**Part A**

**Question 1:** I declare a array named **sensor\_angles** which list the angles from (-40:40) with 10 increment**.** Additionally, I take one same size array to calculate the distances using **getDist** function. I run a loop and calculate the distances for each angle and also add a condition for low/no single.Then the four sweep were taken (1:4) for right side and (6:end) for left side of the robot orientation from **distances** array in the code. Later on, I use average of these four distances for each side to find the turn with existing **fuzAvoidObstacle** file. I attached a file partA.m with proper comments.

**Question2:** My solution with average distance combines the 4 distance sweeps on each side to a single “distance” to input to the fuzzy system is because taking average will helps avoid erratic behavior by providing a stable response from smooth data. For example, if I try to use minimum of 4 distance, the robot reacts to outlier obstacles (tested with minimum value in the current code). For example, if the sensor reads [40, 45, 50, 5], using the minimum would result in an overly reactive turn based on 5, but using the mean would result in 35, leading to a more balanced reaction. Using average will ensure a balanced view of the environment and making the navigation to avoid the obstacle. Other methods like maximum might be ignore nearby obstacles and leading to potential collisions.

Outputs: Attached code filename: partA.m

A screenshot of a computer

Description automatically generated

**Part B**

**Question 1**: I attach a **partB.m** file for the second problem and create a separate fuzzy system named fuzSpeed. I added 3 different obstacle in the code.

* **Membership functions**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

Description automatically generated

* **Rules**

**A screenshot of a computer

Description automatically generated**

**Question 2:** fuzSpeed input into Matlab. I also added the fuzSpeed file.

* **Sharp**

**A screenshot of a computer

Description automatically generated**

* **Moderate**

**A screenshot of a computer

Description automatically generated**

* **Straight**

**A screenshot of a computer

Description automatically generated**

**Question 3:** I added partB.m file matlab code. Other files are unchanged from baseline (geDist.m and moveRobot)

**Question4:** I tested with different speed limit with multiple obstacles. One of the challenges I face to set the input values for member function for the turn. The reason is the turn is the output of the other fuzzy system. Additionally, the attached code file has comment of each addition. The output of part b.

**A black and white image of a rectangular object

Description automatically generated**

**Part C**

I have two scenarios for partC. The file names are **partC\_no\_obstacle.m and partC\_one\_obstacle.m.** The only differences between these two files are adding walls and testing with multiple obstacles. I created the same scenarios with three obstacles (the current code is with one) to experiment, and it worked perfectly.

Next, the problem is to add another robot extended from part B. I add a 128-pixel robot. Furthermore, I need to create the robot 11x11 pixels. So, I modified the robot drawing array index: **(RrCat-5:RrCat+5, RcCat-5:RcCat+5).** I also need to make two fuzzy inference system (FIS) files named **fuzCatchMouse and fuzAvoidCat**. Both robots have two sensors, and the overall scanning range is 360 (-180:180). In part A, I can avoid obstacles around them using the display sweeping using the fuzAvoidObstacle file. fuzCatchMouse will output the cat's turning angle to catch the mouse based on distance and direction towards the mouse. Moreover, fuzAvoidCat will have opposite rules to run away from the cat. From the partB, I controlled the speed. In this problem, I use the same file(fuzSpeed) to control the speed of both the mouse and the cat. The only change I made here was to assign an initial speed=4 because I needed to change the code structure to solve this problem.

Additionally, I create an function file named **getDistanceAngle.m** which is an extention function to calculate the distance and angle for mouse and cat. Additionally, I also use **getDist.m** function file to calculate the distance from obstacle. **partC\_no\_obstacle.m and partC\_one\_obstacle.m has the detailed implementation with comment.**

* **Membership functions (fuzCatchMouse).**

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Rule Design (fuzCatchMouse)**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **A screenshot of a computer

  AI-generated content may be incorrect.Membership functions (fuzAvoidCat)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Rule design (fuzAvoidCat)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**The screenshots above show the membership functions and rules designed for fuzCatchMouse and fuzAvoidCat. From the first glitch, it may look the same. However, the rules are different, actually opposite. One scenario here is that if the cat is on the right side, the mouse will turn to the left side.**

**I basically have two turn angles for both robots. I use weighted of two angles. I chose the weight ratio using a trial-and-error method. The mouse is mostly aware of avoiding obstacles because it doesn’t know the cat will chase them. So, I used 0.1 weight for avoiding the cat and 0.9 for obstacle weight. Multiplying them and the sum will give me the final turn output for a mouse (0.1\*weight\_of\_avoiding\_cat + 0.9\*weight\_of\_avoiding\_obstacle). On the other hand, the cat is more likely to catch a mouse, so the weight to catch the mouse is 0.9, and I set the avoiding obstacle at 0.1. The total formula looks like (0.9\*weight\_of\_catching\_cat + 0.1\*weight\_of\_avoiding\_obstacle). At the end, I also display a message if cat can catch the mouse.**

**Part D**

I tried to remove the rules and try to observe the situation. When I remove rule 1 and rule 2, the cat doesn’t spend much time to catch. However, when I removed rule 3, the cat required longer to catch a mouse. Then I first remove rule1-rule8 and rule21-rule25 because rule21-rule25 is very\_far. I observe that although a cat can catch a mouse, there are certain scenarios when fuzzy doesn’t have any rules to fire, especially after removing 12 rules. Another observation is that after removing all very\_far and very\_close, the cat can catch. However, removing the medium from rule 10 made the cat’s movement weird. The cat can catch a mouse until I removed 17 rules. However, it might be unable to catch the mouse, but it doesn’t conclude the requirement is not satisfied. From my observation, I need to keep three rules (2,3,4) to satisfy the requirement. The reason is that the cat only needs the distance and position of the mouse to chase. I only keep close and medium, but sometimes, the cat can’t judge the position of the mouse. That results in the cat always being a distance from the mouse. I tried to increase the iteration from 1500 to 3000 to catch the catch. I succeeded in 2876 iterations. It only happens when the cat finds the mouse in very\_close.