Team Lucid

>>Renegade

The Team

Justin McGettigan	**David Ciccarello**	**Alex Olinger**
Team Lead	Hardware Integration	Programmer
Lead Developer	Programmer	Debugging
	Mathematician	

The Challenge

Wall Detection	Detect the orientation and distance of a wall to the right and left of the car.
Lane Centering	Use the wall detector to determine the steering angle and speed, keeping the car equidistant between walls.
Object Detection / Full Stop	Detect objects in front of the car. / Stop if an object is in front of the car.

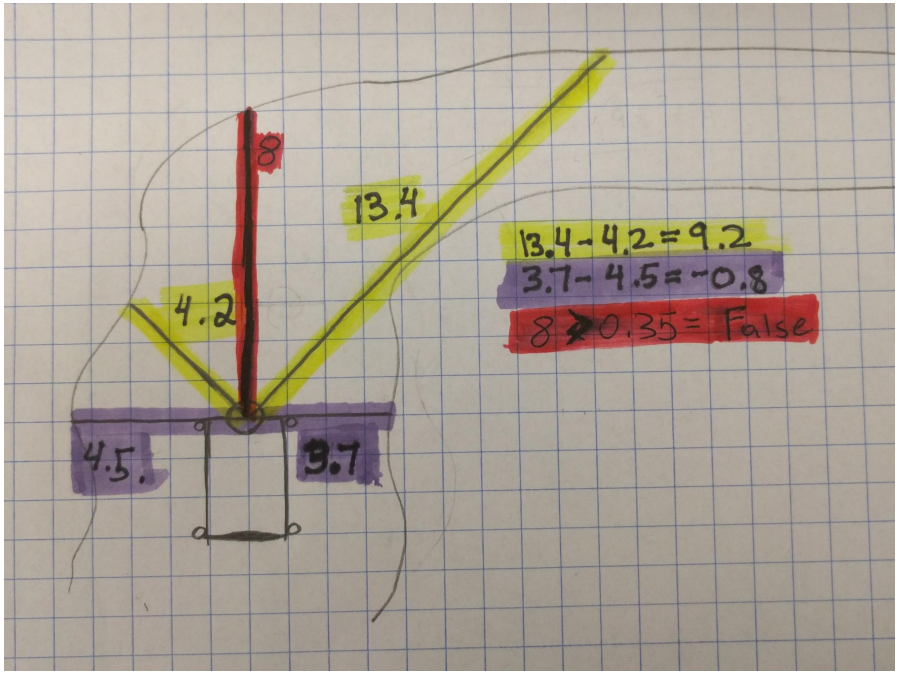
Wall Detection

When determining our steering angle, we take into account:

- **The distance from the wall at 90 and 45 degrees.**
- **The average slope of the walls.**

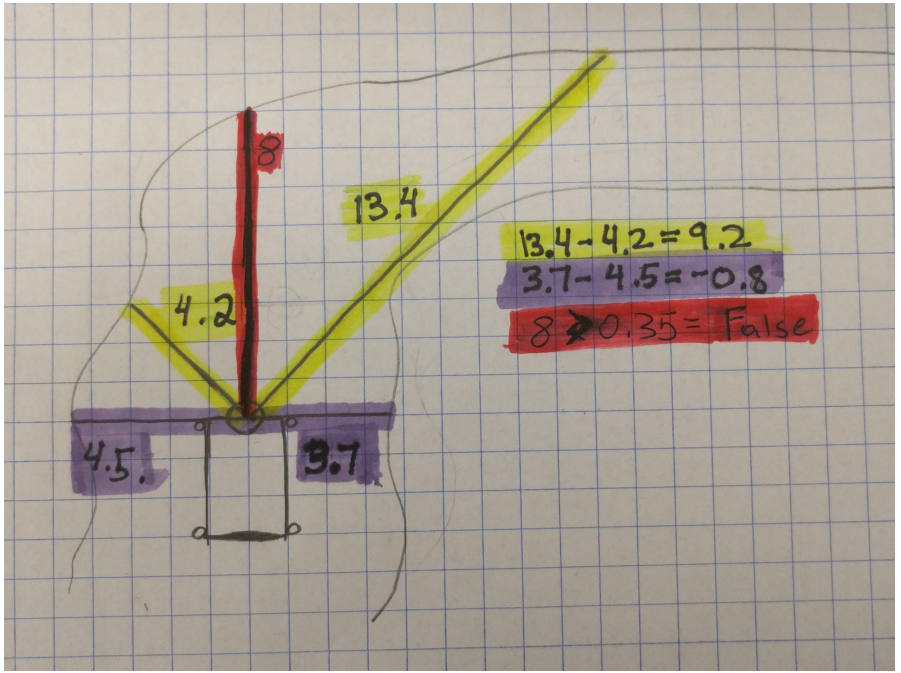
Algorithms

1. Forward Offset
    a. Determine the difference in the distance between -45° and 45° angles
    b. Allows for sight into the future
2. Wall Offset
    a. Determine the difference in the distances between -90° and 90° angles
    b. Enables lane centering



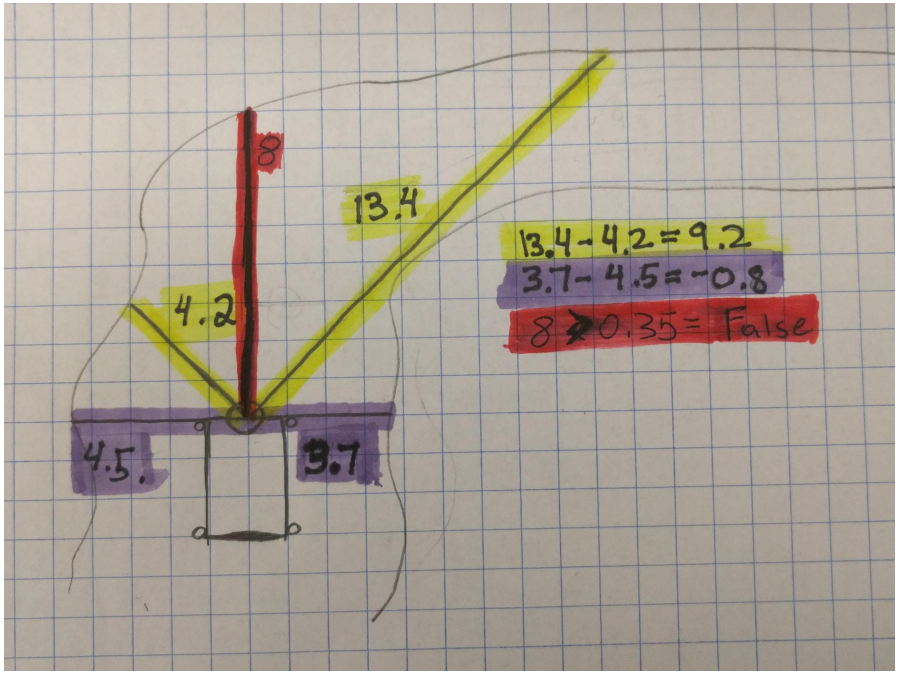
Algorithms

3. Immediate object detection
    a. If object is within 35 cm
        i. Reverse robot
        ii. Reduces chance of getting stuck around a sharp turn or fringe scenario
    b. If reversed for more than 1.5 total seconds. STOP



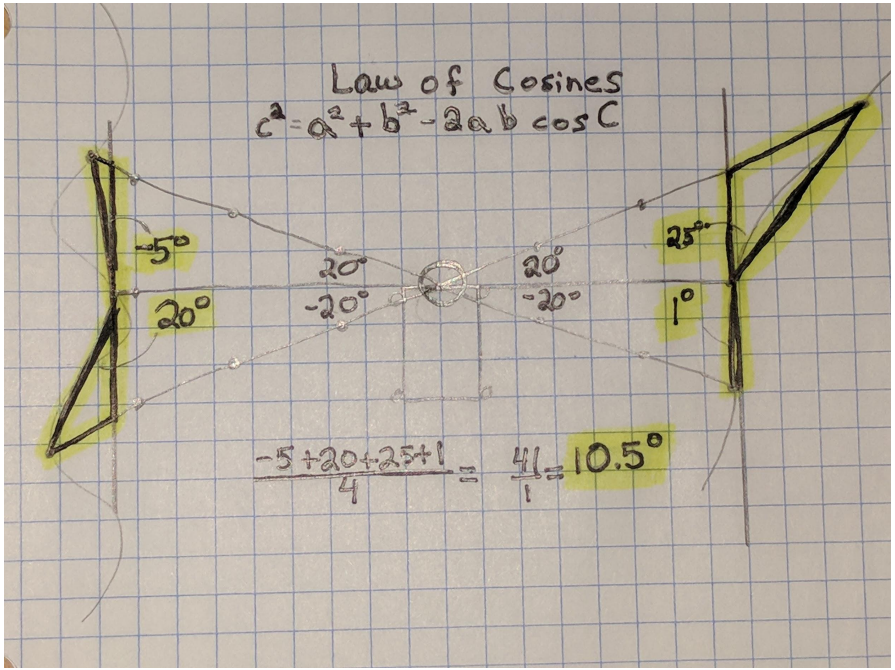
Algorithms

4. Speed modifier
    a. Increase speed when root can see further ahead
    b. speedLimit = 1.7
    c. power = 0.4
    d. speed = speedLimit - $(1/distance)^{power}$
    e. speed = 1.7 - $(1/distance)^{0.4}$



Algorithms

5. Wall Slope
    a. Calculate the overall slope of the left and right walls
    b. Improve turning smoothness



Lane Centering

Using the above wall detection algorithms, the car:

- **can anticipate the path ahead.**
- **can prevent itself from turning too sharply.**

Lessons Learned

- Go to a math professor sooner than later when dealing with calculus problems.
- Keep your code clean.

Future Modifications

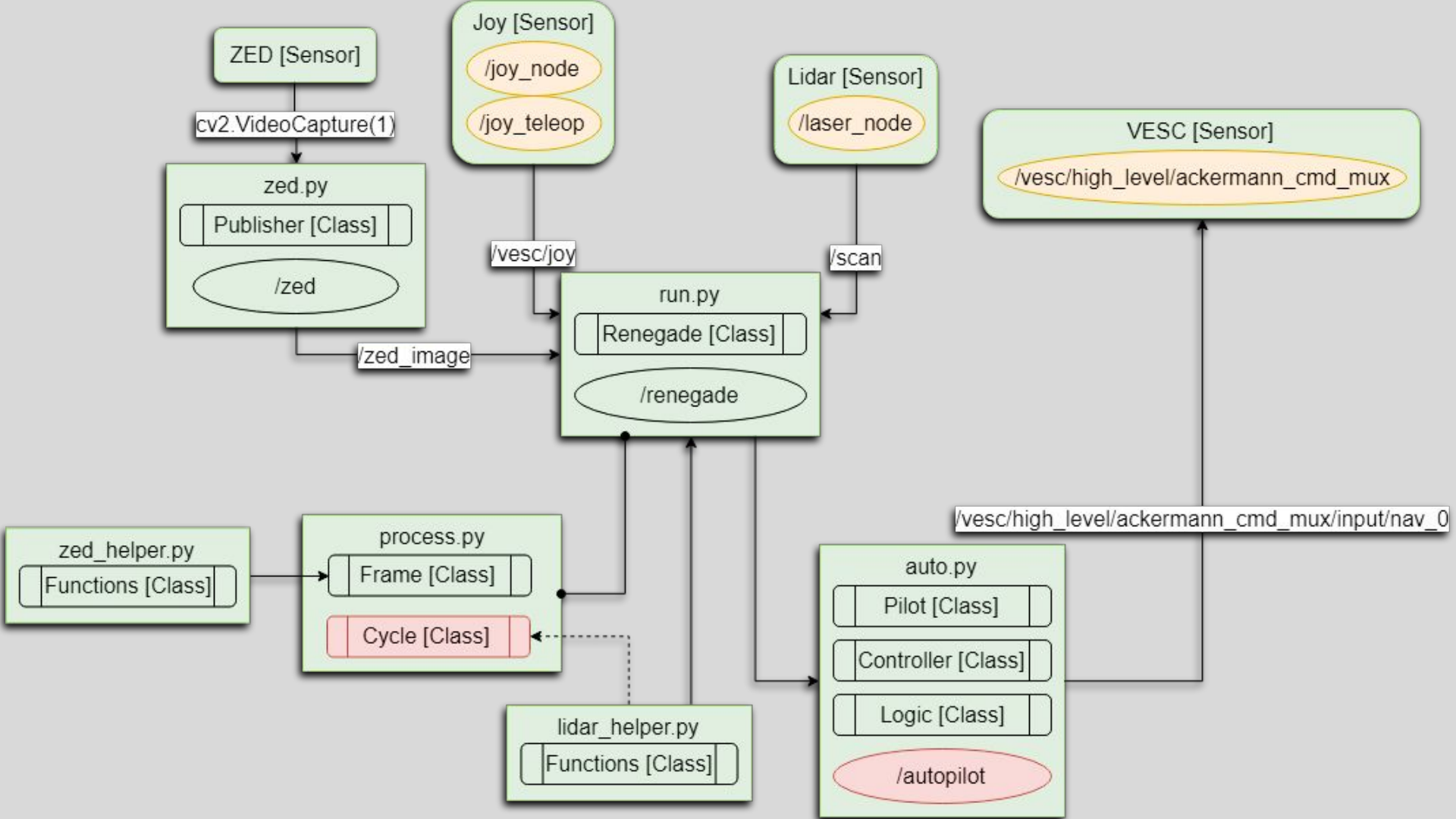
Calculate average slope of wall using linear regression of derivative of distances to wall.
Calculate average distance of a wall from the car over an area.
Make code conform to OOP standards.
Make code beautiful and short.
Comment code better.

The End

The Code

We will keep if brief.

We will only show code relevant to the lidar.
(what we show may not accurately represent our code)


ZED [Sensor]
cv2.VideoCapture(1)
zed.py
Publisher [Class]
/zed
Joy [Sensor]
/joy_node
/joy_teleop
Lidar [Sensor]
/laser_node
VESC [Sensor]
/vesc/high_level/ackermann_cmd_mux
/vesc/joy
/scan
/zed_image
run.py
Renegade [Class]
/renegade
zed_helper.py
Functions [Class]
process.py
Frame [Class]
Cycle [Class]
lidar_helper.py
Functions [Class]
auto.py
Pilot [Class]
Controller [Class]
Logic [Class]
/autopilot
/vesc/high_level/ackermann_cmd_mux/input/nav_0

run.py

```
class Renegade:
    def init:
        subscribe to lidar
    def lidar_callback(data):
        li = lidar_helper.Lidar(data, self.theta)
        wallOffset = li.offsetBetweenwalls(90, 5) #90° right of vehicle, 5° FOV
        forwardOffset = li.offsetBetweenWalls(45, 5) #45° right of vehicle, 5° FOV
        slope = li.slopeOfWalls(70, 90, 110, 3) #20° difference in angles, 3° FOV
        data = [slope, wallOffset, forwardOffset]
        auto.Pilot(data)
```