**Assumptions and Limitations**

* Assumptions:
  + The warehouse environment is a rectangular grid.
  + Obstacles are detected within 0.2m using sensors.
  + The robot moves with a constant linear velocity (1 m/s) and can rotate with an angular velocity of 3 rad/s.
  + The robot starts from a predefined point and tries to navigate towards a target goal while avoiding obstacles.
* Limitations:
  + No consideration for dynamic obstacles (obstacles are static).
  + Sensors have a limited detection radius (0.2m).
  + We assume a simple 2D plane for the simulation.

**Environment and Robot Setup**

* Robot Wheel Specs:
  + Diameter = 10 cm (0.1 m)
  + Wheel Distance = 30 cm (0.3 m)
* Obstacle Setup:
  + There are 10 obstacles randomly placed on the floor, with at least 1m separation between them.
* Robot Sensors:
  + Simulate LIDAR-like behavior using a simple proximity sensor model that triggers when an obstacle is within 0.2m.

**Algorithm Design: Obstacle Avoidance Logic**

A good choice is the Bug 2 Algorithm:

* Step 1: The robot moves towards the goal directly unless it encounters an obstacle.
* Step 2: If an obstacle is detected within 0.2m, the robot rotates left/right and follows the obstacle boundary.
* Step 3: Once clear, the robot continues towards the goal.

**Code Implementation:**

**Code: Autonomous Robot Navigation using pygame**

First, install the required dependencies

**pip install pygame**

**Then we will create the code:**

**import pygame**

**import math**

**import random**

**pygame.init()**

**WIDTH, HEIGHT = 800, 600**

**screen = pygame.display.set\_mode((WIDTH, HEIGHT))**

**pygame.display.set\_caption("Robot Navigation")**

**ROBOT\_RADIUS = 15**

**OBSTACLE\_RADIUS = 20**

**SENSOR\_RANGE = 40**

**LINEAR\_VELOCITY = 1**

**ANGULAR\_VELOCITY = 3**

**WHITE, BLACK, RED, BLUE = (255, 255, 255), (0, 0, 0), (255, 0, 0), (0, 0, 255)**

**robot\_pos, robot\_angle = [100, 100], 0**

**goal\_pos = [700, 500]**

**obstacles = [(random.randint(50, WIDTH - 50), random.randint(50, HEIGHT - 50)) for \_ in range(10)]**

**def distance(p1, p2):**

**return math.hypot(p2[0] - p1[0], p2[1] - p1[1])**

**def detect\_obstacle(robot\_pos):**

**return next(((obs, obs) for obs in obstacles if distance(robot\_pos, obs) <= SENSOR\_RANGE + OBSTACLE\_RADIUS), (None, None))**

**def rotate(angle, direction):**

**return (angle + (ANGULAR\_VELOCITY if direction == "left" else -ANGULAR\_VELOCITY)) % 360**

**def move\_towards\_goal()**

**global robot\_angle**

**angle\_to\_goal = math.degrees(math.atan2(goal\_pos[1] - robot\_pos[1], goal\_pos[0] - robot\_pos[0]))**

**if abs(robot\_angle - angle\_to\_goal) > 5:**

**robot\_angle = rotate(robot\_angle, "left" if angle\_to\_goal > robot\_angle else "right")**

**else:**

**robot\_pos[0] += LINEAR\_VELOCITY \* math.cos(math.radians(robot\_angle))**

**robot\_pos[1] += LINEAR\_VELOCITY \* math.sin(math.radians(robot\_angle))**

**def avoid\_obstacle(obstacle\_pos):**

**global robot\_angle**

**robot\_pos[0] -= 10 \* math.cos(math.radians(robot\_angle)) # Back off slightly**

**angle\_to\_obstacle = math.degrees(math.atan2(obstacle\_pos[1] - robot\_pos[1], obstacle\_pos[0] - robot\_pos[0]))**

**angle\_diff = (angle\_to\_obstacle - robot\_angle) % 360**

**if 0 < angle\_diff > 180: # Obstacle is in front**

**robot\_angle = rotate(robot\_angle, "right") # Turn right**

**else:**

**robot\_angle = rotate(robot\_angle, "left") # Turn left**

**robot\_pos[0] += LINEAR\_VELOCITY \* math.cos(math.radians(robot\_angle))**

**robot\_pos[1] += LINEAR\_VELOCITY \* math.sin(math.radians(robot\_angle))**

**running = True**

**while running:**

**screen.fill(WHITE)**

**pygame.draw.circle(screen, BLUE, goal\_pos, 10)**

**for obs in obstacles:**

**pygame.draw.circle(screen, RED, obs, OBSTACLE\_RADIUS)**

**pygame.draw.circle(screen, BLACK, (int(robot\_pos[0]), int(robot\_pos[1])), ROBOT\_RADIUS)**

**obstacle\_detected, obstacle\_pos = detect\_obstacle(robot\_pos)**

**if obstacle\_detected:**

**avoid\_obstacle(obstacle\_pos)**

**else:**

**move\_towards\_goal()**

**if distance(robot\_pos, goal\_pos) < 10:**

**print("Goal reached!")**

**running = False**

**pygame.display.flip()**

**pygame.time.delay(50)**

**for event in pygame.event.get():**

**if event.type == pygame.QUIT:**

**running = False**

**pygame.quit()**