Final Project

Yengkong Sayaovong

Information Technology, Arizona State University

IFT 360: Applications in AI

Durgesh Sharma

Due Date: December 5, 2024

Final Project

Objective:

The primary goal of this project is to develop a simulation of an autonomous agent navigating a grid-based environment using a Finite State Machine (FSM). The agent aims to reach a predefined destination while avoiding obstacles detected using a simulated LiDAR system. This project demonstrates how FSM logic can dynamically control an agent's decision-making process in a simple grid-world scenario.

Key Features:

1. Finite State Machine: Implements states such as Start, Forward, Turn, Avoid Obstacle, Slow Down, Stop, and Stop-End Trip. The FSM transitions between states based on environmental inputs.
2. Obstacle Detection and Avoidance: Simulates obstacle positions on the grid using random placement and LiDAR-like detection. The agent adjusts its path dynamically to avoid collisions.
3. Grid Navigation: The environment is represented as a 10x10 grid, with the agent starting at [0, 0] and the destination set at [9, 9]. Movement is restricted to grid boundaries.
4. Dynamic Input Simulation: Simulates environmental factors like GPS commands and obstacle presence to test the agent's adaptability.
5. Termination Criteria: The program ends when the agent either reaches the destination or exceeds a set number of steps.

Dataset:

This project does not use external datasets. Instead, it generates data dynamically during runtime:

- Grid Environment: A 10x10 grid with randomly placed obstacles.

- Obstacle Positions: Generated using a random number generator.
- Agent Movements: Determined by FSM logic and simulated percepts like LiDAR readings and GPS commands.

Implementation:

The project uses Python for the simulation, leveraging functions to handle state transitions, agent movement, obstacle detection, and LiDAR simulation. The core FSM logic is implemented in a while loop, ensuring state transitions occur based on environmental inputs.

Expected Outcomes:

1. The agent successfully navigates the grid, avoiding obstacles and reaching the destination.
2. The program demonstrates dynamic decision-making by transitioning through appropriate states.
3. Clear results are shown at each step, including state changes, agent movements, and final outcomes.

Significance:

This project highlights the practical application of FSMs in robotics and artificial intelligence, providing a foundation for real-world implementations such as autonomous vehicle navigation and robotics control systems.

Code:

```
import random


# Define All States
states = {"Start", "Stop", "Slow Down", "Forward", "Turn",  "Stop-End Trip", "Avoid Obstacle"}
```

```
# Initialize Percepts
light = {'red': 0, 'green': 0, 'yellow': 0}
dist_to_light = {'near': 0, 'far': 0}
destination_reached = {'y': 0, 'n': 0}
front_car = {'near': 0, 'far': 0, 'stopped': 0}
gps_command = {'right': 0, 'left': 0, 'u-turn': 0, 'none': 0}
intersection = {'yes': 0, 'no': 0}

# LiDAR readings (simulated grid)
grid_size = 10
obstacle_positions = set()
agent_position = [0, 0]
destination = [9, 9]

# State Initialization
state = 'Start'
max_steps = 1000  # Maximum steps before stopping
current_step = 0

print('Initial State =', state)

# Helper Functions
```

```
def simulate_lidar():
    """Simulates LiDAR readings in a grid environment."""
    global obstacle_positions
    obstacle_positions = {(random.randint(0, grid_size - 1), random.randint(0, grid_size - 1)) for _
in range(5)}
    print("LiDAR detected obstacles at:", obstacle_positions)


def move_agent(action):
    """Moves the agent in the grid."""
    if action == "Forward":
        agent_position[1] = min(agent_position[1] + 1, grid_size - 1)  # Move up
    elif action == "Turn Right":
        agent_position[0] = min(agent_position[0] + 1, grid_size - 1)  # Move right
    elif action == "Turn Left":
        agent_position[0] = max(agent_position[0] - 1, 0)  # Move left
    elif action == "U-Turn":
        agent_position[1] = max(agent_position[1] - 1, 0)  # Move down
    print("Agent moved to:", agent_position)


def check_obstacle():
    """Checks if there's an obstacle in the agent's path."""
    return tuple(agent_position) in obstacle_positions
```

```
# Start Navigation
while current_step < max_steps:
    current_step += 1  # Increment step counter
    print(f"\nStep: {current_step}")

    # Simulate environment changes
    simulate_lidar()
    obstacle_nearby = check_obstacle()
    destination_reached_flag = agent_position == destination

    # FSM Logic
    print("==================================")
    if state == 'Start':
        if obstacle_nearby:
            state = 'Avoid Obstacle'
        else:
            state = 'Forward'
    elif state == 'Stop':
        if destination_reached_flag:
            state = 'Stop-End Trip'
        elif not obstacle_nearby:
            state = 'Forward'
    elif state == 'Slow Down':
```

```
    if destination_reached_flag:
        state = 'Stop-End Trip'
    elif obstacle_nearby:
        state = 'Avoid Obstacle'
    else:
        state = 'Forward'
elif state == 'Forward':
    if destination_reached_flag:
        state = 'Stop-End Trip'
    elif obstacle_nearby:
        state = 'Avoid Obstacle'
    elif random.choice([True, False]):  # Simulate intersections
        state = 'Turn'
    else:
        move_agent("Forward")
elif state == 'Turn':
    gps_action = random.choice(['right', 'left', 'u-turn'])
    if gps_action == 'right':
        print("Turning Right...")
        move_agent("Turn Right")
    elif gps_action == 'left':
        print("Turning Left...")
        move_agent("Turn Left")
```

```
        elif gps_action == 'u-turn':
            print("Making a U-Turn...")
            move_agent("U-Turn")
        state = 'Forward'
    elif state == 'Avoid Obstacle':
        print("Obstacle detected! Calculating new path...")
        # Implement basic avoidance (e.g., move around the obstacle)
        move_agent(random.choice(["Turn Left", "Turn Right"]))
        state = 'Forward'
    elif state == 'Stop-End Trip':
        print('Destination Reached! Trip Ended!')
        break
    else:
        print("ERROR: Unknown State!!!")
        break

    # Adding state output
    print(f"State changed to: {state}")

# Check why the loop ended
if current_step >= max_steps:
    print("Maximum steps reached. Stopping the program.")
elif state == 'Stop-End Trip':
```

print("Program terminated successfully after reaching the destination.")+

Screenshot:

```python
import random

# Define All States
states = {"Start", "Stop", "Slow Down", "Forward", "Turn", "Stop-End Trip", "Avoid Obstacle"}
```

```python
# Initialize Percepts
light = {'red': 0, 'green': 0, 'yellow': 0}
dist_to_light = {'near': 0, 'far': 0}
destination_reached = {'y': 0, 'n': 0}
front_car = {'near': 0, 'far': 0, 'stopped': 0}
gps_command = {'right': 0, 'left': 0, 'u-turn': 0, 'none': 0}
intersection = {'yes': 0, 'no': 0}
```

```python
# LiDAR readings (simulated grid)
grid_size = 10
obstacle_positions = set()
agent_position = [0, 0]
destination = [9, 9]
```

```python
# State Initialization
state = 'Start'
max_steps = 1000  # Maximum steps before stopping
current_step = 0
```

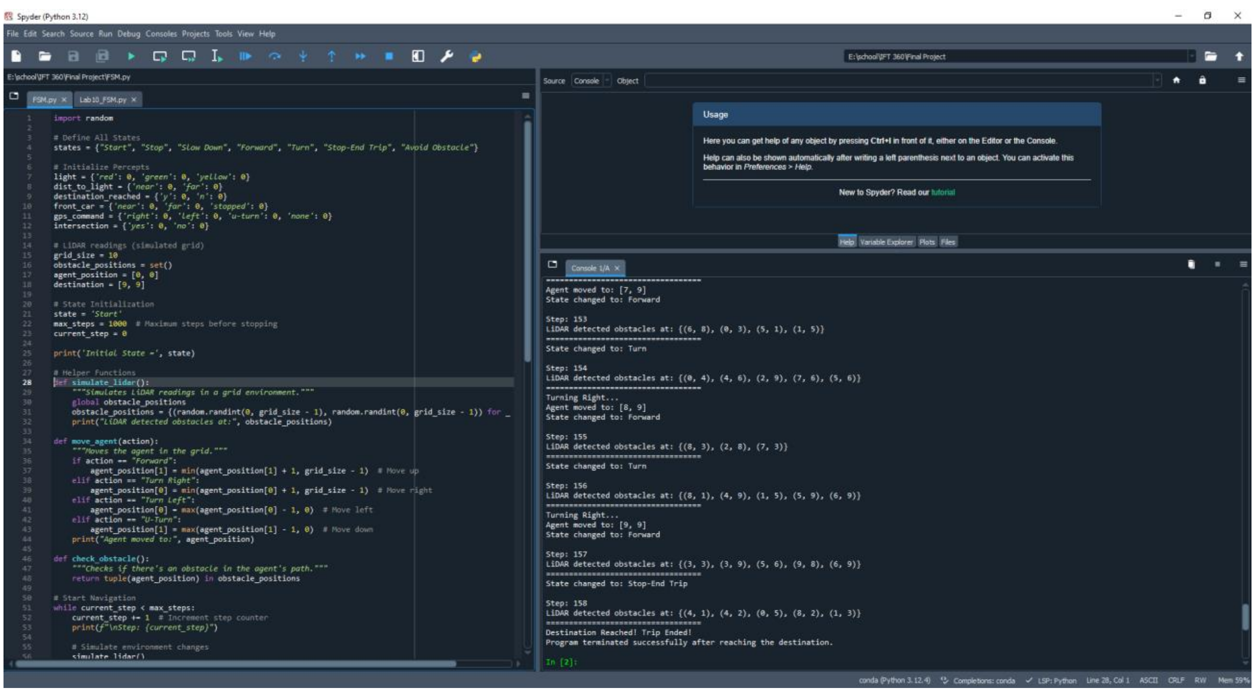
```python
if state == 'Start':
    if obstacle_nearby:
        state = 'Avoid Obstacle'
    else:
        state = 'Forward'
elif state == 'Stop':
    if destination_reached_flag:
        state = 'Stop-End Trip'
    elif not obstacle_nearby:
        state = 'Forward'
```

```python
elif state == 'Avoid Obstacle':
    print("Obstacle detected! Calculating new path...")
    # Implement basic avoidance (e.g., move around the obstacle)
    move_agent(random.choice(["Turn Left", "Turn Right"]))
    state = 'Forward'
elif state == 'Stop-End Trip':
```

```python
elif state == 'Turn':
    gps_action = random.choice(['right', 'left', 'u-turn'])
    if gps_action == 'right':
        print("Turning Right...")
        move_agent("Turn Right")
    elif gps_action == 'left':
        print("Turning Left...")
        move_agent("Turn Left")
    elif gps_action == 'u-turn':
        print("Making a U-Turn...")
        move_agent("U-Turn")
    state = 'Forward'
```

```python
elif state == 'Avoid Obstacle':
    print("Obstacle detected! Calculating new path...")
    # Implement basic avoidance (e.g., move around the obstacle)
    move_agent(random.choice(["Turn Left", "Turn Right"]))
    state = 'Forward'
elif state == 'Stop-End Trip':
    print('Destination Reached! Trip Ended!')
    break
else:
    print("ERROR: Unknown State!!!")
    break
```

Result:



Link to Video:
https://drive.google.com/file/d/1goVuKGyhltWsjMmmE2xwwnWjNPYxne7P/view?usp=sharing

References

Anchor Rainbow. (2013, October 13). *Let's Learn Python #19 - Finite-State Machines (FSM)*. YouTube. https://www.youtube.com/watch?v=E45v2dD3IQU

Tech With Tim. (2020). A* Pathfinding Visualization Tutorial - Python A* Path Finding Tutorial [YouTube Video]. In *YouTube*. https://www.youtube.com/watch?v=JtiK0DOeI4A