* Run two perimeter runs while saving the data into rosbag files.
* The program ~/project\_notes/code\_for\_testing/archive/rosbag\_utilities/rosbag\_latlon\_to\_json\_and\_csv.py reads from the /fix topic in a rosbag and extract the lat/lon data and put into .csv file.
* Open the script and adjust the output and input file names for your recent data.
* Run the script and open the .csv file output
* Extract the points for the cutting area and placed that data in a different .csv file (e.g. 'collins\_dr\_62\_A\_from\_rosbag\_combined.csv').
* I visually inspected the plots from the two separate runs and kept points that I deemed best or an option is to choose four corners and make a simple polygon instead of a full perimiter.
* Part of that process included attributing each point with identifiers to show which segment the point was associated with and the sequence the point should be run in for the counter-clockwise path. I used Plotjuggler to review the quality of the points and get time stamps to determine which points to keep and to discard.
* 1
* 2
* 3
* 4
* Once you either have a hard code polygon or a series of points that represent a perimeter, open /home/tractor/ros1\_lawn\_tractor\_ws/project\_notes/code\_for\_testing/archive/boustrophedon/path\_boustrophedon\_coverage\_variable\_degree.py
* • Update ‘csv\_path’ with the filename of the innermost ring to use
* • Run the program and adjust the degree’s until you are happy (i.e. path = main(19)). I used 19 degree lines. The output will be used as input for ~/archive/boustrophedon/plotIntersectingPointsV8.py
* When using the Boustrophedon process you may see line segments that are short or have multi-paths, or disconnected line segments. These need to be deleted.
* See additional instructions at ~/project\_notes/Path Planning - Boustrophedon path creation notes.odt
* So if the points are listed as clockwise you can make one list that is clockwise and one list that is counter-clockwise
* First start with a list that also has the distance from 0, 0 to find the closest point
* /home/tractor/ros1\_lawn\_tractor\_ws/project\_notes/code\_for\_testing/archive/path\_step5\_inner\_rings\_v2.py
  + Reads collins\_dr\_62\_A\_step2.csv' and creates 5 inner rings
    - Read the inner ring,
    - make a file that calculates distance;
    - print which record is closet
    - Plot the points around the closest
* After running path\_boustrophedon\_coverage\_variable\_degree.py the stripes are defined but the Dubin’s ‘keyhole’ u-turns are not. To do that the intersection points that will define the start and stop of the u-turns need to be defined.
* Use the spreadsheet to hand build the x, y, angle data as input into the path planner based on the line segments defined in the Boustrophedon script.

Earlier version

* Run two perimeter runs; Save both rosbag files into .csv files with the time delta, time stamp and lat/lon
  + 'collins\_dr\_62\_A\_from\_rosbag\_run1'
  + 'collins\_dr\_62\_A\_from\_rosbag\_run2'
* I have ran two paths. One included more areas beyond the 'main' cutting area which can be used later.
* I have written the program 'rosbag\_latlon\_to\_json\_and\_csv.py' to read from the /fix topic in a rosbag and extract the lat/lon data and put into .csv files
* I then manually merged the .csv data in excel to get the points only for the 'main' cutting area and placed that data in a different .csv file 'collins\_dr\_62\_A\_from\_rosbag\_combined.csv'. I visually inspected the plots from the two separate runs and kept points that I deemed best. Part of that process included attributing each point with identifiers to show which segment the point was associated with and the sequence the point should be run in for the counter-clockwise path. I used Plotjuggler to review the quality of the points and get time stamps to determine which points to keep and to discard. I used PowerPoint to document the segments that I needed to separate out.
* I have written the program 'path\_csv\_file\_to\_cartesian.py' to convert the lat/lon data into Cartesian coordinates (i.e. x and y points), and then plot those coordinates using matplotlib following a counter clockwise pattern. ***(This seems redundant at the moment, because when you saved the rosbag file the /odom statement has x, y position based on the same origin.)***
* So if the points are listed as clockwise you can make one list that is clockwise and one list that is counter-clockwise
* First start with a list that also has the distance from 0, 0 to find the closest point
* /home/tractor/ros1\_lawn\_tractor\_ws/project\_notes/code\_for\_testing/archive/path\_step5\_inner\_rings\_v2.py
  + Reads collins\_dr\_62\_A\_step2.csv' and creates 5 inner rings
    - Read the inner ring,
    - make a file that calculates distance;
    - print which record is closet
    - Plot the points around the closest
* What do I need to do to make a path for input.txt
  + Calculate angle and place it in the .csv file
    - 'code\_for\_testing/archive/path\_step3\_v1.py' currently reads a .csv file with only the lat/lon in it. The it outputs lat, lon and angle using 'extended\_coordinates\_list.append((round(x1, 2), round(y1, 2), round(angle, 2)))' so do you want to add this logic to your existing program or have 'path\_step3\_v1.py' read the .csv file or have some other program output a file that can be used by 'path\_step3\_v1.py'?
    - 'collins\_dr\_62\_A\_step3.txt' has the angle, look ahead and speed in it
    - Update the 'cartesian.py' to output a file that can be read by 'step4.py'
    - Update 'step4.py' to update the .csv file with the angle, lookahead and speed
  + Add angle, look ahead and speed
* Next tests:
* Drive around the obstacles to get their position
* Build out the yard and complete a full mowing

Earlier Notes:  
  
Current Origin: origin\_lat:=40.48524688166667, origin\_lon:=-80.332720941667

<https://www.gpsvisualizer.com/map_input?form=google>

Getting Google Maps Key for GPS Visualizer

<https://console.cloud.google.com/google/maps-apis/credentials?project=scenic-torch-398017>

Code: AIzaSyDrHr7Y0NkLQKMEIhl2mKGNTuVdA\_qFnOk