Objection Detection Documentation

Minghan Li, Xuechi Xiang

May 17, 2022

Contents

Abstract

The documentation demostrates our ideas. It is not the final version as further details would be added and optimization might be applied in future.

1 Overall Process

1. LiDAR would scan through an object n times in one direction. Then we could obtain n arrays with distance information inside them.

2. Cut and fit these arrays to the same length.

3. Apply Kalman filtering algorithm to reduce the effect of noise on data.

4. Further analysis on the data would determine the type of object (Final Result).

2 Assumptions

1. The object would be scanned through one or two directions.

2. Each time the LiDAR scans, the deviation of angular displacement is negligible. $Range: 20° - 160°$.

3. The scanning speed (or angular speed) of LiDAR is fixed.

. Data from LiDAR(distance) and Gyro(angular physical quantities) after filtering are accurate enough to be further processed.

3 Detailed Process

3.1 Scanning

The scanning process is given and illustrated below:

1. Starts from a initial position (eg. $-45°$)
2. Laser sensor rotates horizontally and return data.
3. When it reaches horizontal final position(eg. $45°$), it would stop and start mext horizontal scan in opposite direction.

Problems might occur in this process is the rotation speed and initial/final angular position. These should be calibrated well before scanning.

3.2 Data Cutting and Fitting

Each times the laser scanner scans, the number of data would be different. Hence we need to cut and fit these data to be further processed. The procedure is given:

First, after the sensor scanned n times, n integar arrays would be obtained. We calculate average length of these arrays and use it as a reference length.

Then other arrays are compared to the reference. If the length of array is greater than reference value, the first and the last element of that array is removed alternatively until it reaches the same length as reference.

If the length of an array is smaller than reference value, we attach a value: 0 to the start and the end of that array until it reaches the same length as reference.

3.3 Kalman Filtering

3.3.1 Overview

The derivation process of Kalman filtering algorithm is complicated. Hence it is ignored and we just demostrate the algorithm itself

First we establish a mathematical model to simulate a real world problem. For example, a car moving on a road can be modelled by a rectlinear motion. The model can be quite simple or complicated. In this case, an error would raise when we want to use the model to perform calculations because the difference between model and real world situation. It is called: *Process Noise*. If the model is complicated and close to real situation, then the difference between them would decrease hence the error would be smaller. Conversely, if the model is too simple, error would boost.

Calculation results based on the model is called *Estimation* as a model would never be identical to real situation. Hence we need a corresponding value from real situation. It is called ***Measurement***. Measurement is obtained by sensors which reveals the real-time state of the object(eg. speed of a car measured by a speedometer). However, error still present in measured value which is called: ***Measurement Error***. The error would be affected by accuracy of sensors or methods of measuring a specfic quantity.

In conclusion, when we want to obtain the true state of an object and true values of some physical quantities. We must consider both the model and measurement as well as corresponding errors. This can be done by Kalman Filtering.

Generally, Kalman filtering algorithm would comprehensively utilize estimation (estimated value) and measurement (measured value) to make a optimal estimation. Hence it is essentially a recursive process. It consists of two main steps: Prediction and Update.

3.3.2 Prediction

$$\hat{x_{t^-}} = F\hat{x}_{t-1} + Bu_{t-1} \tag{1}$$

$$P_{t^-} = FP_{t-1}F^T + Q \tag{2}$$

where $\hat{x_t}$ is current prior estimation, $\hat{x_{t^-}}$ is the last prior estimation, F is state transfer matrix, B is control matrix, P_{t^-} is covariance matrix for last prior estimation, P_{t-1} is covariance matrix for last posteriori estimation, Q is variance of progress noise.

3.3.3 Update

$$\hat{x_t} = \hat{x}_{t^-} + g_k(Z_t - H\hat{x}_{t^-}) \tag{3}$$

$$g_k = \frac{P_{t^-}H^T}{Hp_{t^-}H^T + R} \tag{4}$$

$$P_t = (I - k_tH)P_{t^-} \tag{5}$$

where Z_t is current measurement, g_k is Kalman's gain, P_t is current posteriori estimation, I is unit matrix, H is system measurment coefficients matrix.

3.4 Object Detection (Machine Learning)

One possible way to achieve the determination of different objects is measuring the distance between some points on the surface of the object and a reference plane, then calculate gradients between these data points. Gradients indicates the smoothness of a surface, hence we could determine whether the surface of object is flat or curved.

A prerequisite of applying this method is that measurement error must be small. However, this is hard to achieve as traditional filtering algorithm cannot be implemented when determining an object. Hence a new method which is less effected by noise is needed for this problem.

We know that if we scan an object in one direction several times, raw datasets would follow the same trend(they have simliar shape) even if there are some variation presents between them. Hence a machine learning model can be applied.

Generally, a large number of data would be fed to a model, then a database can be established. When new data comes, it would be compared with each dataset in that database. Meanwhile, the difference between new data and recorded data would be calculated. New data would be assigned to a dataset if the difference between them is minimal. Finally, the object could be determined by checking the result of the matched dataset.

Two algorithms can implement the method: K-nearest neighbour and Naive Bayes classifier. We could compare the speed and accuracy of these two models and select one performs best. Details see Appendix ii.

4 Appendix i: Detailed Description of Kalman Filtering

4.1 Formula Interpretation

4.1.1 Prior Estimation

Equation (1) is called: prior estimation formula. We know that Kalman filtering algorithm is a recursive process, hence we need results from last cycle to calculate values needed in current cycle. The formula calculates prior estimation using optimal estimation from last cycle. F is the state transfer matrix which represent

4.1.2 Posteriori Estimation

4.1.3 Kalman gain

4.2 Code Implement

5 Appendix ii: Details of Machine Learning Algorithms

5.0.1 K-nearest neighbour algorithm

Provided a determined dataset(eg. a dataset of scanning a coffee cup, a glass bottle and a cereal box), all data dispersed in a plane and we know which single data point belongs. If a new data comes, we The object that new data belongs can be determined by checking maximum number of data in the range.

5.0.2 Naive Bayes classifier