Bayesian filter

This project aims at exploiting probabilistic properties of the data collected by the robot to run estimation on several topologic parameters such as the occupancy and the height of the floor, and build a model of the surface.

To make the estimation easier, we discretize the world as square and equal cells and run the required estimation on each cell independently. We aim at using every data from the past and the present to get the most accurate estimation. That is why Bayesian filters seem relevant in this context. We use Bayesian filter in two different contexts: first in a binary space to estimate the occupancy of a cell, second in a continuous space to estimate the height of the floor. We then build a regression model using two inputs compare the risk of each regression to define the amount of confidence we can put in the information contained in the elevation map:

- raw data points collected from points clouds all over the world
- estimated height of the elevation map

1 Binary bayesian filter on traversability classification

1.1 Environment

The robot lives in a 3D Vrep environment which floor is a surface contained in the hypercube $[5, 5] \times [5, 5] \times [0, 1]$. To reduce the continuous problem of where is the floor traversable to a simpler discreet problem, we divide the world in $CN \times CN$ cells and used a bayesien filter.

1.2 Probabilistic model

If X_t is the random variable see at the moment t we wish to estimate but cannot see directly; then we can try to compute $P(X_t \backslash U_1 ... U_t Z_1 ... Z_t)$ where U_t is the input of the system at the moment t and Z_t is the observed variable at the moment t. Then the bayesian filter affirms that :

$$P(X_t \backslash U_1 ... U_t Z_1 ... Z_t) = P(Z_t \backslash X_t) \int_{X_{t-1}} P(X_t \backslash X_{t-1}) P(X_{t-1}\ U_1 ... U_t Z_1 ... Z_t)$$

Then with the knowledge of $P(X_0)$ for each values of X_0 it becomes possible to evaluate X_t for every t. In our case $X_{t,i,j}$ is the random variable equal to 1 when the floor of coordinates i and j in our model observed at the moment t. U_t means nothing here and Z_t is the random variable that the point cloud see in i, j represent a traversable floor.

1.3 Implementation

Bayesian filter We implement the bayesian filter with the C++ Eigen library. The values of $P(X_{0,i,j} = \text{traversable})$ is equal to the value of $P(X_{0,i,j} = \text{nontraversable})$ is set at $\frac{1}{2}$. Without any information about the world, this distribution is the best one. To reduce the number of parameters we choose to set $P(Z_t = \text{traversable} \backslash X_t = \text{traversable})$ equal to $P(Z_t = \text{nontraversable} \backslash X_t = \text{nontraversable})$. For the same reason we set $P(X_t = \text{traversable} \backslash X_{t1} = \text{traversable}) = P(X_t = \text{nontraversable} \backslash X_{t1} = \text{nontraversable})$.

Callback Each time the robot receives a new point cloud from its sensor it runs the following steps:

- Point pre-processing
  - It drops points that are too far from it
  - It drops bogus points that are below the floor
- Point distribution
  - It stores each point in the corresponding cell : each cell has a list to store the points
  - It computes the coordinates of the plan thanks to a regression from those points.
- Bayesian filter processing
  - It calculates both probabilities of X t with the previously computed probabilities and stores the new ones instead of the old one.
- Result publication
  - Publish rather the place it is traversable or not by comparing the probabilities and considering the greater to be the correct one.

1.4 Parameter setting

xxt The probability for $P(X_t = \text{traversable} \backslash X_{t1} = \text{traversable})$ and $P(X_t = \text{nontraversable} \backslash X_{t1} = \text{nontraversable})$. It must be set over 0.5.

xz The probability for $P(Z_t = \text{traversable} \backslash X_t = \text{traversable})$ and $P(Z_t = \text{nontraversable} \backslash X_t = \text{nontraversable})$. It must be set over 0.5.

angle It is a symbolic representation of the angle between the plan calculated and the horizontal one. Here the value is used to major the coefficients a and b in $ax + by + c = z$. (if the plan is horizontal those coefficients are supposed to be equal to 0). We recommend an angle of 25.

Nb_{min} This number indicate the minimum number of points needed in a cell to compute the plan. This is function of the resolution chosen for the plan. For a resolution of 100, we recommend 10.

1.5 Observation

Naïve Bayesian Approach Since the floor is not moving as function of the time, let's xxt be equal to 1. This means that the observation is the only criteria to determine if the floor is traversable or not. Here is the result with nb_min equal to 5 on the left and 10 on the right.

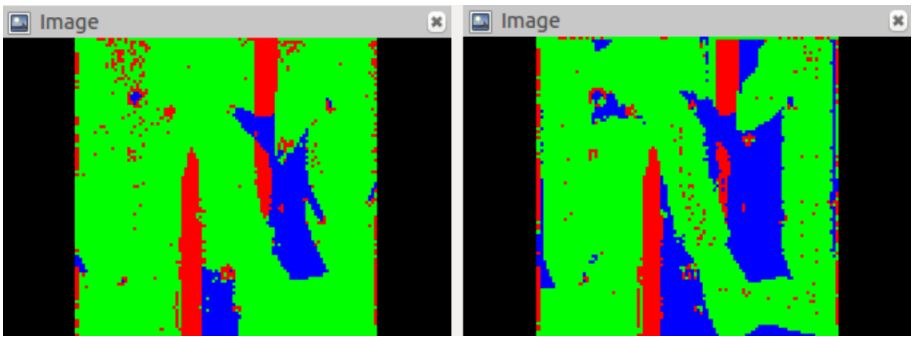


Figure 1: Height estimation: green/traversable, blue/non traversable, red/non observed

A first expected result is the surface covered is reduced when nb_{min} gets bigger. We can also see that some noise is present. An explication could be that the last observation of those case, while having a number of point sufficient still returned some wrong results. If this hypothesis is true, than another value of xxt should solved the problem.

Not Naïve Bayesian Approach

Here we will try to use a xxt of 0.8 and the same nb$_m in 5 and 10$

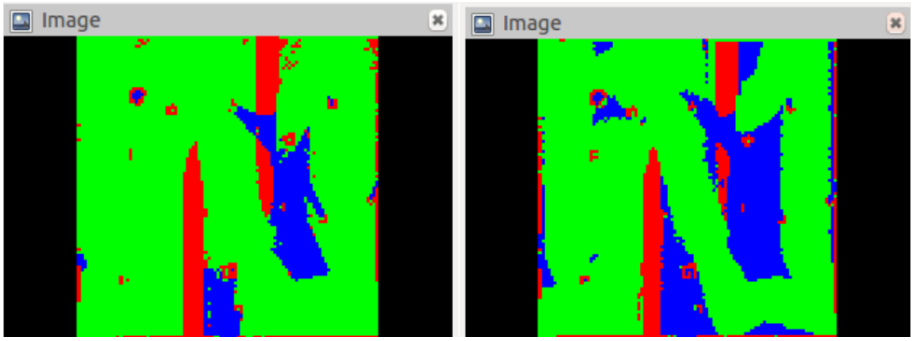


Figure 2: Height estimation: green/traversable, red/non traversable, blue/non observed

Here we can clearly see that a hugh amount of error is gone and confirm that our hypothesis was good. The remaining amount of noise in the case $nb_{max} = 5$ can be explain by the fact that the amount of observations returning false when it is true is high than with $nb_{max} = 10$. The coverage is however far more important so we need to compromised between a precise but uncomplete mapping and a more complete one but with a bit of noise. However we can be assured that the Naïve Bayesian approach is not the good one to go for this problem

Ransac We also tried to used the ransac method for the observations.

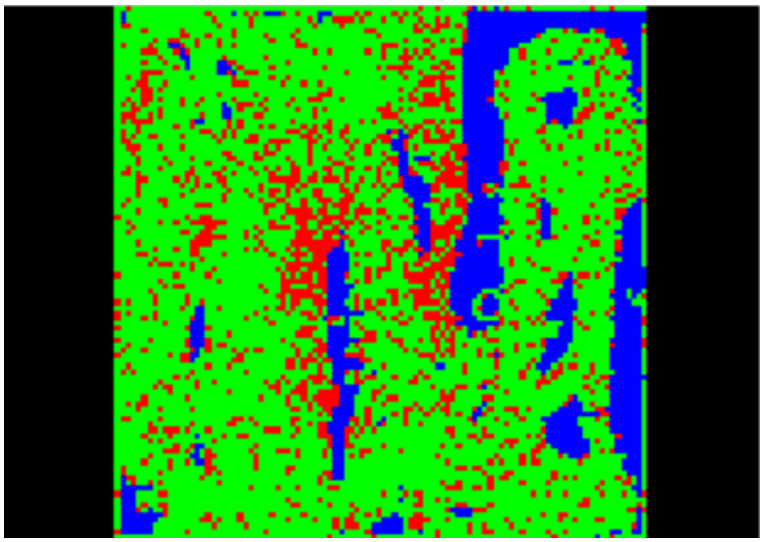


Figure 3: Height estimation: green/traversable, red/non traversable, blue/non observed

However, the result was way less efficient that with the regression method, both in term of resukts and computational time.

2 Kalman filter for elevation map

2.1 Environment

The robot lives in a 3D Vrep environment which floor is a surface contained in the hypercube $[5,5] \times [5,5] \times [0,1]$. We estimate the height of the floor by estimating its height locally. To do so we divide the world in $CN \times CN$ cells and run the Kalman filter described below.

2.2 Probabilistic Model

Given a cell, it can only have one correct height so we can model the distribution of the floor height as a unimodal distribution. We will choose the Gaussian distribution to make calculations easier. This way, we can use a specific Bayesian filter: the Kalman filter. We now have to specify the model.

Notations for each cell:

- x_t : estimated height of the cell after t observations, x_t is real
- z_t t^{th} observation i.e. point cloud in this cell

We define the model and evolution equation as follow:

$$x_t = x_{t-1}A_t + R_t \tag{1}$$
$$z_t = x_t C_t + \delta_t \tag{2}$$

Equation (1): Since the environment does not change with time, the only quantity that changes with time is the amount of information collected in each cell. This means that we can assume $x_t x_{t1}$ and that is why we assume $A_t = 1$ for all t. There still exist the model uncertainty modelled by R_t and that we will set experimentally. We will assume that this uncertainty does not change with time since the environment is invariant.

Equation (2): Regarding the observation equation, there is a linear relation between the observed point's height and the height of the cell. To make sure that we measure the cell height and not another height, we will pre-process the observation by selecting only the points near the floor. Then z_t is a column vector that contains the 3^{rd} coordinate of the pre-processed points.

Given x_t the height of the floor, we make the assumption that z_t is vector whose values are distributed around the floor with a variance that is modelled by δ_t . So we set C_t to be the vector column of size L made of 1 where L is the number of points in the observed point cloud. For the moment we assume that L does not need to be set by the model or by the environment and may be different from one cell to another. We will set the value of δ_t by measuring the variance of observations around the floor height with training observation. With all these assumption our model becomes

$$x_t = x_{t-1} + r$$
$$z_t = x_t C_t + \delta_t$$

With r, δ_t parameters to set. We keep C_t and δ_t indexed by t because even though the numerical value of their element will not change since the environment does not change with time, their size may change since we did not decide if the size observation L must be set or not. Still we assume that C_t and δ_t are respectively a column vector of size L whose elements are 1 and a multivariate Gaussian which covariance matrix is square diagonal matrix whose diagonal elements are all equal to q. This diagonal form comes from the fact that we assume that the noise is a white noise that is independent for each observed point.

2.3 Implementation

Kalman filter We implement the Kalman filter equations via the Eigen library and using the following algorithm:

Table 1: Average estimation error

Prediction phase	Update phase
$x_t^* = ax_{t-1}$ $\Sigma_t^* = \Sigma_{t-1}^* a + r$ $K_t = \Sigma_t^* C_t^\top (C_t \Sigma_t^* C_t^\top + Q_t)^{-1}$	$x_t = x_t^* + K_t(z_t - C_t x_t^*)$ $\Sigma_t = (1 - K_t C_t)\Sigma_t^*$

Callback Each time the robot receives a new point cloud from its sensor it runs the following steps:

- Point pre-processing
  - it drops points that are too far from it
  - it drops points that are too high from the floor : $z > up_{bound}$
  - it drops bogus points that are below the floor
- Point distribution
  - Store each point in the corresponding cell : each cell has a list to store the points
  - Compute the average height of the point cloud in this cell after the distribution. This average will be the reference to measure the accuracy of the filter.
- Kalman filter processing
  - Once a cell has collected "enough" point, it process them through the filter to estimate a new average and variance of the floor height distribution.
- Result publication
  - Publish the estimated height, the variance/confidence and error of the estimation.

Parameter setting

L At first glance, there did not seem to be reason for L to go under constraint from the probabilistic point of view. However from a time complexity point of view, we can not let L unbounded because it then leads to a filter processing that takes more time than the data collection period T_d . Still to get observation as rich as possible, we test several values of L, measure the average period of a callback T_c and take the maximum L for which $T_c < T_d$. We measure $T_d = \frac{1}{60}s$ and plot the several measured T_c.

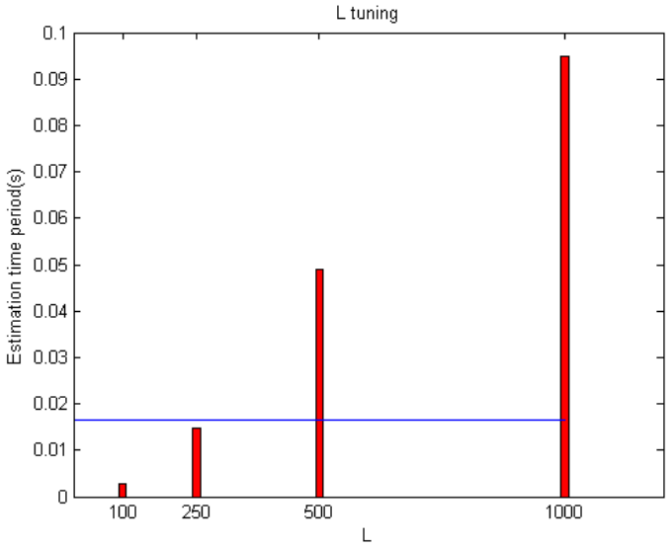


Figure 4: Height estimation: green/traversable, blue/non traversable, red/non observed

We compute the average time period of the callback loop, that is the processing and estimation loop for different size of observation L. We compare them to the time period of observation, that is the frequency at which vrep sends us a point cloud in blue. We see that L is then limited by computational time constraints : the bigger L, the more time is takes to compute inverse and the more lag we will get in the estimation; To avoid this we could set $L_{max} = 250$. Especially when the graph below shows that the error decreases with the observation size. This error is measured by the difference between the height estimation and the average of the observed heights. The average is a good estimation of the ground height because we only took points near the floor.

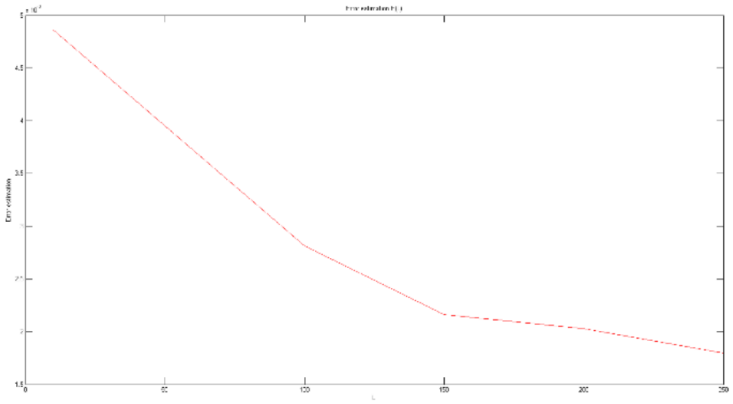
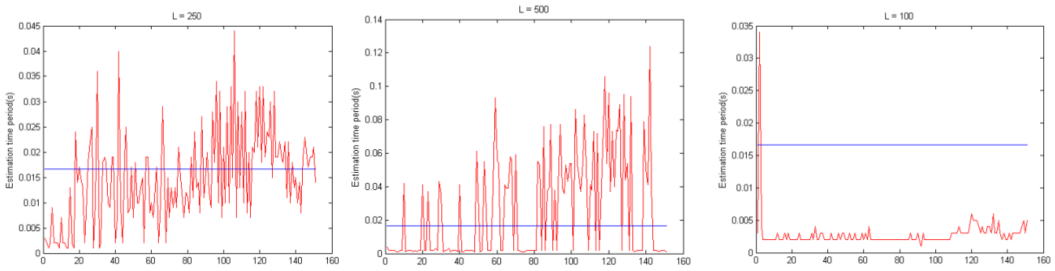


Figure 5: Height estimation: green/traversable, blue/non traversable, red/non observed

Still, when we look at the variations of the callback time execution, we see that it has a strong variance. This is due to the fact that many callbacks occur whith cells that still didn't collect enough points to run estimation, and many cells reach the required number of observations L at the same time. That is the callback has to perform costly operations many time at once. We now proceed to a greater comparison of the callback period compared to the observation period and we see that when $L = 250$, they are not synchronous for half of the time. This may not be acceptable and that is why we choose a lower L given that the error stays in the same order of 10^{-6} .



(a) L=250 (b) L=500 (c) L=100

Figure 6: Evolution of the error with respect to the regulizer λ, $maxIter = 10$, Model complexity legend : (red,1), (blue,3), (green,5)

Given these experimental results, we set an upper bound for L : $L_{max} = 250$.

Q We record the third coordinate of all the useful points all over the world given the pre-processing described above. Such a data cloud is typically the distribution of point cloud we will use in the filtering and according to the pre-processing, the values are constrained around the height floor. We measure the variance of these data and assume it is a good estimation of $q = 0.017261407$.

x_0 We must feed the filter with an initial value of x_0 . Given the topology of the experimental environment, there are almost as many cell whose floor height is 0 as cell whose floor height is 1. We model this unknown by setting x_0 to 0,5. This means we do not include any knowledge in the filter.

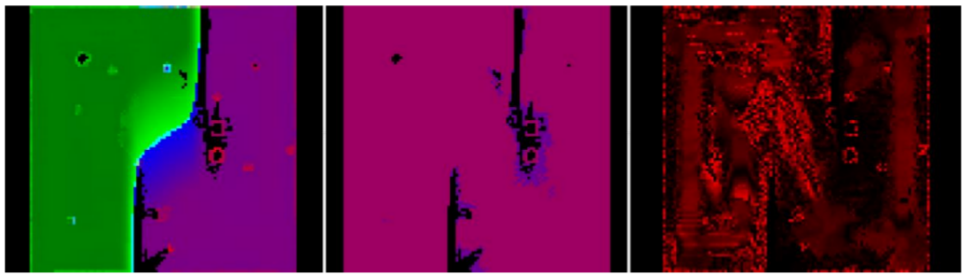
2.4 Observations

Noise Measurement On the lowest part of the environment where the height of the cells are approximately 0, a lot a data points if not the majority are below the ground. A first attempt to drop these points led to an unacceptable wholes in our estimations. The bluer, the nearest to 0 / the purplish, the nearer to 1.

ations. The bluer, the nearest to 0 / the purplish, the nearer to

Figure 7: Height estimation: green/traversable, blue/non traversable, red/non observed

That is why we decided to still consider these points and run specific processing on heights that ended up negative. In such cells it means that the floor is at altitude 0. To measure the range of this phenomenon, we represent the height of these particular cells in a gradient of green. The greener it is, the nearest to 0 it is. For regular cells, we keep the previous gradients and we get the following result in the left image.



	Negative		Low		Low
	Near 0		High		High
	Positive				

Figure 8: Height estimation: green/traversable, blue/non traversable, red/non observed

Estimation uncertainty We interpret the variance of the estimated height as the uncertainty we have on the estimation. We can observe its variation between $1.69685 \cdot 10^{-4}$ and $1.69733 \cdot 10^{-4}$. The pinkest it is, the higher the covariance and the more uncertainty we have. It first appeared counter intuitive but we can explain this way: the whole estimation increases the uncertainty globally because the correction step does not decrease the variance as much as the prediction step increases it. This is due to the high absolute value that we assigned to the measurement noise $r = 0.01$ to make our model flexible. We also notice that the variance is extremely low but this is not enough to conclude that the estimation is relevant. We must also observe the error value.

Error estimation We observe the error of the estimation compared to the average height observed. Given that this average height is computed on data points specifically taken near the floor, we assume it is good reference against which to compare the estimation. During the experiment running, we observe that the more data point the robot gets for a cell, the less error it has which is intuitive. The darkest it is, the less error there is. The average value of error is in the order of 10^{-6} which is far more than acceptable. However, the extremely low uncertainty and error may presage overfitting of the filter for this environment.

Obstacle detection The elevation map highlight the obstacle when computing discontinuous heights with respect to neighbouring cells. This can be observed in the map through the lighter irregularities which corresponds to cylinder, flowers or rectangles. The higher the map discretization, the more accurate the detection is.

Conclusion

We have experimented the relevancy of Bayesian filter in two context :

- State estimation in a binary context
- Height estimation in a gaussian context

In both cases, we have used quite simple model compared to the complexity of the real environment but still got consistent result in terms of errors and uncertainty. This is due to recurrent updates of estimation through batch of new observations. This shows the strength of the Bayesian filter that allows this adjustment “synchronously” with the new data collection. However we must stay conscious that we run these experiment in a very smooth environment and got rid of any possible outlier. We expect to get less relevant result in a noisy environment especially for the Kalman filter because the correction step treats equally all the observation points.