In this section describes the implementation of obstacle avoidance using fuzzy control.

Requirements

- Avoid simple local obstacles while maintaining course towards the robots short-term goal.

0.0.1 Design

The primary focus of the fuzzy controller is that i does not make sense to avoid an obstacle that is to the left when going straight. To accommodate it makes sense to look at a rectangle with a width equal or greater than the width of the robot. The controller also needs to be able to steer towards a goal, so the controller should also needs to be aware of obstacles in its immediate vicinity to determine whether it is safe to turn towards it goal. The described zones can be seen in figure **??**.

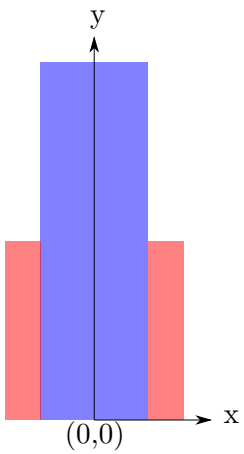

Figure 1: *Zones of interest for obstacle avoidance with coordinate system. Blue is the immediate obstacle avoidance zone. Red is the implicit avoidance zone.*

0.0.2 Implementation

Preprocessed input data The lidar data return polar coordinates. These are transformed into the cartesian coordinates seen in figure **??**.

Fuzzy input variables

- *Distance*
  Distance is the smallest y coordinate in the interval 0m $\leq y \leq$ 1m.
  Distance also implements the following terms:
  - Close
    a ramp function got from (0,1) to (0,1,).
  - Far
    a ramp function going from (0,0) to (1,1).
- *Obstacle*
  Obstacle is the total center of mass of any objects in the immediate avoidance zone (see figure **??**).
  Obstacle makes use of the following terms:

  - Left
    a ramp function got from (0,1) to (0,1,).
  - Right
    a ramp function going from (0,0) to (1,1).

- *ErrorYaw*
  ErrorYaw is the difference between the robots global orientation from the orientation required for the robot to look straight at its goal.
  ErrorYaw makes us of the following terms:

  - Left
    a ramp function going from (0,0) to (3.15,1).
  - Right
    a ramp function going from (-3.15,1) to (0,0).
  - Center
    a triangle function going from (-1,0) to (0,1) to (1,0).

Fuzzy output variables

- *Velocity*
  Velocity is the velocity of the robot.

- *mSteer*
  mSteer is the angular acceleration of the robot.

Rulebase To avoid a weird wiggle while driving along obstacles that intersects with the direct path to goal, 2 rule bases are used.
The rule base seen in listing **??** is responsible for obstacle avoidance and is not active when within 1 meter of the goal.
The rule base seen in listing **??** is responsible for steering towards the goal and it is active when nothing is in the implicit avoidance zone and nothing is in the immediate avoidance zone.
Rule base 1 is active when the robot is not in the vicinity of its goal. Rule base 2 is active when rule base 1 is not or when there is no obstacle in the implicit avoidance zone.

Listing 1: *Rulebase for obstacle avidance*

```
1  rule: if obstacle is left then mSteer is right
2  rule: if obstacle is right then mSteer is left
3  rule: if distance is close then velocity is slow
4  rule: if distance is far then velocity is fast
```

Listing 2: *Rulebase for going towards goal*

```
1  rule: if erroryaw is right then mSteer is left
2  rule: if erroryaw is center then mSteer is straight
3  rule: if erroryaw is center then velocity is fast
4  rule: if erroryaw is not center then velocity is slow
```

0.0.3 Discussion

A weakness of this controller is if there is an obstacle within 1 meter of the short-term goal of the robot as this would obstacle would not be detected and would therefore not be avoided.
An improvement could be to look in the direction of the goal and see if there is an obstacle that needs to be avoid, similar to the bug3 algorithm.