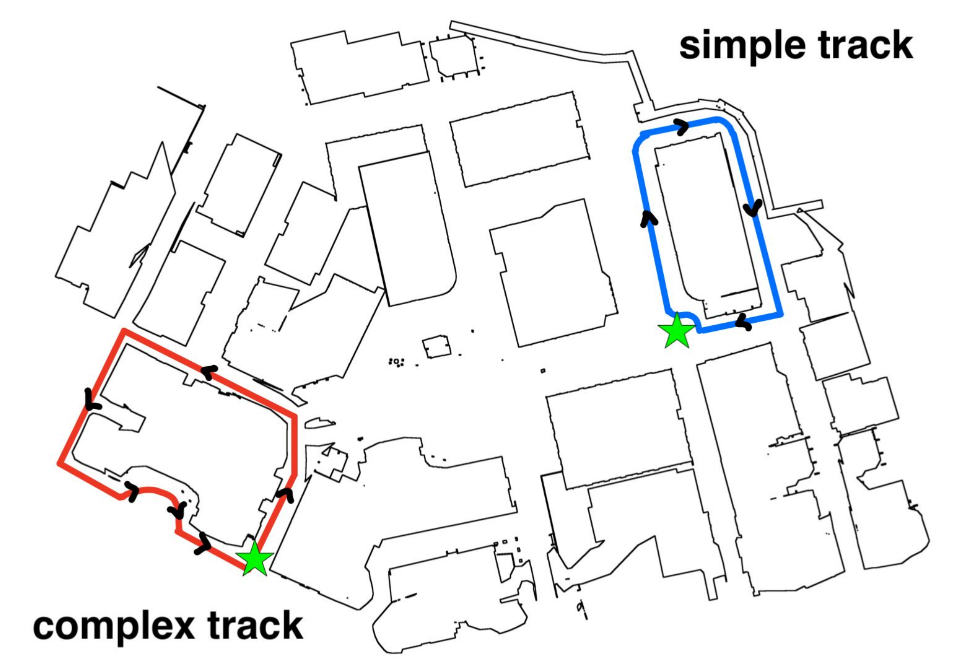
Wall Following in TESSE

RSS-2021 | Lab 3 | Team 11

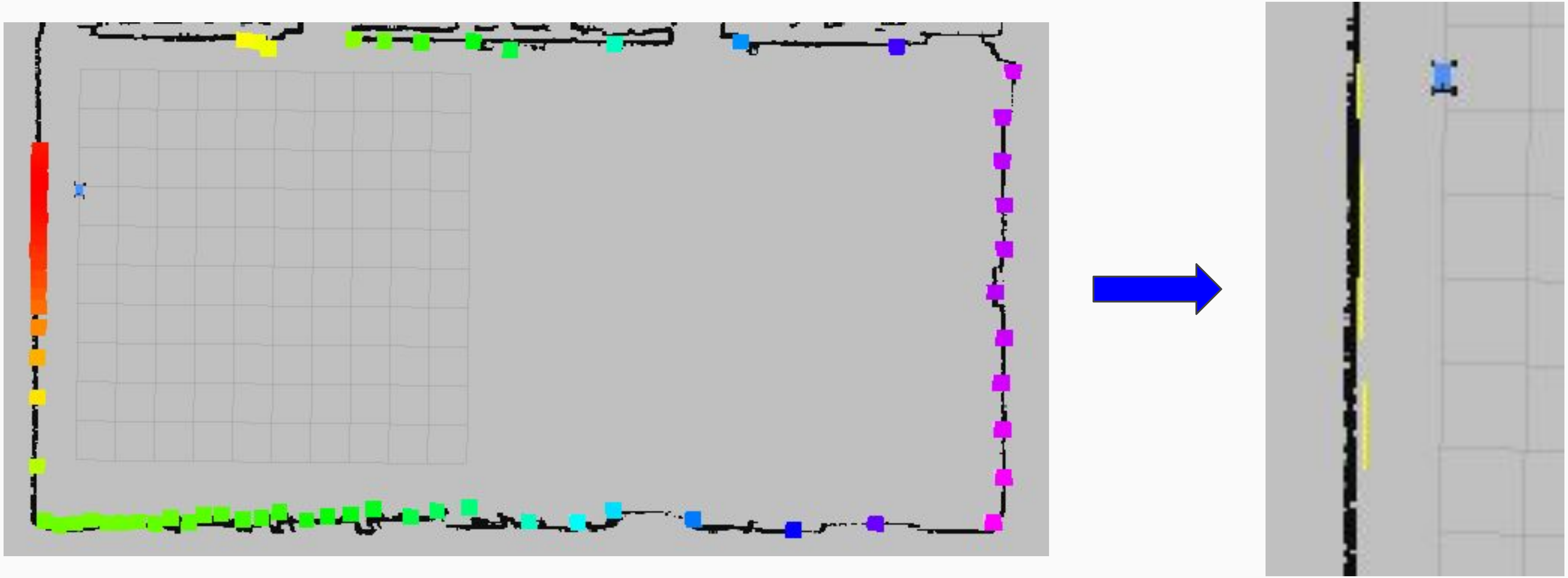
Lab Goals



- Move from 2D to 3D simulations
- Complete both wall following tracks with TESSE
- Maximize speed of car

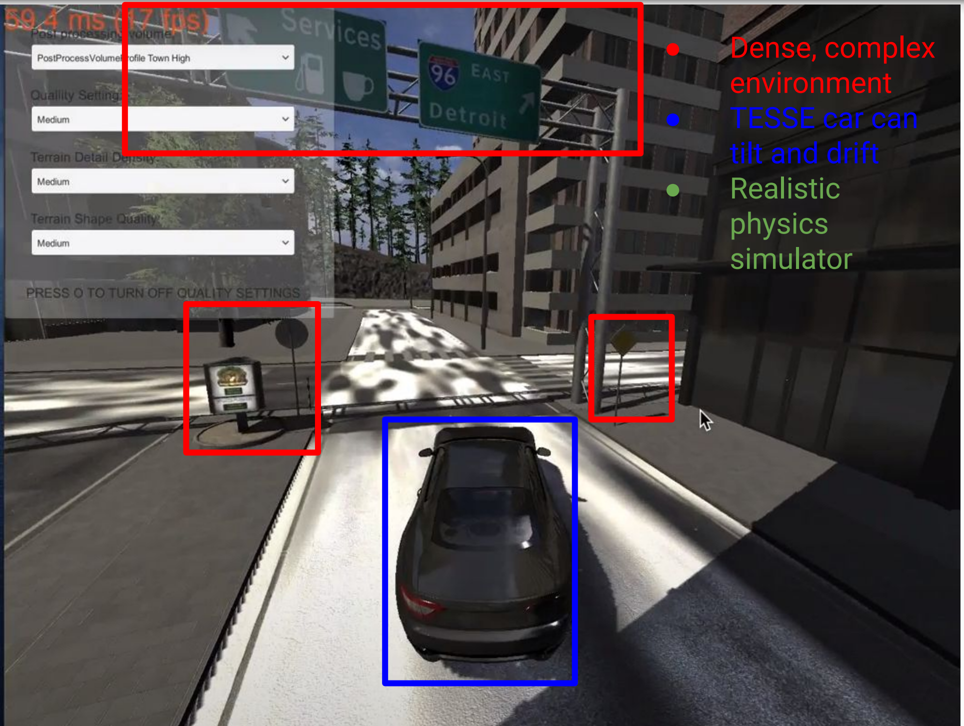
There are additional considerations when moving from 2D to 3D

2D Case

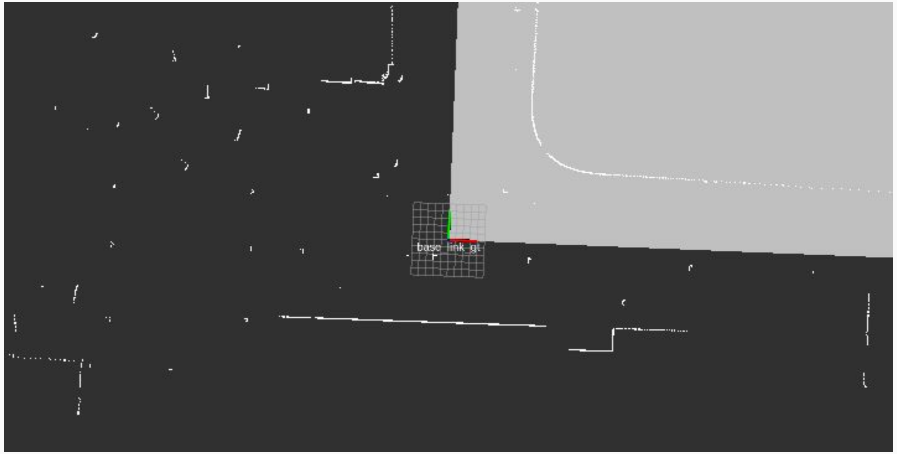


There are additional considerations when moving from 2D to 3D

3D Case



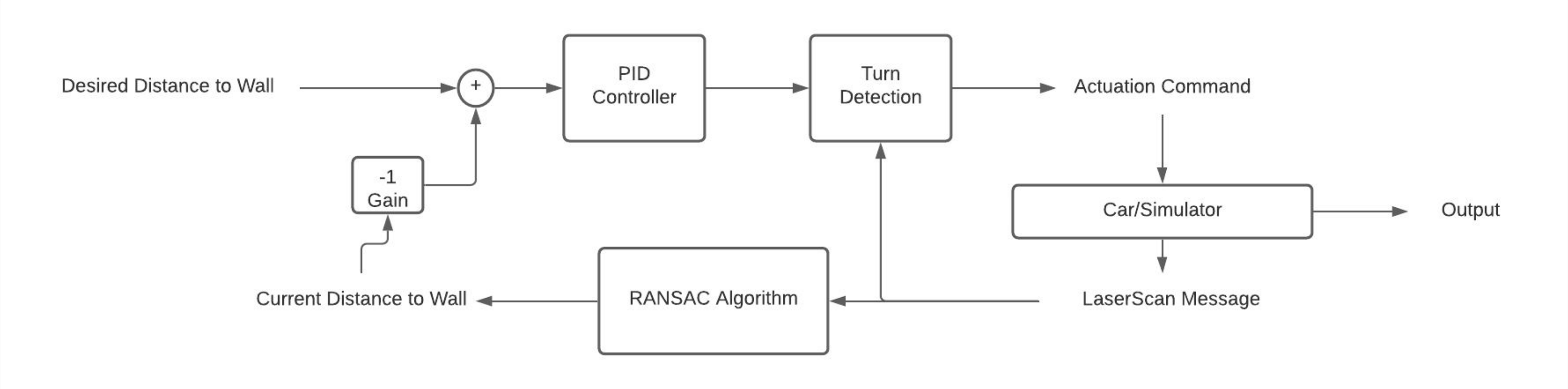
- Dense, complex environment
- TESSE car can tilt and drift
- Realistic physics simulator



Maybe show our initial sim when just using code from lab 2 on tesse

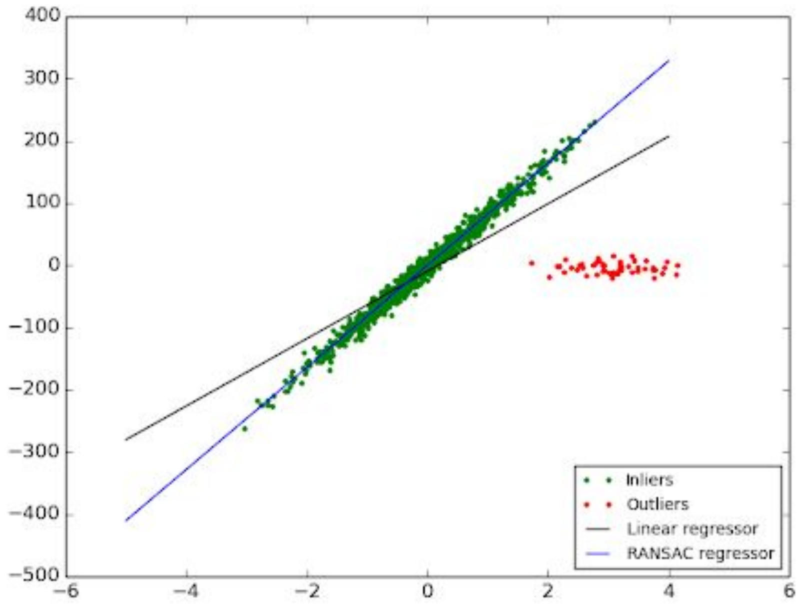
-short video of sim, car oscillates, speed not maximized

Combining PID, RANSAC, and Turn Detection builds a more robust controller



- Simulator extracts a laserscan message
- RANSAC algorithm uses this to calculate distance the car is from the wall
- This is compared w/ the desired distance from the wall to generate the error
- The error is fed through a PID controller which sets the steering command
- This is all fed through a turn detector which acts as an arbitrator detecting upcoming turns and setting appropriate steering and velocity commands

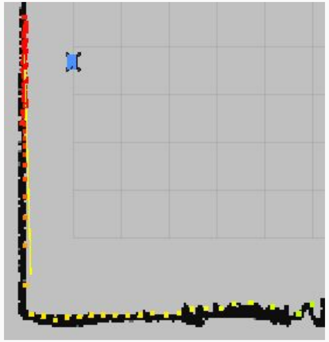
The RANSAC algorithm can make wall detection more robust to outliers



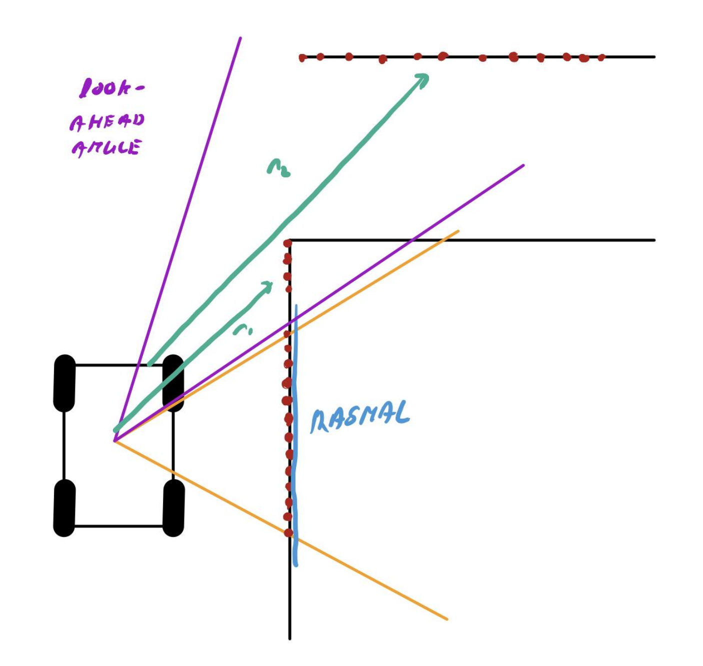

```python
from sklearn import linear_model, datasets
```

```python
# Robustly fit linear model with RANSAC algorithm
ransac = linear_model.RANSACRegressor()
ransac.fit(X, y)
```

Altering the speed of the car improves how the car handles turns



- split scan into two section, “control” section and look-ahead section
- control used to generate RANSAC line, look-ahead looks for distance jumps in the scan to determine when a turn is approaching
- car slows when turn is approaching

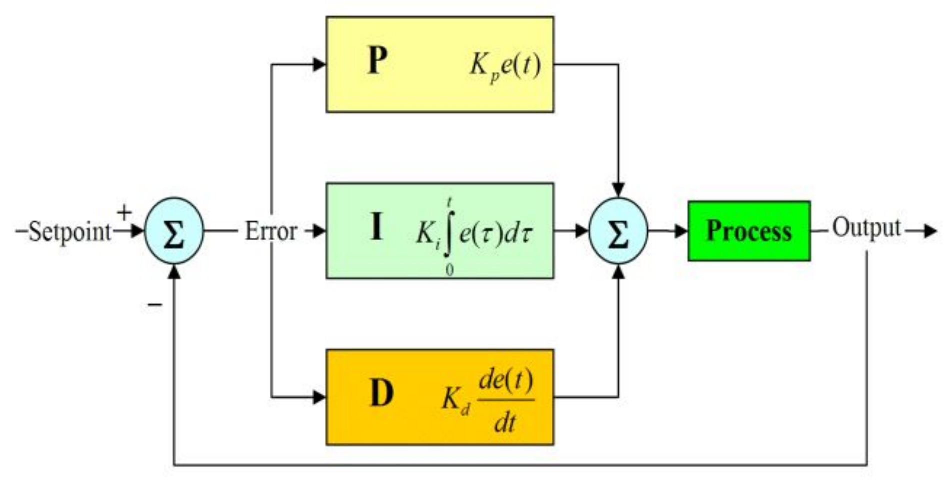


Tuning (PID + Turn Detection)

- P - drives the system towards the setpoint, increases disturbance rejection
- I - drives the steady-state-error of the system near zero
- D - stabilizes P and I to prevent overshoot + oscillation

***Used the above when tuning the PID Parameters

- Turn detection lookahead angles and velocities were tuned with trial and error



https://www.researchgate.net/figure/PID-Block-Diagram-PID-stands-for-Proportional-Integral-Derivative-control-A-PID_fig1_316709017

Need to set parameters in the two regions, straight-line and during a turn

Setting velocity of car from turn detection (distance to turn):

$$V_f^2 = V_i^2+2*a*D$$

$$a=(V_f^2 - V_i^2)/2*D$$

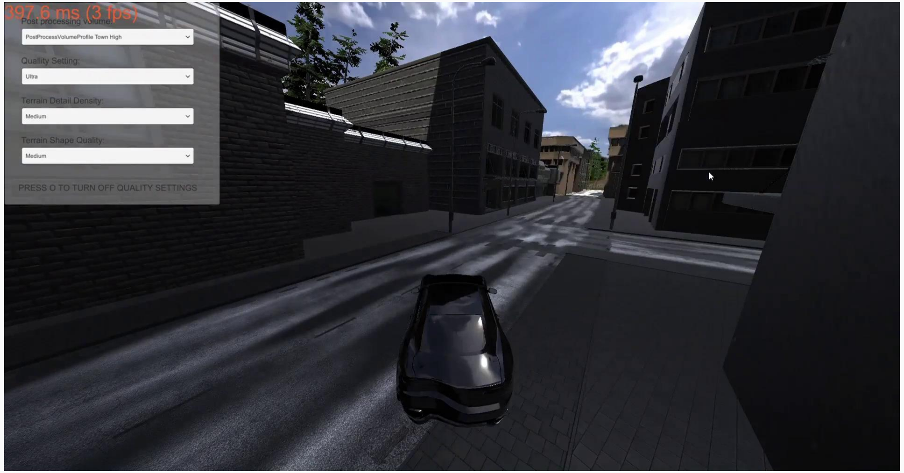
D - distance to the wall

V_f - tuned turn speed @ 2m/s

V_i - current car velocity

a - desired acceleration to publish

Current progress on system



- Car goes around the simple track!!!
    - This is w/ Linear Regression (not RANSAC) basic PD tuning (no integral), and turn detection
    - No other intelligence or filtering in the system
    - **Kp = 1.0, Ki = 0, Kd = 0.44**
- Also tested w/ initial implementation of turn-detection + initial RANSAC tests
- Video shows lap of the track in the TESSE simulator
- Needs some more tuning

Goals for Friday

Base Goals:

- Start w/ implementing the same controller as above w/ the RANSAC algorithm
- Increase the speed and attempt to tune the PID to the highest speed achievable with just PID gains
- Do this for both tracks

Reach Goals:

- Implement turn detection and kinematics for turn control based on previous description
- Have car run at higher speed along the track and slow down as we approach a wall at a specified acceleration to turn
- Optimize on both tracks