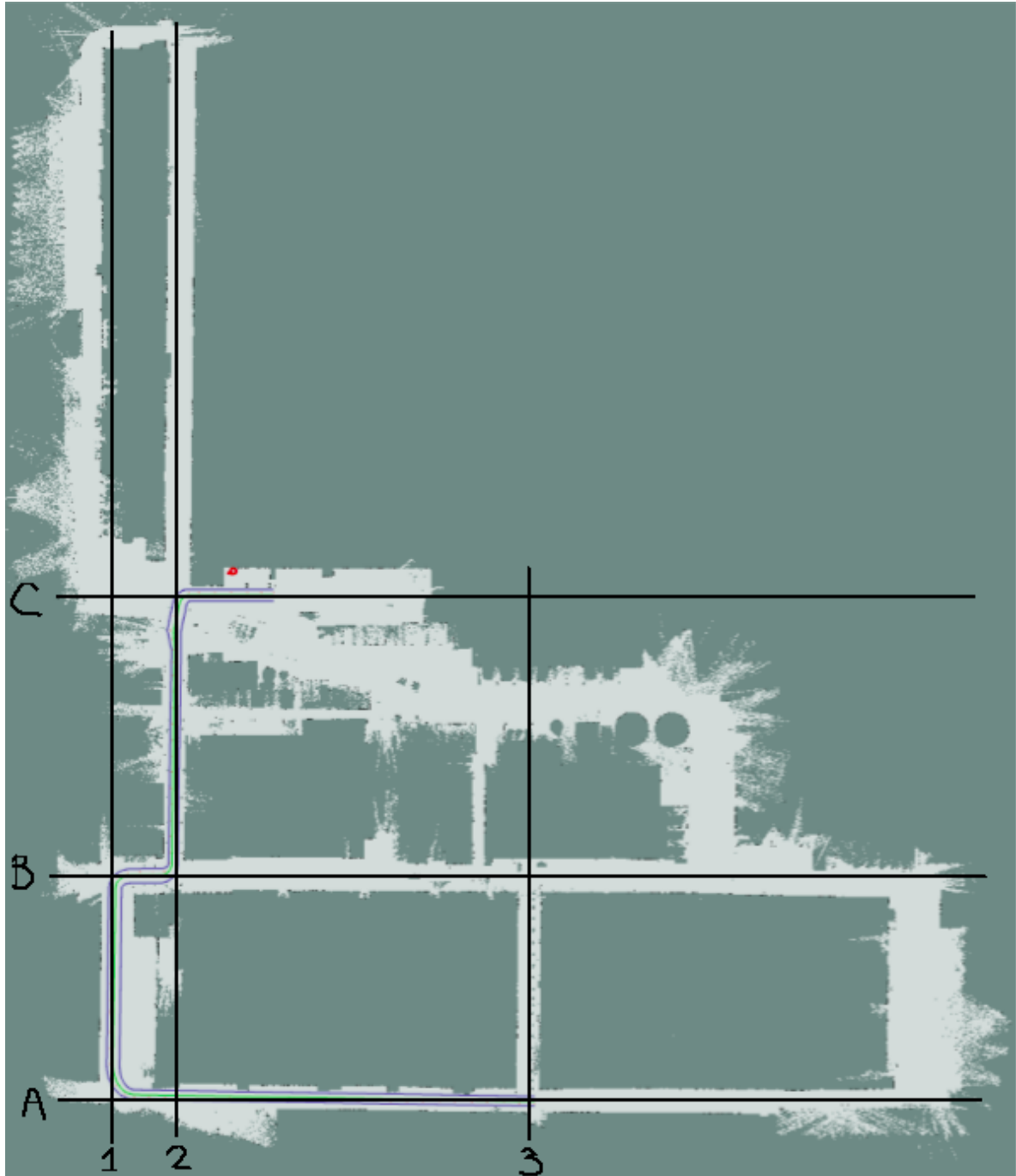


Lanes

Naming Convention for Lane Sections



- For straight paths: **str_XXX**
 - eg: str_31A is the straight path in the 10X to 10A direction (*column 3 -> 1, row A*)
 - naming for direction is always from 10X -> 10A
- For curved paths: **crv_XX**
 - eg: crv_1A is the curved path at *column 1, row A*

Method to Obtain Raw Coordinates of Paths

1. Record the (x, y) coordinates while driving on the actual centerline of the 2 lanes (also the right boundary of the 2 lanes)

```
roslaunch amcl_pose
```

2. Convert bag file to csv file

```
rostopic echo -b <filename>.bag -p /amcl_pose <new_filename>.csv
```

3. Copy only the (x, y) coordinates (*pose.pose.position.x* and *pose.pose.position.y*) into a new path file (named as described above, extension .path) in the "agv_lanes/path_files/" folder. **Only need to do so for straight paths.** Way to create curved path files will be explained later on.

Generate Lane Boundaries by Offset



Naming of lane boundaries (red arrow is in the direction 10X -> 10A)

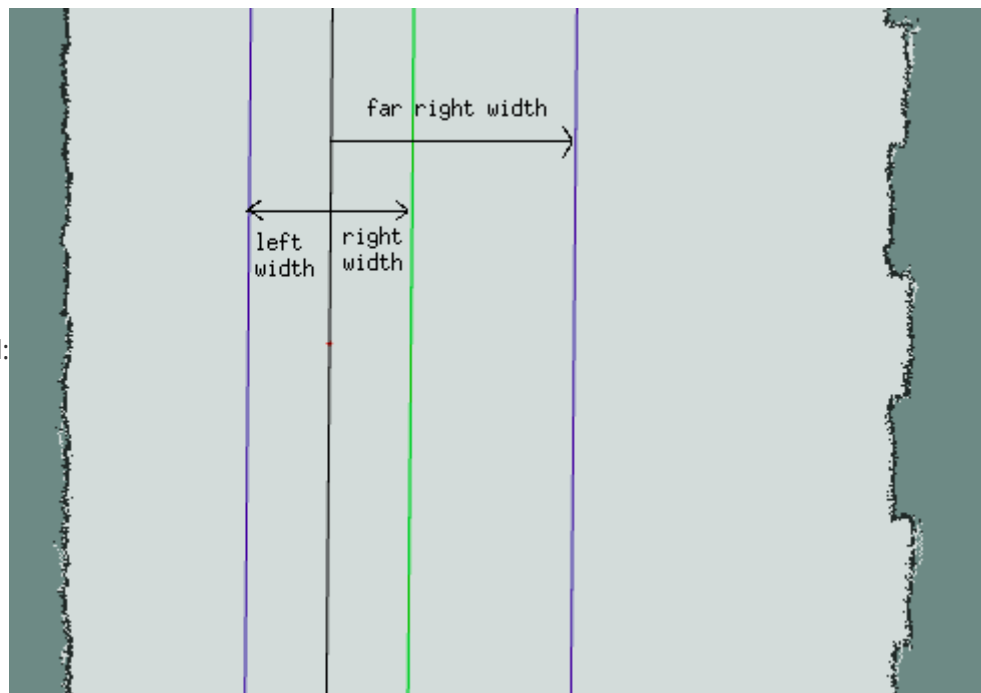
- Green line: right_boundary
- Left blue line: left_boundary_xa
- Right blue line: left_boundary_ax
- Left black line: centerline_xa
- Right black line: centerline_ax

Main idea:

1. Perform linear fit on straight paths and ellipse fit on curved paths.
2. Measure lane widths of different lane sections and offset from the actual centerline (green line).

To run codes (codes are on Brina's UBUNTU):

In new terminal:



Currently using Jupyter Notebook (dependencies: need to install Anaconda/mini-conda), but can explore the new Jupyter Lab.

```
conda activate
```

```
jupyter notebook agv_lanes
```

NOTE: to run any block of code, click on the chunk of code and press shift+enter.

- **Lane boundaries for straight paths**

- click to open `Lane Boundaries for Straight Paths (str_xxx).ipynb`
- remember to change the parameters accordingly:
 - input file name
 - separation between waypoints (eg. 10 if waypoints are to be 10 meters apart)
 - offset to the centerlines and the left boundaries of each of the 2 lanes
- run the blocks of code sequentially
- check that offsets are on the correct sides
- once all straight boundaries done, concatenate files (*refer to last section of documentation*) then visualize on rviz:

In new terminal:

```
roslaunch agv unit3.launch
```

In separate terminal:

```
rostopic pub /path_filename_topic std_msgs/String "data:  
'<filename>.path'"
```

--> vip3a (for 10X -> 10A)

--> vip3x (for 10A -> 10X)

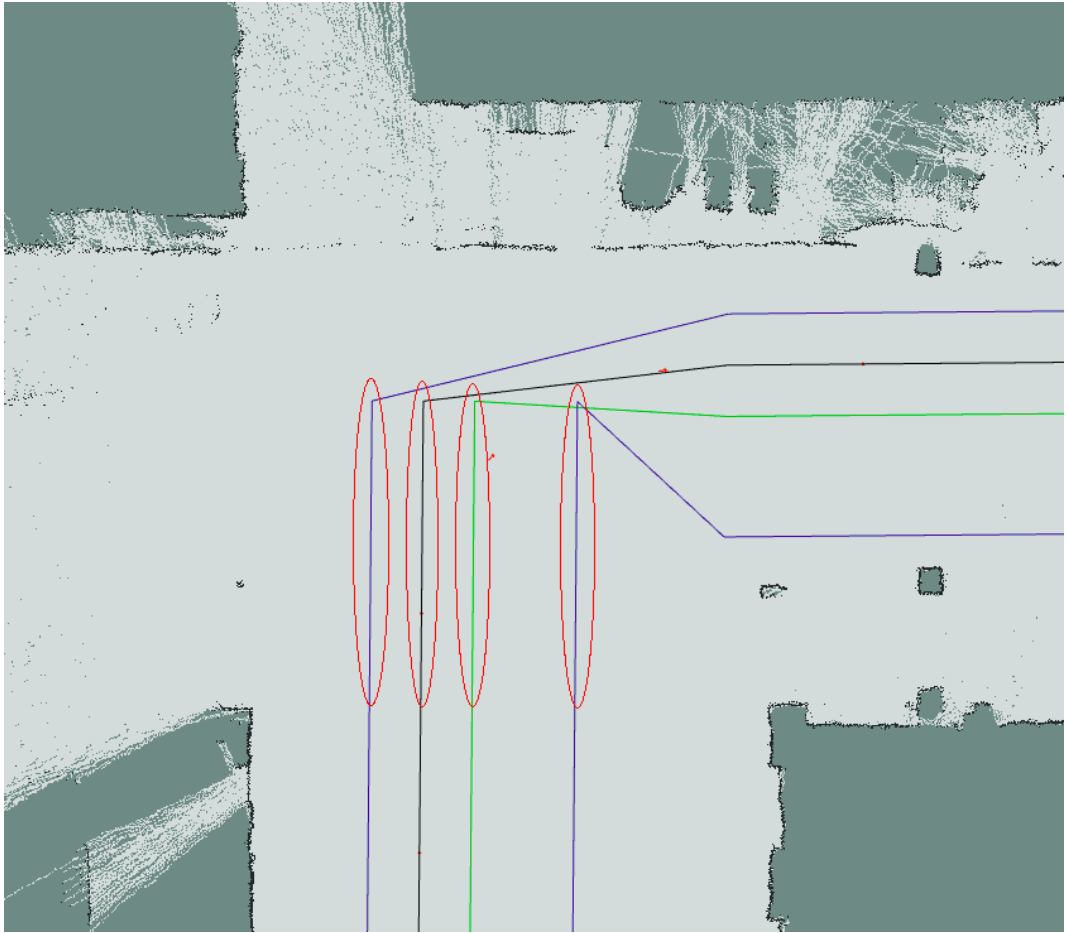
- manually delete points that are out of bounds (shown below in red)

To determine coordinates of a point,

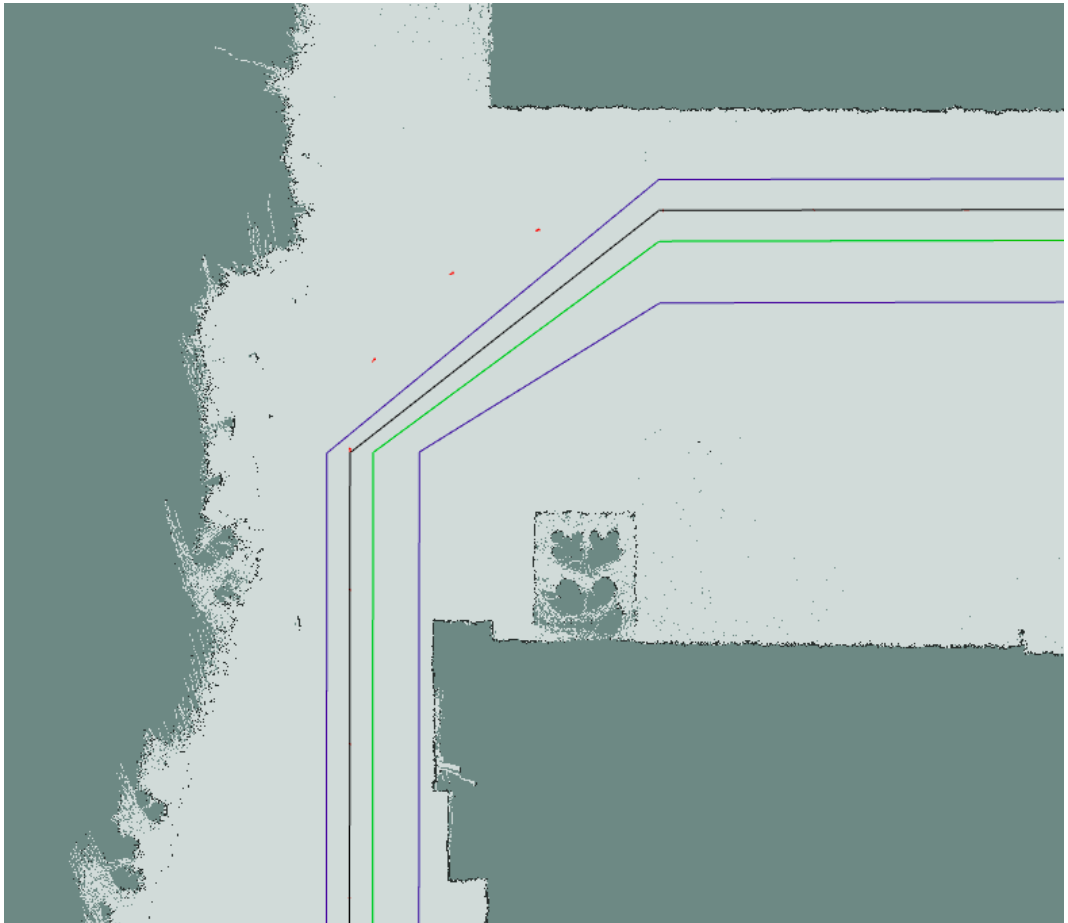
- In new terminal:

```
rostopic echo /move_base_simple/goal
```

- Using 2D Nav Goal, select point on map to obtain coordinates of the point



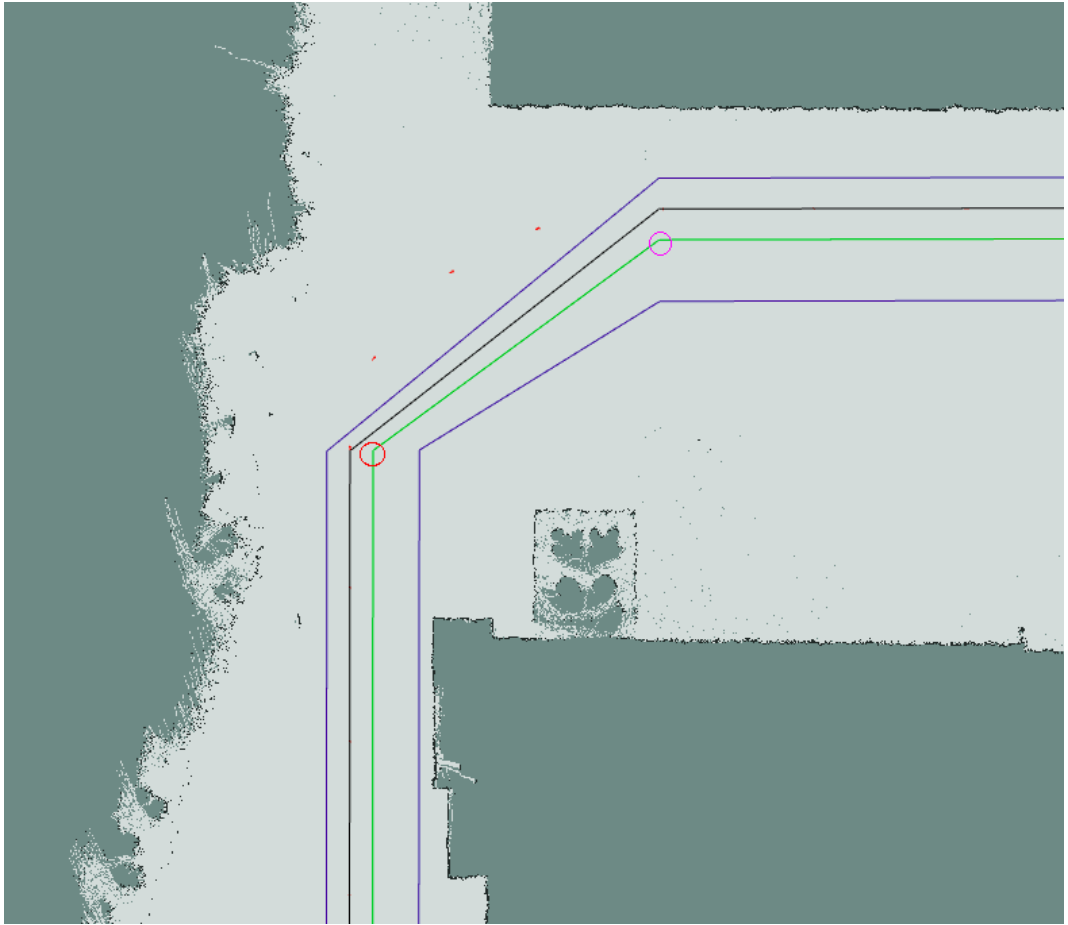
- all straight sections should look like:



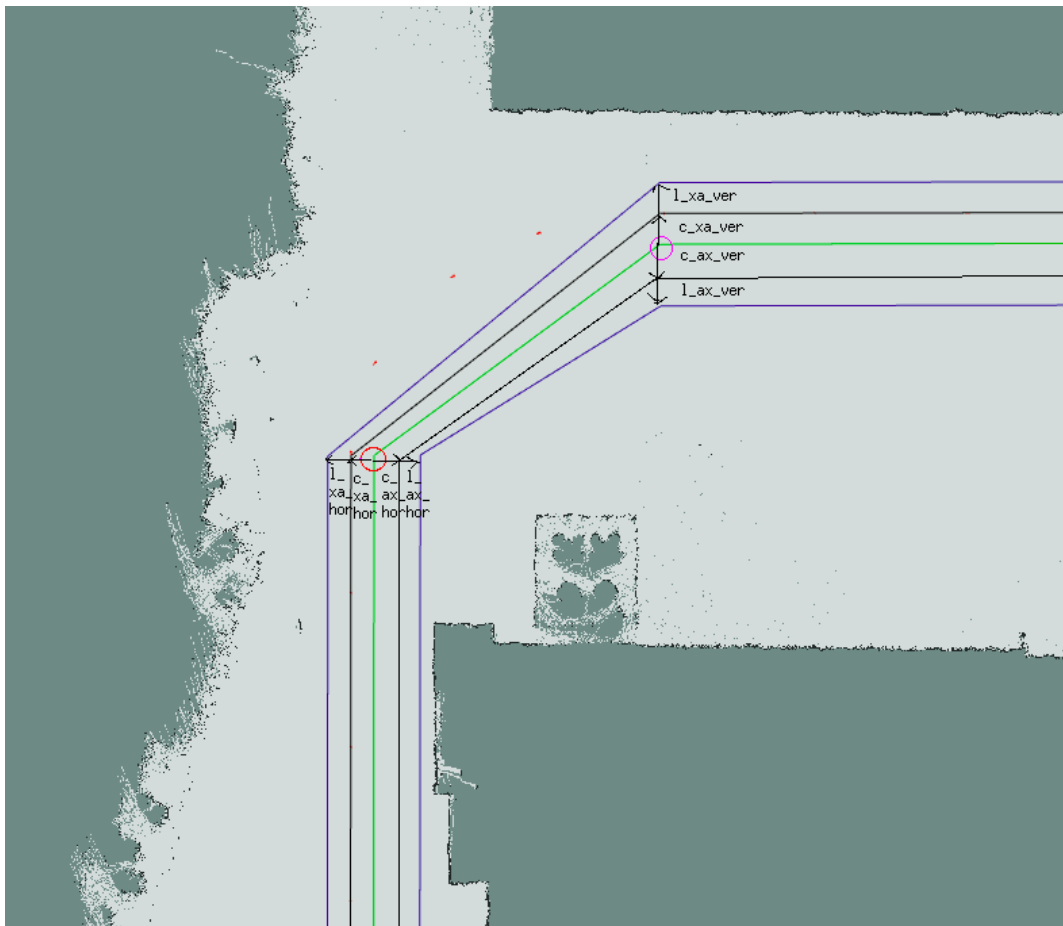
- proceed to plot curved boundaries

- **Lane boundaries for curved paths**

- click to open `Lane Boundaries for Curved Paths (crv_xx).ipynb`
- the path file should contain 2 points:



- 1st point (red circle) - last waypoint of the previous straight section (the adjusted .lane file in the Right_boundary folder) (*not the raw coordinates*)
 - 2nd point (pink circle) - first waypoint of the next straight section (also the .lane file in the Right_boundary folder)
 - **NOTE: Tangents to the two points taken must be perpendicular to one another!**
- remember to change the parameters accordingly:
 - input file name
 - offset **from the green line** to the centerlines (c) and the left boundaries (l)
hor -> horizontal offset parallel to the x-axis; ver -> vertical offset parallel to the y-axis



- in block 4, determine where the center of ellipse should be (based on 1st and 2nd points), and comment out whichever is not applicable. (eg. for lane section above, center of ellipse is on the right of the 1st waypoint)

```
# comment out whichever is not applicable!!!

# if center of ellipse is on the left or right of 1st waypoint
h=ellipse_waypoints[1][0]
k=ellipse_waypoints[0][1]
a=ellipse_waypoints[0][0]-h
b=ellipse_waypoints[1][1]-k

# # if center of ellipse is on the left or right of the 2nd waypoint
# h=ellipse_waypoints[0][0]
# k=ellipse_waypoints[1][1]
# a=ellipse_waypoints[1][0]-h
# b=ellipse_waypoints[0][1]-k
```

- check that the arc shape for actual centerline is correct
 - adjust the values in the equations to obtain a nice arc with enough number of points.
NOTE: may get nan due to out-of-bound points, but can delete those later on
 - check that the offsets are on the correct sides
 - once all curved boundaries done, add to the straight lane files (*sequence of points matters*)
 - visualize on rviz and smoothen boundaries (*by adjusting coordinates of points because straight and curved sections may not be connected smoothly*)

Generate Lane Info files

- Create lane info files when boundaries are finalized

Format of lane info files:

≡ str_31A.laneinfo x

home ▸ micron ▸ agv_lanes ▸ Lane_info_xa ▸ ≡ str_31A.laneinfo

```
1 25
2 -38.671 105.356 1.000 -0.006 0.000 1.500 1.500 4.500 0
3 -38.616 114.501 1.000 -0.006 9.145 1.500 1.500 4.500 0
4 -38.556 124.500 1.000 -0.006 19.145 1.500 1.500 4.500 0
5 -38.496 134.500 1.000 -0.006 29.145 1.500 1.500 4.500 0
6 -38.436 144.500 1.000 -0.006 39.145 1.500 1.500 4.500 0
7 -38.376 154.500 1.000 -0.006 49.145 1.500 1.500 4.500 0
8 -38.316 164.500 1.000 -0.006 59.145 1.500 1.500 4.500 0
9 -38.255 174.499 1.000 -0.006 69.145 1.500 1.500 4.500 0
10 -38.195 184.499 1.000 -0.006 79.145 1.500 1.500 4.500 0
11 -38.135 194.499 1.000 -0.006 89.145 1.500 1.500 4.500 0
12 -38.075 204.499 1.000 -0.006 99.145 1.500 1.500 4.500 0
13 -38.015 214.499 1.000 -0.006 109.145 1.500 1.500 4.500 0
14 -37.955 224.499 1.000 -0.006 119.145 1.500 1.500 4.500 0
15 -37.895 234.498 1.000 -0.006 129.145 1.500 1.500 4.500 0
16 -37.835 244.498 1.000 -0.006 139.145 1.500 1.500 4.500 0
17 -37.775 254.498 1.000 -0.006 149.145 1.500 1.500 4.500 0
18 -37.714 264.498 1.000 -0.006 159.145 1.500 1.500 4.500 0
19 -37.654 274.498 1.000 -0.006 169.145 1.500 1.500 4.500 0
20 -37.594 284.497 1.000 -0.006 179.145 1.500 1.500 4.500 0
21 -37.534 294.497 1.000 -0.006 189.145 1.500 1.500 4.500 0
22 -37.474 304.497 1.000 -0.006 199.145 1.500 1.500 4.500 0
23 -37.414 314.497 1.000 -0.006 209.145 1.500 1.500 4.500 0
24 -37.354 324.497 1.000 -0.006 219.145 1.500 1.500 4.500 0
25 -37.294 334.497 1.000 -0.006 229.145 1.500 1.500 4.500 0
26 -37.184 352.786 1.000 -0.006 238.289 1.500 1.500 4.500 0
27
```

1 <number_of_waypoints>

2 <x> <y> <dx> <dy> <s> <left_width> <right_width> <far_right_width>
<special_point>

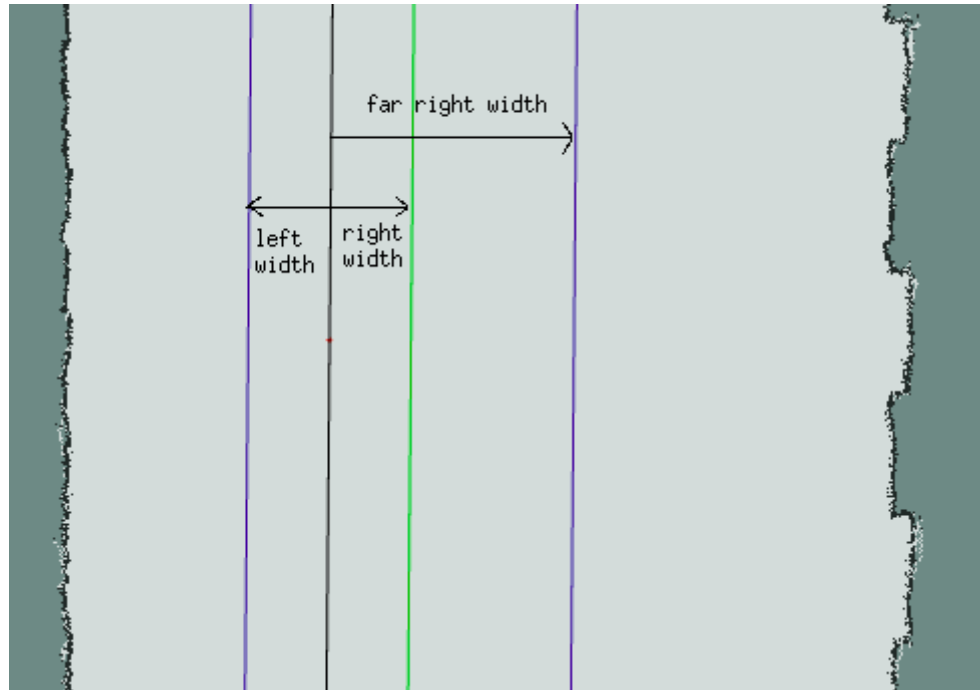
3 <x> <y> <dx> <dy> <s> <left_width> <right_width> <far_right_width>
<special_point>

4 ...

5 ...

- **<special_point>** can take any one of the 4 possible values:
 - 0: normal waypoint
 - 1: endpoint
 - 2: turning point
 - 3: slope point
 - By default, special_point is set to be 0. This value can be manually changed once a waypoint is identified as a special waypoint (eg turning point).
- **Lane info files for straight paths**
 - Click to open `Lane Info for Straight Paths(str_XXX).ipynb`
 - remember to change the parameters accordingly:
 - choose the direction and comment out the folder names that are not applicable

- change the input file name
- change the left, right and far right width of the lane section



- set the separation distance between consecutive waypoints

- **Lane info files for curved paths**

- Click to open `Lane Info for Curved Paths(crv_xx).ipynb`
- remember to change the parameters accordingly:
 - choose the direction and comment out the folder names that are not applicable
 - change the input file name
 - set the actual number of waypoints in the file
- set the desired number of waypoints (can set this to be more than needed, then delete unnecessary ones later)
 - set the left, right and far right width for both entry and exit
 - determine where the center of ellipse should be (based on 1st and 2nd points), and comment out whichever is not applicable.

```
# comment out whichever is not applicable!!!

# if center of ellipse is on the left or right of 1st waypoint
h=ellipse_waypoints[1][0]
k=ellipse_waypoints[0][1]
a=ellipse_waypoints[0][0]-h
b=ellipse_waypoints[1][1]-k

# # if center of ellipse is on the left or right of the 2nd waypoint
# h=ellipse_waypoints[0][0]
# k=ellipse_waypoints[1][1]
# a=ellipse_waypoints[1][0]-h
# b=ellipse_waypoints[0][1]-k
```

- **IMPORTANT NOTE: Must ensure that the the tangents to the first and last waypoints are perpendicular since these points determine the center of ellipse. The ellipse fit should be as accurate as possible (points lie on arc).**

Concatenate files

- Concatenate files to get:
 - centerline_xa.lane
 - centerline_ax.lane

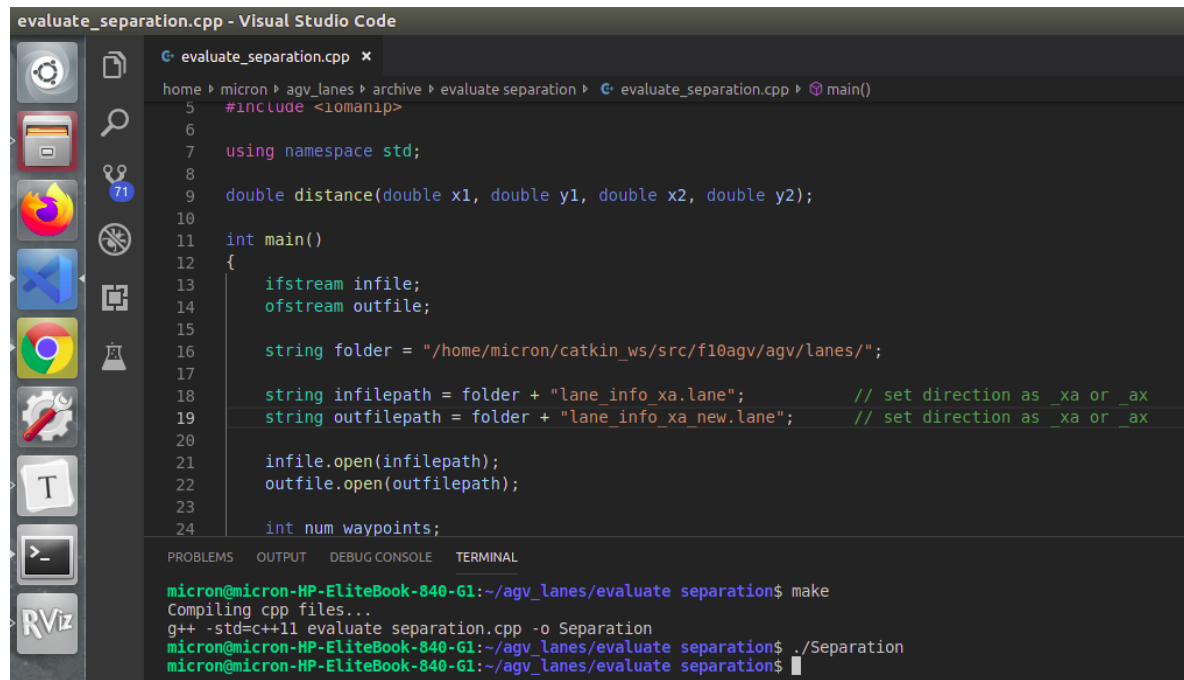
- left_boundary_xa.lane
- left_boundary_ax.lane
- right_boundary.lane
- lane_info_xa.lane
- lane_info_ax.lane

NOTE: When adding a new lane section, just insert new data into old files!

- For lane info files, due to concatenation, need to **manually add up** the number of waypoints for each lane section and indicate only the **total** number of waypoints at the **first line** of the file. (Refer to lane info file format)

Evaluate Separation

- When lane info files are finalized, need to recalculate the cumulative separation between waypoints (*s value for each waypoint represents the approximate distance of that waypoint from the first waypoint*)



The screenshot shows the Visual Studio Code editor with the file `evaluate_separation.cpp` open. The code defines a `distance` function and a `main` function that reads lane info files and writes a separation file. The terminal at the bottom shows the execution of `make` and `./Separation`.

```

evaluate_separation.cpp - Visual Studio Code
home ▸ micron ▸ agv_lanes ▸ archive ▸ evaluate separation ▸ evaluate_separation.cpp ▸ main()
5  #include <iomanip>
6
7  using namespace std;
8
9  double distance(double x1, double y1, double x2, double y2);
10
11 int main()
12 {
13     ifstream infile;
14     ofstream outfile;
15
16     string folder = "/home/micron/catkin_ws/src/fl0agv/agv/lanes/";
17
18     string infilepath = folder + "lane_info_xa.lane";           // set direction as _xa or _ax
19     string outfilepath = folder + "lane_info_xa_new.lane";     // set direction as _xa or _ax
20
21     infile.open(infilepath);
22     outfile.open(outfilepath);
23
24     int num waypoints;

```

```

micron@micron-HP-EliteBook-840-G1:~/agv_lanes/evaluate separation$ make
Compiling cpp files...
g++ -std=c++11 evaluate separation.cpp -o Separation
micron@micron-HP-EliteBook-840-G1:~/agv_lanes/evaluate separation$ ./Separation
micron@micron-HP-EliteBook-840-G1:~/agv_lanes/evaluate separation$

```

- Change to the desired direction for infilepath and outfilepath (`_xa` or `_ax`)
- In terminal:
 - change directory to `agv_lanes/evaluate_separation`
 - to compile: `make`
 - to run executable: `./Separation`
- *NOTE: s values are currently not used, but can be used to calculate ETA in future*