Unscented Kalman Filter Highway Project

Meets Specifications

Dear student,
Congratulations on passing the final project of the course. **You have put great effort in learning a new skill that is most critical in the robotics industry. You have demonstrated that you are now skilled enough to do advanced projects in the autonomous vehicle industry.** On behalf of everyone at Udacity, I wish you all the very best for your future endeavours. Hope this course will give you all the essential knowledge to excel in the robotics industry. Once again All the very best.

Here are some external links to further your learning process.

In real time project, you can implement your own UKF filter like you did now or you can refer to the following available packages which you can directly integrate with your code. One good thing about using available package is that some of the nuances or advanced optimization that you haven't learned in this course is already implemented and available to you for use.

- This matlab page explains the implementation of UKF using matlab.
- This ros package implements both the **Extended Kalman Filter** (EKF) and Unscented Kalman Filter (UKF), It is well documented here and the best part of this is you can straight away fuse and filter a lot of different sensors out-of-the-box.

Happy learning. Have a great day.

Cheers.

Compiling and Testing

The project code must compile without errors using `cmake` and `make`.

✅ The project code must compile without errors using cmake and make.

Feedback:
Well done👏. **Your project is able to compile with the `cmake` and `make` commands without any error.**

Pro Tips 💫:

- Here is a tutorial blog from the CMake Organization which covers all the important functions that we use in **`CMakeLists.txt`` file. Feel free to take a look at the file and learn more about how each component of the CMakeLists.txt file works.**
- I can see that you've submitted the build directory in your submission. There are two reasons why you should not submit your build directory.
  - The build file are system dependent meaning what works in your system will not work on others system. When you work with a larger team, using build directory from another system will cause a lot of confusions.
  - In this project the number of files are small and so the build directory size is also small. But usually on larger projects, the size of build directory is larger compared to the actual code. So when you push the codes to online sources like github will end up using more space that is required.

Code Efficiency

Your code does not need to sacrifice comprehension, stability, or robustness for speed. However, you should maintain good and efficient coding practices when writing your functions.

Here are some things to avoid. This is not a complete list, but there are a few examples of inefficiencies.

- Running the exact same calculation repeatedly when you can run it once, store the value and then reuse the value later.
- Loops that run too many times.
- Creating unnecessarily complex data structures when simpler structures work equivalently.
- Unnecessary control flow checks.

✅ Your code does not need to sacrifice comprehension, stability, or robustness for speed. However, you should maintain good and efficient coding practices when writing your functions.

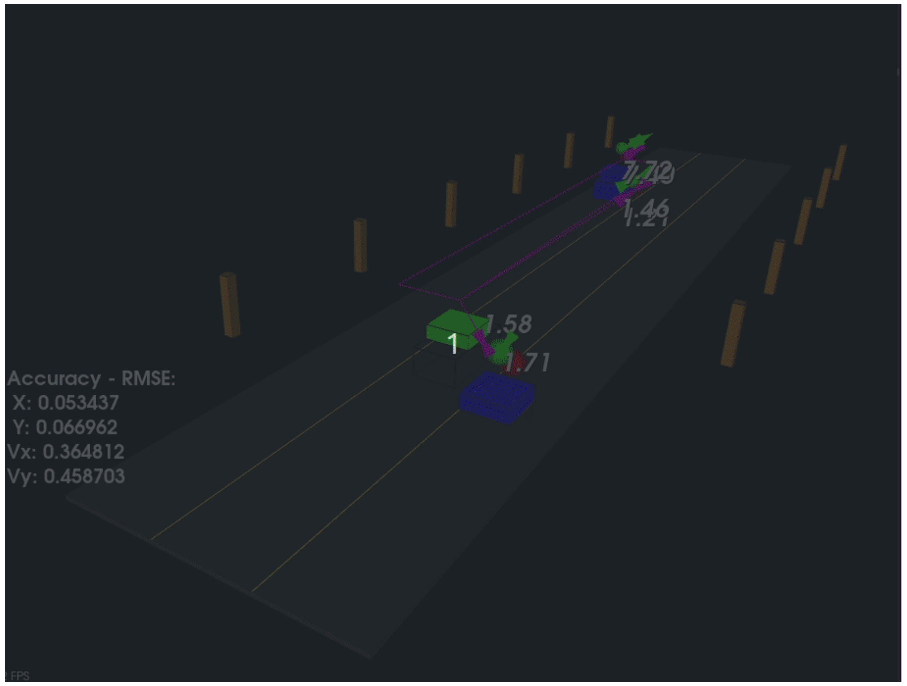
Feedback:
Great work👏. I must **appreciate you for writing such a clean code with inline comments wherever appropriate**. Inline commenting is a good practice since it will help you understand your own code when you come back to your code after a long break. Keep it up👏😃

Accuracy

The simulation collects the position and velocity values that your algorithm outputs and they are compare to the ground truth data. Your px, py, vx, and vy RMSE should be less than or equal to the values [0.30, 0.16, 0.95, 0.70] after the simulator has ran for longer than 1 second. The simulator will also display if RMSE values surpass the threshold.

✅ The simulation collects the position and velocity values that your algorithm outputs and they are compare to the ground truth data.

Feedback:
I was monitoring the **RMSE value and I wasn't able to see any error message popping up on the simulation**. This means that **you've done a great job** 👏👏 **with your code and standard deviation estimation**. The values you've chosen for `std_a_` & `std_yawdd_` as 2.0 and 1.0 are good enough to estimate the vehicle location is good accuracy. Nicely done😃👏



```
Accuracy - RMSE:
 X: 0.053437
 Y: 0.066962
Vx: 0.364812
Vy: 0.458703
```

Follows the Correct Algorithm

While you may be creative with your implementation, there is a well-defined set of steps that must take place in order to successfully build a Kalman Filter. As such, your project should follow the algorithm as described in the preceding lesson.

✅ While you may be creative with your implementation, there is a well-defined set of steps that must take place in order to successfully build a Kalman Filter. As such, your project should follow the algorithm as described in the preceding lesson.

Feedback:
Well done of implementing one of the highly used filtering and sensor fusion algorithm in the autonomous car industry👏. You must be proud of yourself for such a great feat of achievement.😃👏

DOWNLOAD PROJECT

Rate this review

START