Ground plane detection using Hough Trasform and RANSAC

We work on finding the ground plane in a 3D point cloud provided by the simulated kinect on the V-REP robot. We compare two methods: Hough Trasform and RANSAC.

1 Hough transform

The goal is to use the general principles of the Hough Transform to find likely plane parameters from the point cloud and in particular the ground plane. The Hough Transform relies on a nD accumulator. This is provided in the accumulator member variable, which is implemented as a 3D OpenCV matrix.

Range discretization In the space, a plane is defined by three parameters (a, b, c) with the equation: $z = ax + by + c$. So for each point (x, y, z), there is plane created in the parameters space (a, b, c) : that is why our accumulator is setup with 3 dimensions. Our accumulator is the discretization of the 3D space $[a_{min}, a_{max}][b_{min}, b_{max}][c_{min}, c_{max}]$.

Range choice We define the range based on the range on the camera sensor. In our experiment, we set the camera `max_range=5m` from the coordinate origin with the base of the robot. We already know that we want to estimate the ground plane and then that the accumulator should have its maximum values around the first indices of row, column and depths of the accumulator. It is then wiser to shift $[a_{min}, a_{max}][b_{min}, b_{max}][c_{min}, c_{max}]$ around the origin of the space :

- To be able to detect bogus points which would have negative altitude (bogus can come from noise on measurements).
- To reduce the computational complexity when finding the accumulator maximum by setting c_{min} and c_{max} to -0.5 and 0.5. It does have consequences on the accuracy of our estimation because any points above the plane $z = 0.5$ is an outlier.

Discretization choice We want a optimum discretization that allows a precise estimation of the ground plane with the minimum computations: the bigger is the discretization, the less are the computations but the less precise the model is and vice versa. We define the minimum discretization required based on try-and-error method.

Error minimization First we observe the error between the estimated and real plane with respect to the discretization range in the case where the robot in on plane ground in front of a an obstacle 1.

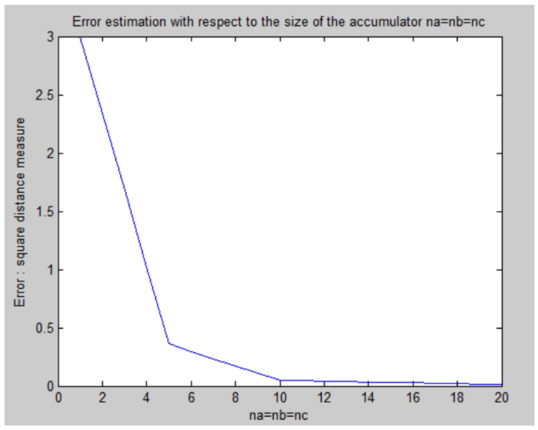

Figure 1: Error with respect to the discretization range

We chose this setting because running these observation without an obstacle did not show enough differences in the error as the discretization range lowers. We observe that the bigger discretization range we take, the bigger is the error as it would be expected. We want to highlight that from $n_a = n_b = n_c = 4$, the error becomes acceptable. Conclusion 1 : To get an acceptable error, each axis space observation must be discreted over at least four point.

Computation load minimization We observe the evolution of the computation time in seconds with respect to the range discretization (Figure 2).

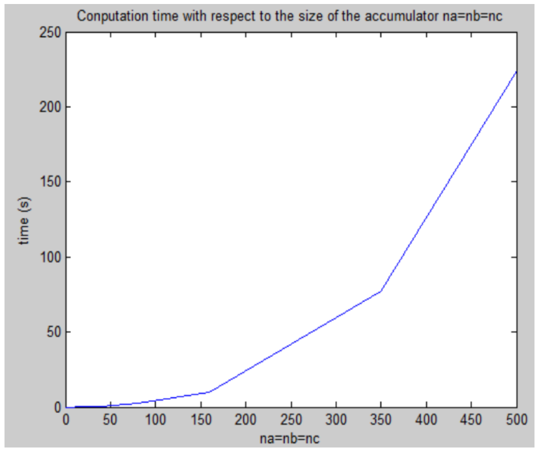

Figure 2: Computation time with respect to the discretization range

Computation time increases exponentially with respect to the range. This curve materializes the down side of the Hough transform on heavy models. We must be very careful to minimize the choice of parameter. Conclusion : The choice of $n_a = n_b = n_c = 4$ is leads to a computation time of $0.011s$. Our choice passes the test of the computation load.

Discretization ranges correlations We already know that we want to estimate the ground plane so the most important feature point to study is its altitude. That is why, we want to privilege a high discretization of z compared to the x and y discretization. We plot the variation of the error with respect to the discretization range on ground axes (x, y) when vertical range discretization $n_c = 20$

(red) and with respect to the discretization range on vertical axis z when ground range discretization $n_a = 20 = n_b$ (blue) 3.

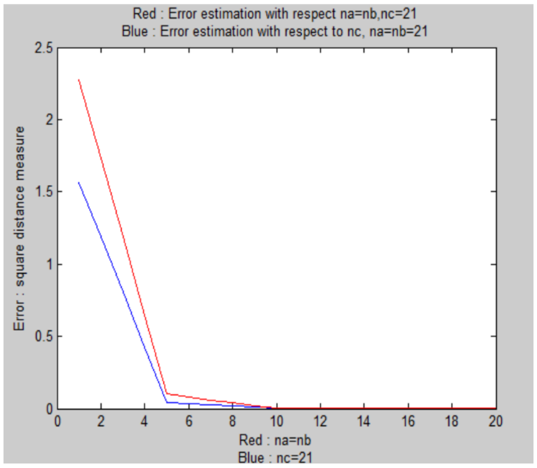

Figure 3: Computation time with respect to the discretization range

Then each plot shows the weight given to error on ground axis and vertical axis. We confirm our intuition by observing that the error is more sensitive to vertical discretization than the horizontal one. We see that variation on nc have almost as many impact as the variation of both parameters n_a and n_b. Incresing nc increases the precision of the discretization linearly while increasing na and nb increases it squarely. Still the error does not follow the same rule since it grows linearly to both (na,nb) and nc at the same rate. Conclusion : For a given model complexity $n_a + n_b + n_c$, it is better to set an relatively high vertical discretization at the price of a lower ground disceretization.

Noisy environment We now know how to chose the discretisation parameters to optimize the plan estimation. There is another point that could be improved regarding the algorithm which is the algorithm itself. This optimization can be done only once a certain number of iterations have been done. Given a point (x, y, z) in the data space defining a set of (a, b, c) in the parameter space, instead of upgrading all the corresponding (a, b, c), we would upgrade only the one for which the accumulator is the maximum over this particular (a, b, c) and its neighbors in a certain distance. If the corresponding (a, b, c) accumulator is not the maximum, we increment instead the accumulator which present the highest value in the neighborhood. In harsh environment, this would allow us to use bogus point smartly to get the right plane estimation

2 RANSAC

We randomaly sample 3 points to compute a plan $\mathcal{P}$ and then compute how many other data points of the data set belongs to this plan. Given the discrete simulation environment and the measurement noise, we say that a point A belongs to $\mathcal{P}$ if $d(A, \mathcal{P}) < ⊔≀↕⌉∇⊣\⌋⌉$.

We repeat this protocol "enough" times until we get a plan $calP$ that holds enough data points. The more it is runned, the better the plane parameter estimation but the computations take more time (Figure 4).

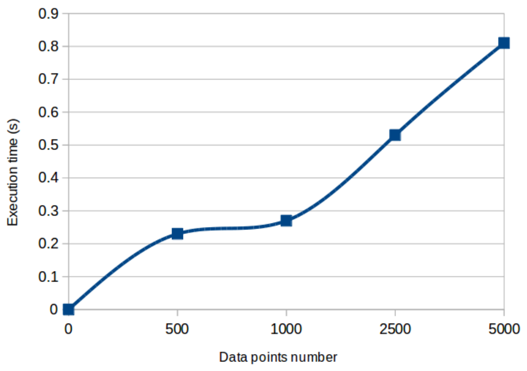

Figure 4: Computation time with respect to number of protocol execution

Since the robot moves during the computations, we have to limit the execution time so that the robot can detect conitguous part of the ground. We choose to repeat the protocol 1000 times because it is the maximum value before the computation time seems to exponentially increase.

Regarding *tolerance*, a low value ensures an accurate estimation given that the points used in the estimation are the nearest to the computed plan $\mathcal{P}$. But there is a risk that there is not enough point if the point cloud is too sparse and the *tolerance* too low. A relevant value would be the noise measurement n_m: the distance between the points belonging to the ground plane and the ground plane is in average the noise measurement. So if we include all the points n_m-distant of the plan to detect, we are sure that we have detected the plan to detect. We make the robot complete a round of the world and each time the sensor samples a 3D point cloud, we sort the points only to keep the ones on the ground and compute their variance. We average the results over the number of computation made in one round. Given that the world is discrete, we take the points which altitude is $z = 0 + \epsilon$ with $\epsilon = 0.01m$. We deduce the value $tolerance = 0.2$.

Conclusion Apart of the programming part, we have faced the challenge on parameters setting for both of this techniques. Both are proved to be effective but the accuracy of the result come at the price either of computational time or trial-and-error time to find the optimum parameters. Each method has it pros and cons :

- The Hough method computational time increases exponentially with the model complexity which fastly limits its usage in practical cases. However, the accuracy of the result may be worth it once we are able to reduce the complexity. This implies prior knowledge on the estimation which we may not have.
- The RANSAC is less hungry is terms of computation time but allows to get pre-estimation of the model by adjusting the sensitivity by trial-and-error even if we lack information about it. However we must always be aware that the result is not always the right.