PROIECT SINCRETIC

Echipa: MazeMasters
Autori: Ardelean Ioana,
Bucurici Ruth,
Goagără Bianca.

Cuprins

1. Introducere

Proiectul MazeMaster are ca scop dezvoltarea unei soluții autonome utilizând platforma ROS2, modelul de robot mobil TurtleBot3 Waffle Pi și limbajul de programare Python. Robotul navighează printr-un labirint până identifică un obiect de culoare roșie, la care se oprește automat. Acest proiect reprezintă o aplicație practicată de robotică, vizualizare computerizată și inteligență artificială.

Navigarea autonomă și detecția obiectelor reprezintă componente cheie ale roboților moderni. Utilizând tehnologii precum ROS2, Gazebo și OpenCV, proiectul demonstrează cum un robot mobil poate combina navigarea cu procesarea imaginilor pentru a rezolva o sarcină specifică.

2. Descriere generală a proiectului

1. Tehnologii utilizate

- ROS2 (Humble): Pentru gestionarea comunicării între modulele robotului (publicatori, subscriberi).
- TurtleBot3 Waffle Pi: Robot mobil utilizat ca platformă virtuală.
- Gazebo: Simularea mediului de testare.
- OpenCV: Procesare de imagine pentru detectarea obiectelor de culoare roșie.
- Python: Limbaj de programare principal pentru implementare.

2. Arhitectura generală

Robotul folosește trei subsisteme principale:

- Navigare autonomă: Bazată pe datele primite de la senzorul LaserScan.
- Procesare imagine: Utilizarea camerei pentru identificarea obiectelor de culoare roșie.
- Controlul mișcării: Calcularea și publicarea comenzilor de mișcare prin topicul /cmd_vel.

3. Funcționalități cheie

1. Navigarea autonomă

Robotul utilizează informațiile de la senzorul LaserScan pentru a evita obstacolele din labirint. Algoritmul calculează distanțele față de pereți și ajustează direcția pentru a asigura o deplasare sigură.

2. Detectarea obiectului roșu

Robotul prelucrează imaginile capturate de cameră utilizând biblioteca OpenCV. Culoarea roșie este identificată folosind spațiul de culoare HSV și o combinație de filtre.

3. Oprirea automată

La detectarea obiectului roșu, robotul se oprește și nu mai navighează prin labirint. Această funcționalitate este gestionată printr-o variabilă de control red_detected.

4. Implementarea tehnică

1. Structura codului

Codul este organizat într-o singură clasă, MazeSolver, care:

- Publică comenzile de mișcare către topicul /cmd_vel.
- Primește informații de la senzorii LaserScan și cameră.
- Procesează imaginile pentru identificarea obiectelor roșii.

2. Explicații ale secțiunilor din cod

- Inițializarea nodului

```
class MazeSolver(Node):
    def _init_(self):
        super()._init_('maze_solver')

        # Publisher for robot velocity
        self.cmd_vel_publisher = self.create_publisher(Twist, '/cmd_vel', 10)

        # Subscriber for LaserScan
        self.laser_subscriber = self.create_subscription(LaserScan, '/scan', self.laser_callback, 10)

        # Subscriber for camera feed
        self.camera_subscriber = self.create_subscription(Image, '/camera/image_raw', self.camera_callback, 10)

        # Movement control variables
        self.twist = Twist()
        self.red_detected = False

        # OpenCV Bridge
        self.bridge = CvBridge()

        # Red color range in HSV
        self.red_lower_bound = np.array([0, 120, 70])
        self.red_upper_bound = np.array([10, 255, 255])
        self.red_lower_bound_2 = np.array([170, 120, 70])
        self.red_upper_bound_2 = np.array([180, 255, 255])
```

Explicație: Această secțiune inițializează nodul ROS2, definește publicatorii și subscriberii, și setează variabilele necesare pentru procesarea imaginilor și mișcarea robotului. Intervalele HSV sunt utilizate pentru detectarea culorii roșii.

- Callback pentru senzorul LaserScan

```
def laser_callback(self, msg):
    if self.red_detected:
        return  # Stop maze navigation if red object is detected

    # Process laser data for obstacle avoidance
    left = min(min(msg.ranges[60:120]), 10.0)  # Left side
    front = min(min(msg.ranges[0:30] + msg.ranges[330:360]), 10.0)  # Front
    right = min(min(msg.ranges[240:300]), 10.0)  # Right side

    safe_distance = 0.5  # Distance to avoid obstacles

    if front < safe_distance:
        # Turn to avoid obstacle
        if left > right:
            self.twist.linear.x = 0.0
            self.twist.angular.z = 0.5  # Turn left
        else:
            self.twist.linear.x = 0.0
            self.twist.angular.z = -0.5  # Turn right
    else:
        # Move forward if the path is clear
        self.twist.linear.x = 0.3
        self.twist.angular.z = 0.0

    # Publish velocity command
    self.cmd_vel_publisher.publish(self.twist)
```

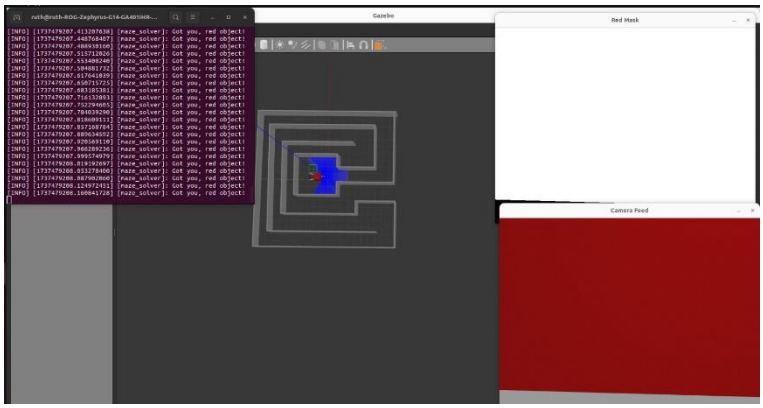
Explicație: Algoritmul analizează datele de la senzorul LaserScan pentru a detecta obstacolele. Distanțele din față și din lateral sunt comparate, iar robotul virează în direcția mai sigură sau merge înainte dacă drumul este liber.

- Detectarea culorii roșii

```python
def camera_callback(self, msg):
    if self.red_detected:
        # Adjust movement to approach the red object
        self.follow_red_object(msg)
        return

    # Convert image to OpenCV format
    frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Create masks for red color
    mask1 = cv2.inRange(hsv_frame, self.red_lower_bound, self.red_upper_bound)
    mask2 = cv2.inRange(hsv_frame, self.red_lower_bound_2, self.red_upper_bound_2)
    mask = cv2.bitwise_or(mask1, mask2)

    # Check if red object is detected
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    if contours:
        self.red_detected = True
        self.get_logger().info("Red object detected! Switching to follow mode.")
        self.follow_red_object(msg)

    # Optionally display the camera feed and mask (for debugging)
    cv2.imshow("Camera Feed", frame)
    cv2.imshow("Red Mask", mask)
    cv2.waitKey(1)
```

Explicație: Funcția prelucrează imaginile din cameră pentru a identifica obiectele de culoare roșie. Masca este creată utilizând intervalele HSV, iar contururile sunt analizate pentru a verifica dacă un obiect roșu este prezent.

3.Rezultat



Explicație: La găsirea obiectului roșu, robotul s-a oprit și s-a afișat în terminal mesajul: “Got you, red object!”.

5. Concluzie

Proiectul demonstrează cum un robot mobil poate combina navigarea autonomă cu procesarea imaginilor pentru a rezolva sarcini complexe. Prin integrarea mai multor module, robotul este capabil să navigheze printr-un mediu necunoscut și să reacționeze la stimulii din mediul său.

6. Bibliografie

- Link proiect: https://github.com/ioana199/MazeMasters
- https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/
- https://github.com/noshluk2/ROS2-Autonomous-Driving-and-Navigation-SLAM-with-TurtleBot3
- https://www.youtube.com/watch?v=S8pwfsK-F9w
- https://chatgpt.com/