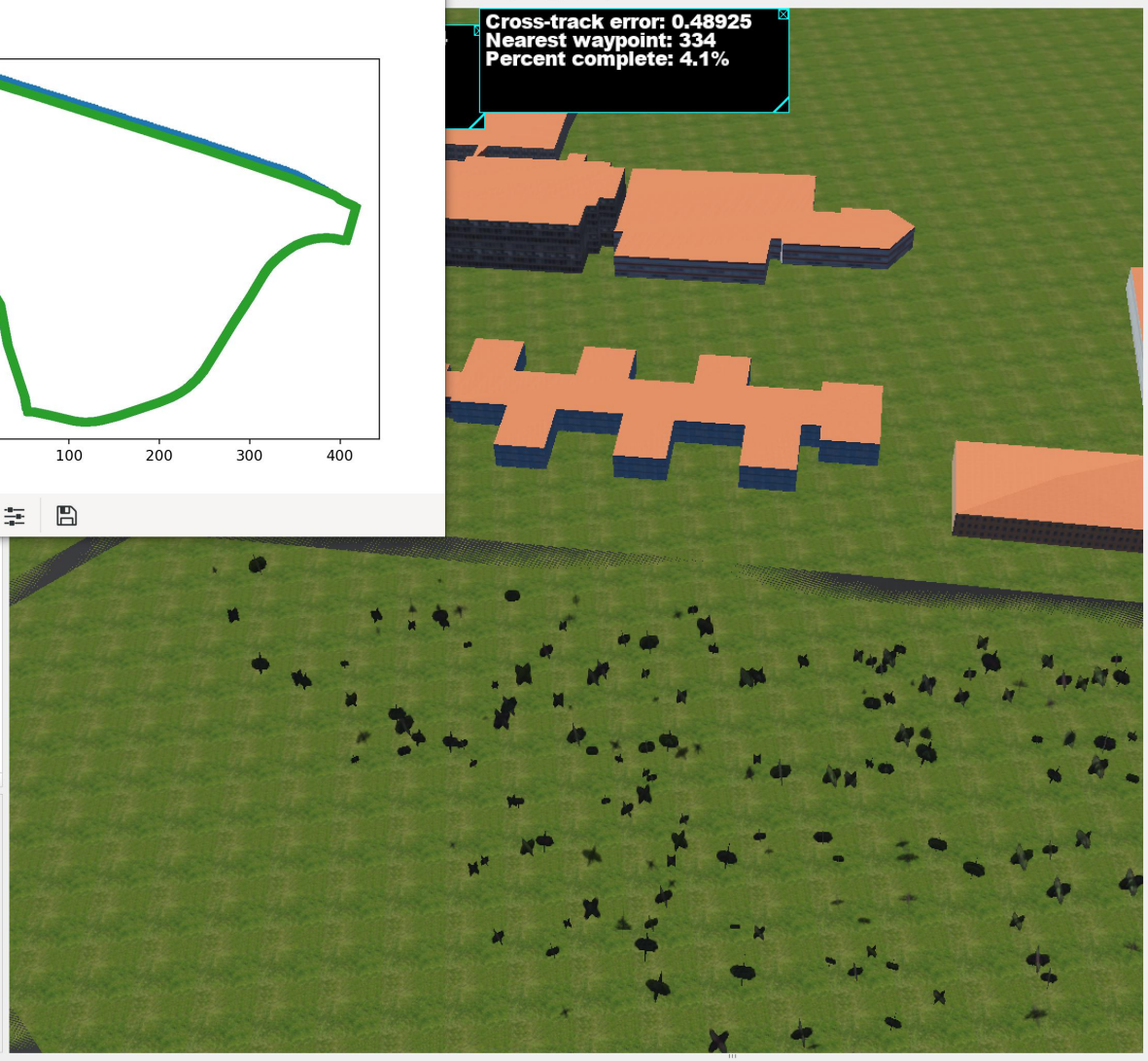
/root/Downloads/P3_student/P3_student/P3-AStar/worlds/Project3.wbt (P3-AStar) - Webots R2021b (on 1125836cf4b5)

File Edit View Simulation Build Overlays Tools Wizards Help

Simulation View

Figure 1 (on 1125836cf4b5)





...ownloads/P3_student/P3_student/P3-AStar/controllers/main/your_controller_lqr_astar.py

your_controller_lqr_astar.py

```python
# Fill in the respective functions to implement the controller

# Import libraries
import numpy as np
from base_controller import BaseController
from scipy import signal, linalg
from scipy.signal import StateSpace, lsim, dlsim
from util import *

# CustomController class (inherits from BaseController)
class CustomController(BaseController):

    def __init__(self, trajectory):

        super().__init__(trajectory)

        # Define constants
        # These can be ignored in P1
        self.lr = 1.39
        self.lf = 1.55
        self.Ca = 20000
        self.Iz = 25854
        self.m = 1888.6
        self.g = 9.81
        self.integralPsiError = 0
        self.previousPsiError = 0
        self.previousXdotError = 0

        # Add additional member variables according to your need here.

    def dlqr(self,A,B,Q,R):
        S = np.matrix(linalg.solve_discrete_are(A, B, Q, R))
        K = -np.matrix(linalg.inv(B.T@S@B+R)@(B.T@S@A))
        return K
    def update(self, timestep):

        trajectory = self.trajectory

        lr = self.lr
        lf = self.lf
        Ca = self.Ca
        Iz = self.Iz
        m = self.m
        g = self.g

        # Fetch the states from the BaseController method
        delT, X, Y, xdot, ydot, psi, psidot,xobs,yobs = super().getStates(

        # Design your controllers in the spaces below.
        A = np.array([[0, 1, 0, 0], [0, -4*Ca / (m * xdot), 4*Ca / m, -(2*
        B = np.array([[0], [2*Ca / m], [0], [(2 * Ca* lf) / Iz]])
        C = np.eye(4)
        D = np.zeros((4,1))

        # Find the closest node to the vehicle
        _, node = closestNode(X, Y, trajectory)
# Choose a node that is ahead of our current node based on index
        forwardIndex = 100


        # Remember, your controllers will need to use the states
        # to calculate control inputs (F, delta).

        # ---------------|Lateral Controller|-----------------------
```

Console - All

```
DEPRECATION: Robot.getDisplay is deprecated, please use Robot.getDevice instead.
DEPRECATION: Robot.getDisplay is deprecated, please use Robot.getDevice instead.
DEPRECATION: Robot.getDisplay is deprecated, please use Robot.getDevice instead.
```

Activities Webots Nov 14 22:10

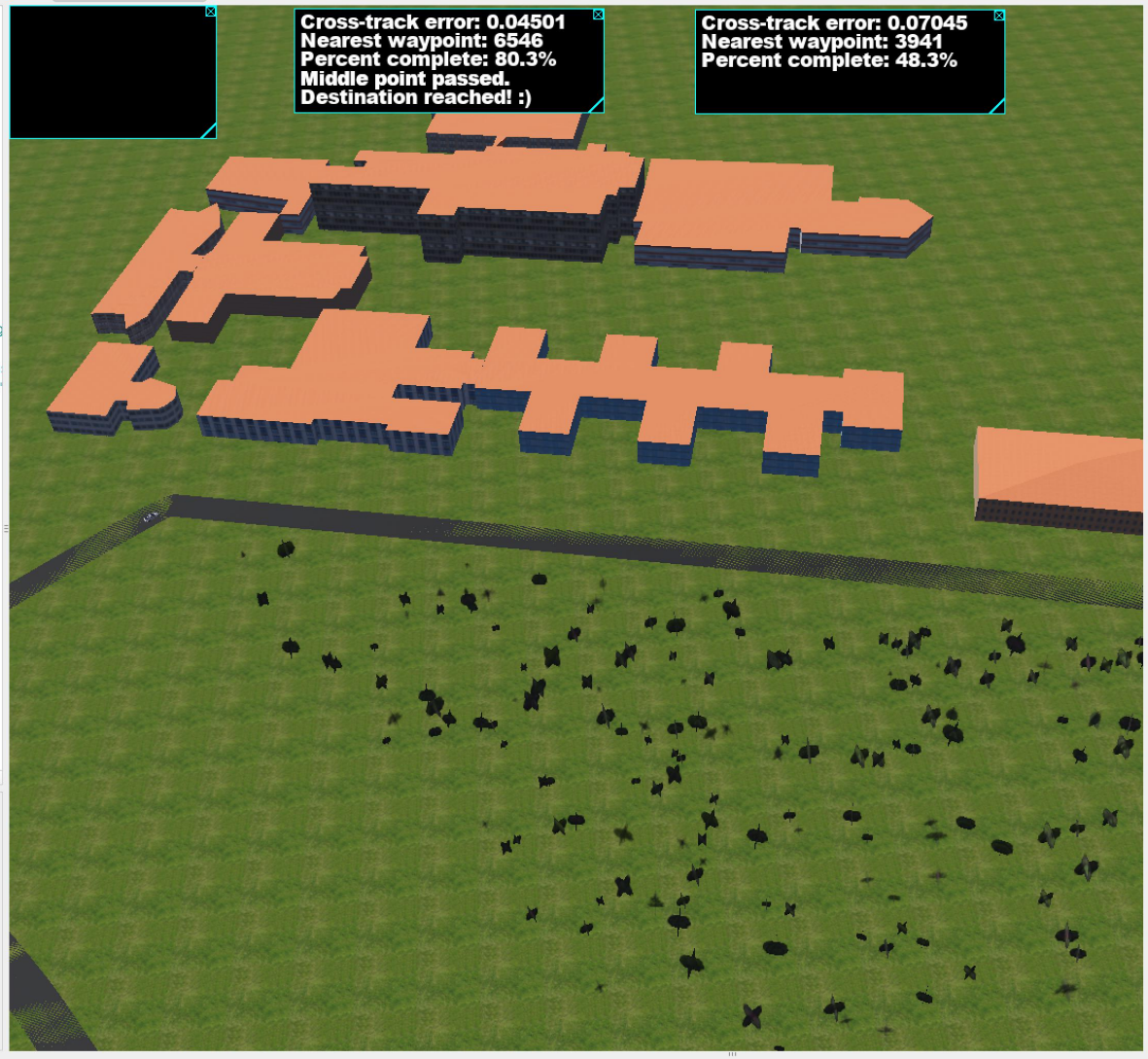
/root/Downloads/P3_student/P3_student/P3-AStar/worlds/Project3.wbt (P3-AStar) - Webots R2021b (on 1125836cf4b5)

File Edit View Simulation Build Overlays Tools Wizards Help

Simulation View

0:01:59:968 - 0.00x

- WorldInfo
- Viewpoint
- TexturedBackground
- TexturedBackgroundLight
- DEF TESLA Robot
- DEF OBSTACLE Robot
- Floor "floor"
- Road "road"
- Slide "slide"
- Swing "swing"
- Forest
- SimpleBuilding "building(2)"
- SimpleBuilding "Newell-Simon Hall"
- SimpleBuilding "Scott Hall"
- SimpleBuilding "Doherty Hall"
- SimpleBuilding "Wean Hall"
- SimpleBuilding "Roberts Engineering
- SimpleBuilding "Hamerschlag Hall"
- SimpleBuilding "Tepper School of Bu
- SimpleBuilding "College of Fine Arts"
- SimpleBuilding "Hunt Library"
- SimpleBuilding "Baker Hall"
- SimpleBuilding "Porter Hall"
- SimpleBuilding "Scaife Hall"



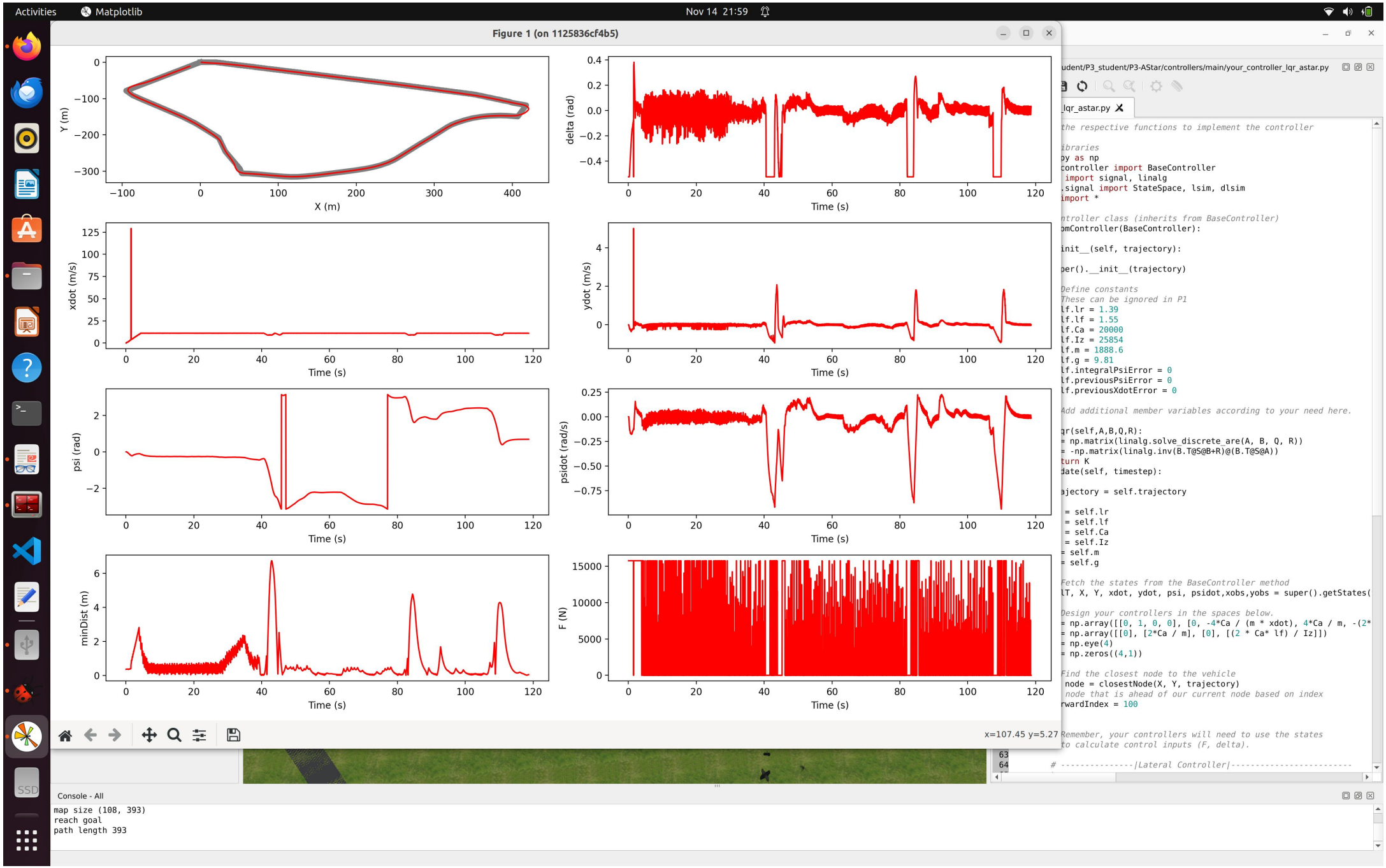
...ownloads/P3_student/P3_student/P3-AStar/controllers/main/your_controller_lqr_astar.py

your_controller_lqr_astar.py

```python
# Fill in the respective functions to implement the controller

# Import libraries
import numpy as np
from base_controller import BaseController
from scipy import signal, linalg
from scipy.signal import StateSpace, lsim, dlsim
from util import *

# CustomController class (inherits from BaseController)
class CustomController(BaseController):

    def __init__(self, trajectory):

        super().__init__(trajectory)

        # Define constants
        # These can be ignored in P1
        self.lr = 1.39
        self.lf = 1.55
        self.Ca = 20000
        self.Iz = 25854
        self.m = 1888.6
        self.g = 9.81
        self.integralPsiError = 0
        self.previousPsiError = 0
        self.previousXdotError = 0

        # Add additional member variables according to your need here.

    def dlqr(self,A,B,Q,R):
        S = np.matrix(linalg.solve_discrete_are(A, B, Q, R))
        K = -np.matrix(linalg.inv(B.T@S@B+R)@(B.T@S@A))
        return K
    def update(self, timestep):

        trajectory = self.trajectory

        lr = self.lr
        lf = self.lf
        Ca = self.Ca
        Iz = self.Iz
        m = self.m
        g = self.g

        # Fetch the states from the BaseController method
        delT, X, Y, xdot, ydot, psi, psidot,xobs,yobs = super().getStates(

        # Design your controllers in the spaces below.
        A = np.array([[0, 1, 0, 0], [0, -4*Ca / (m * xdot), 4*Ca / m, -(2*
        B = np.array([[0], [2*Ca / m], [0], [(2 * Ca* lf) / Iz]])
        C = np.eye(4)
        D = np.zeros((4,1))

        # Find the closest node to the vehicle
        _, node = closestNode(X, Y, trajectory)
# Choose a node that is ahead of our current node based on index
        forwardIndex = 100


        # Remember, your controllers will need to use the states
        # to calculate control inputs (F, delta).

        # ---------------|Lateral Controller|-----------------------
```

Console - All

```
reach goal
path length 393
total steps:  118656
maxMinDist:  6.731735388164265
```


Activities
Matplotlib
Nov 14 21:59
Figure 1 (on 1125836cf4b5)
Y (m)
X (m)
delta (rad)
Time (s)
xdot (m/s)
ydot (m/s)
psi (rad)
psidot (rad/s)
minDist (m)
F (N)
x=107.45 y=5.27
udent/P3_student/P3-AStar/controllers/main/your_controller_lqr_astar.py
_lqr_astar.py
the respective functions to implement the controller
ibraries
py as np
controller import BaseController
import signal, linalg
.signal import StateSpace, lsim, dlsim
import *
ntroller class (inherits from BaseController)
omController(BaseController):
init__(self, trajectory):
per().__init__(trajectory)
Define constants
These can be ignored in P1
lf.lr = 1.39
lf.lf = 1.55
lf.Ca = 20000
lf.Iz = 25854
lf.m = 1888.6
lf.g = 9.81
lf.integralPsiError = 0
lf.previousPsiError = 0
lf.previousXdotError = 0
Add additional member variables according to your need here.
qr(self,A,B,Q,R):
= np.matrix(linalg.solve_discrete_are(A, B, Q, R))
= -np.matrix(linalg.inv(B.T@S@B+R)@(B.T@S@A))
turn K
date(self, timestep):
ajectory = self.trajectory
= self.lr
= self.lf
= self.Ca
= self.Iz
= self.m
= self.g
Fetch the states from the BaseController method
lT, X, Y, xdot, ydot, psi, psidot,xobs,yobs = super().getStates(
Design your controllers in the spaces below.
= np.array([[0, 1, 0, 0], [0, -4*Ca / (m * xdot), 4*Ca / m, -(2*
= np.array([[0], [2*Ca / m], [0], [(2 * Ca* lf) / Iz]])
= np.eye(4)
= np.zeros((4,1))
Find the closest node to the vehicle
node = closestNode(X, Y, trajectory)
node that is ahead of our current node based on index
rwardIndex = 100
Remember, your controllers will need to use the states
to calculate control inputs (F, delta).
----------|Lateral Controller|----------
Console - All
map size (108, 393)
reach goal
path length 393