# Data Analysis and Discussion

April 4th, 2020 S. Vigil

**Introduction:**

The Obstacle Detector State Machine has been produced and run on features data provided by KBRwyle, contactor for NASA. It has analyzed features provided and recognized objects in that data as presented below.

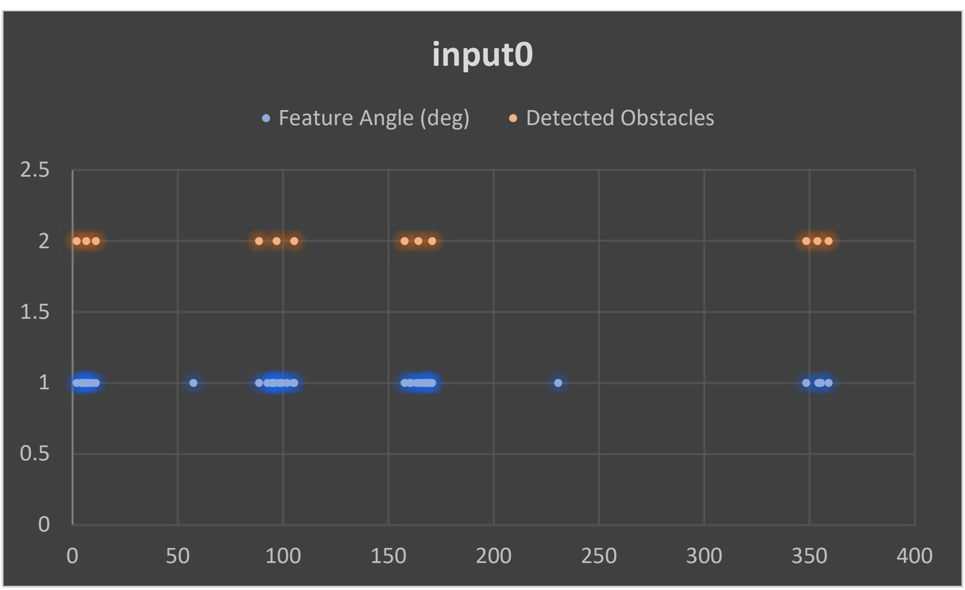
While the Obstacle Detector does a great job of detecting obstacles presented by rather simple feature datasets, it is in the end a demonstration of development capabilities and process on my part.

For a real-world application, more work on the analysis algorithm. Also, more *apriori* knowledge of the features environment will be necessary. This will allow for proper tuning of the Obstacle Detector and more accurate obstacle recognition “in the wild”.

**Discussion:**

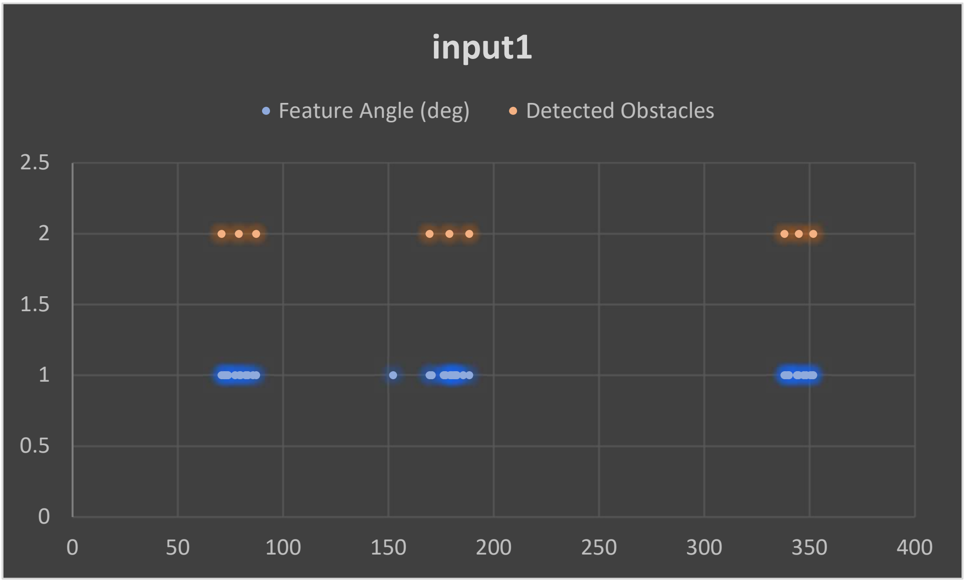
Below is a depiction of data from input0.txt in blue. Note there are outlier points that are ignored by the Obstacle Detector State Machine and that there is excellent correlation with Detected Obstacles (amber). This is with data analyzed with nominal parameters set at ten degrees of separation and minimum of three feature points to be identified as an obstacle. These values seemed reasonable and were chosen before looking at the KBRwyle supplied input data. No special tuning resulted in the results pictured.

Amber points provided by the Obstacle Detector are the left obstacle edge, the obstacle center and the right edge in degrees. Data taken from this input file is analyzed by the Obstacle Detector and captured along with this chart in the file, input0.xlsx.

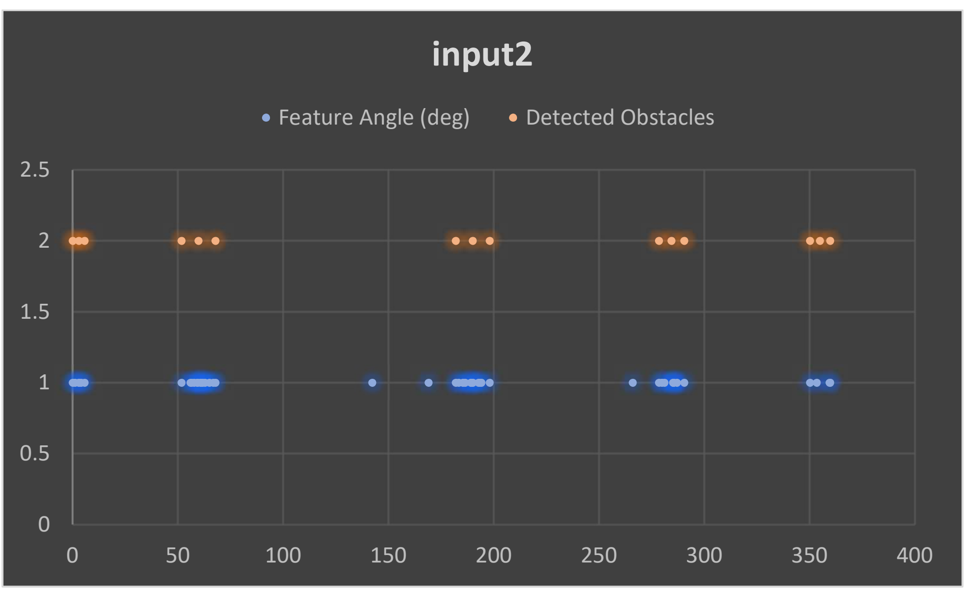



Similar spreadsheets were used to capture data for the rest of the input files presented below.

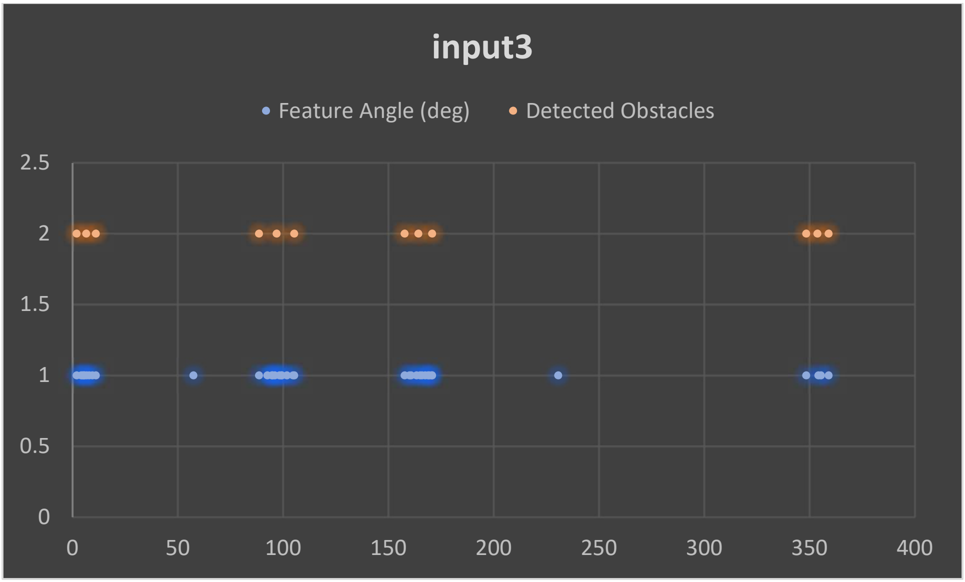
Results for input1.txt are similar. It’s tempting to say that the one outlier point is part of the second obstacle. However, there’s no way to really tell with the data supplied. It could be included if a larger outlier distance is used. Whether that would consistently yield improved results depends upon the obstacle environment. In general, the more this environment can be understood and characterized, the better one is able to “tune” parameters associated with the obstacle detector.



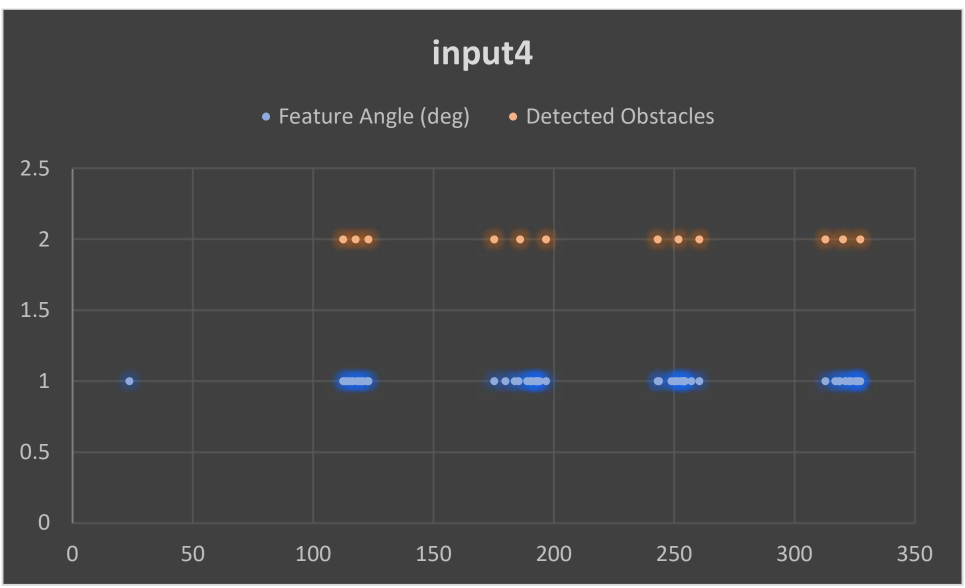

Results for input2.txt are similar. Now, there is some similarity between the third and fourth obstacles. And here, it’s even more tempting to include the outliers in the obstacles… particularly given the similarity. However, the current algorithm is not equipped to correlate one obstacle against another. Each one is a unique cluster of features as far as the utilized algorithm is concerned. This this points to a possible enhancement to the algorithm… however, complexity goes up very quickly and increases likelihood of unintended consequences in obstacle recognition.



For input3.txt results, a known deficiency in my algorithm can be observed. An object that spans values close to 360 and 0 degrees is depicted as two separate obstacles. For an aircraft that only goes forward, this would be fine. However, it is a deficiency for a system that could turn in any direction. Utility of consolidating this data would have be looked at in terms of the requirements for the system.




Results with input4.txt don't appear to yield any new information. The Obstacle Detector again, successfully tracks the data and identifies objects in the data presented to it.




**Conclusion:**

For datasets presented, the Obstacle Detector does its job. Certainly, for real world applications with more ambiguous datasets, more sophisticated algorithms and more attention to characterizing the obstacle environment and more associated parameter tuning will be necessary.