2D localization of outdoor mobile robots using 3D laser range data

Takeshi Takahashi

CMU-RI-TR-07-11

May 2007

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Submitted in partial fulfillment of the requirements for the degree of Master of Science

Abstract

Robot localization in outdoor environments is a challenging problem because of unstructured terrains. Ladars that are not horizontally attached have benefits for detecting obstacles but are not suitable for some localization algorithms used for indoor robots, which have horizontally fixed ladars. The data obtained from tilted ladars are 3D while these from non-tilted ladars are 2D. We present a 2D localization approach for these non-horizontally attached ladars. This algorithm combines 2D particle filter localization with a 3D perception system. We localize the vehicle by comparing a local map with a previously known map. These maps are created by converting 3D data into 2D data. Experimental results show that our approach is able to utilize the benefits of 3D data and 2D maps to efficiently overcome the problems of outdoor environments.

Adviser: Sanjiv Singh

Contents

1 Introduction

Localization is very important for autonomous vehicle to track a path, detect and avoid obstacles properly. Indoor robot localization has been done successfully,however outdoor robot localization is still a challenging problem. In outdoor environments, there are factors that make localization difficult. Terrains are usually not flat. GPS signals degrade in the presence of vegetation, buildings, and terrain features. Environments are being changed because of the presence of people, cars, and other moving objects. Terrains, especially trees, are different in seasons. Construction of roads and buildings also change the environment. Therefore we need a robust localization method that overcomes these problems.
Robots performs many tasks in real time, such as obstacle detection, obstacle avoidance, path planning , path tracking, and localization for real-time operations, and therefore we need an algorithm that does not require high computational costs. Using cameras results in high computational costs and they are not suitable for night operations, while using ladars results in lower computational cost and they are applicable to night operations.
It is hard to use indoor robot localization algorithm because of the property differences between indoor and outdoor environments. Indoor robots usually have horizontally fixed ladars that can obtain data at a certain height and are enough to localize the vehicle because of no elevation and the presence of walls. However, if robots have horizontally fixed ladars, it is difficult to detect obstacles lower or higher than the ladar.
Tilted ladars are able to detect obstacles efficiently, but the vehicle drives at a high speed, and scenes taken from ladars at the same area could be different because the data density is very low. Although we are able to use GPS in outdoors, we cannot always rely on GPS information because of satellite occlusion. Lost GPS information sometimes causes significant errors.
In real-world applications, the use of a known map for localization is a legitimate assumption. For example, security robots are different from reconnaissance robots, which sometimes needs to be operated in an unknown area. Security robots can be operated in factories, port and harbors, and military bases. This means that they are used in known areas and therefore,we do not have to solve SLAM (Simultaneous Localization and Mapping) problems for this kind of security robots.
In this paper, we compare localization methods and map representations, and present a suitable localization method for our outdoor security robot. This paper consists of the following sections: data acquisition (Section 3), map creation (Section 4), and localization algorithm (Section 5). Section 6 describes the results of our experiments in the outdoor environment.

2 Related work

In this section,we review related works in localization algorithms. Particle filters have been one of the most reliable localization algorithms particularly for indoor environments [3] [4]. One problem with particle filters is the computational cost involved. The more the number of particles and state spaces we use, the more the computational

(a) (b)

Figure 1: Our security robots(a)Grizzly and (b)Yogi

cost. Accurate results need large number of particles and state spaces. This is a trade-off between accuracy and computational cost. In order to reduce computational costs, particle filters for real time operations have been studied under changing number of particles and size of incoming data set [5] [6].
Another problem with the particle filter is that the robot cannot recover its pose when no particle is around the true pose in the sace of an accident. Sensor resettings are able to solve this problem [7]. Expansion resettings are also able to overcome this problem and a combination of these two resetting methods can be applied to robust localization algorithms [8].
Studies mentioned above are for indoor robots with horizontally fixed 2D laser sensors. There are the problems with the use of these sensors in outdoor environments [1]. It is difficult for the 2D navigation system to overcome unflat environments [14] and to detect and avoid obstacles; however, while tilted ladars are suitable for these purposes [12].
Histogram Matching can be applied to 3D SLAM [9]. This algorithm can successfully map very large areas and solve the kidnapped robot problem. It does not use less reliable matches. This idea can be applied to matching in the localization algorithm. However, this algorithm cannot be used for global localization. The situation is different from ours because this vehicle has horizontally fixed ladars.
Although 3D outdoor localization algorithm using ladars has been studied, the computational cost involved is extremely high due to the high degree of freedom of the states [10] [11]. Other approaches for outdoor navigation are described in [1]. This system use 3D data for a 2D SLAM algorithm and map representation for semi structured outdoor environments. It has the advantage of 3D perception and less computational 2D navigation. In this paper, we describe how to convert 3D data into 2D representation and how to exploit these advantages.

3 Data acquisition

The ladars we use are 2D laser range finders. One ladar is fixed to the vehicle and is slightly tilted to cover a range of 10 m in front of the vehicle. As the vehicle moves, we can obtain horizontal data. We found that one sensor is not enough to detect obstacles and to avoid these obstacles when the vehicle moves at a high speed [12]. Therefore, we added another sensor that is sweeps back and forth to obtain vertical data. We can obtain 3D data by combining these two sensors.
The ladars need to be well calibrated, because the data difference between fixed and sweeping laser sensors generates false obstacles, which causes serious error in obstacle avoidance.

We have a gyro that give the yaw angle of the vehicle and an encoder that provides the distance of travel. We transform data obtained from ladars into the vehicle frame, and then transform them into a global coordinate using odometry information or GPS information.

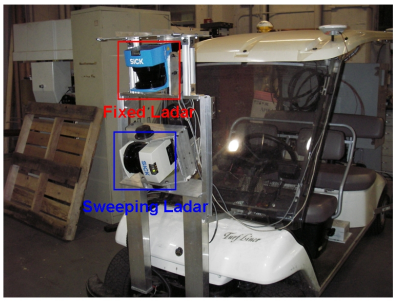
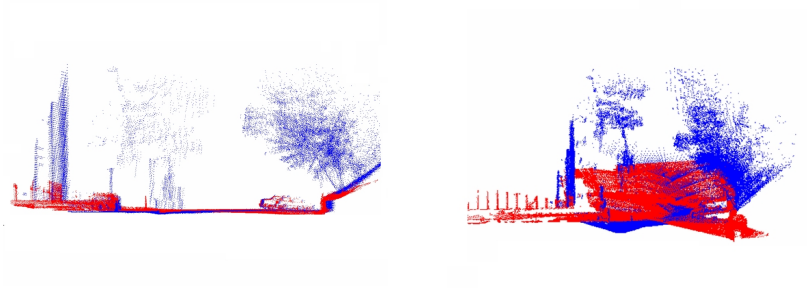


Figure 2: Sweeping and fixed ladars



(a) (b)

Figure 3: (a)(b) 3D point clouds from sweeping (blue) and fixed ladars (red)

4 Map matching

When the robot is operated in an unknown area, we need to create a map of the area. SLAM solves this problem if we can not use GPS. In our situation, we don not consider the computational time for creating a map and use GPS to create a map when GPS signal perception is good. The important point is that we need to create similar maps in order to compare a known map with a local map created during a task. A large difference between the known and the local map will cause serious problems in localization. Dynamic obstacles is one of the main factors that causes the difference in the maps. When the vehicle moves at a high speed, it cannot obtain the density data of environments. Our vehicle is supposed to move at a speed of 5 m/s. Therefore, we need to overcome the dynamic obstacles and the fewness of data points. Because our operation is done on-line, we need to quickly create a local map. There are many methods of creating a map. To create a 3D map or store 3D data points, we need large sources and computational time. Therefore, we use a 2D map that can utilize the features of the 3D data. We create an elevation map in which we discretize the world and each grid has a value representing its highest point, traversable costs or landmarks etc. Finding vegetation takes time to compute, but if we focus on only few features, we can compute in real time. Taking the highest point in a cell might be noisy. In our case, we select a point below which p percent of data points in the cell are included. Drawbacks of the elevation map are that we cannot represent some typical terrains, such as overhang obstacles, trees, tunnels, and roads under bridges. We need to consider these obstacles if the area where the robot is being operated has many of these features. In our case, we conducted experiments in the campus where there are no such objects. We believe that all environments do not have these many features. Partial problem caused by these features in our algorithm is insignificant fecause of its robustness. We fix the vehicle's height to ground level because our gyro cannot measure the pitch of the vehicle and the vehicle is always on the ground.
We suggest three simple map representations: an elevation map known as 2 and a half dimensional map, a feature map, and landmark maps. We compare these representations in the experimental section and describe the results.

4.1 Known map

When we create a known map, we can use any of the methods and devices like beacons and GPS. If we have several robots and only one of them has GPS, we can use the robot to create a known map. Other robots do not need to have GPS, which can reduce the cost involved. In our case, we create a map using GPS and manually remove noises and obtain a decent map.
Due to the property of our gyro, we can measure only the yaw angle of the vehicle. Thus, we cannot consider the pitch and roll. We do not consider the cases where there are overlapped roads and many hills.
The direction of movement of the vehicle when we collect data is important. In particular, because our vehicle moves at a high speed. The data obtained from different directions would be different. Therefore, we need to collect data in at least two opposite directions. Then, we combine these data to create a known map. Our odometry

doesn't provide pitch angles, therefre when we created a prior map, we don't use pitch angles although GPS provides pitch angles of the vehicle. If we use pitch angles to create a known map, slopes in the map would be different from theses in a local map, and that leads to bad matchings. Not using pitch angles causes problems in loclaization when the vehicle starts to go up or down the slope and finishes to go up or down the hill. These problems occurs for only few seconds, thus it doesn't affect localization so much as shown in the result section.

4.2 Local map

The local map is a map created using just ladars and odometry, and not GPS. A gyro and an encoder provide accurate data when the vehicle moves a small distance. Thus, the local map is supposed to be almost the same as the sub map, which is extracted from a known map. We can obtain 3D data points from ladars, but due to the high computational cost invovled, we convert them into 2D data.

4.3 Elevation map

The simplest method of creating a 2D map from 3D data is to compute the elevation of each cell. First, we decide the proper bin size β, which is usually 50 cm or 25 cm. The finer map requires more computational cost. We need to collect data at least in two opposite directions so that we can localize the vehicle in any direction. There are a variety of methods of creating an elevation map. First, we assign a cell $\{i, j\}$ to each 3D data point $p_n = \left\{p_{n_x}, p_{n_y}, p_{n_z}\right\}$ collected from ladars. We assign the cell as follows;
$i = \left\lceil \frac{p_{n_x}}{\beta} \right\rceil$
$j = \left\lceil \frac{p_{n_y}}{\beta} \right\rceil$
$p_n \Longrightarrow p_{k,\{i,j\}}$
where the function $\lceil x \rceil$ gives the smallest integer $\geq x$.
A cell $\{i, j\}$ has K points. After we assign the cells to points, we can compute the elevation of a cell.

4.3.1 The Highest point

$m_{ij} = max_k \left\{p_{k_z,\{i,j\}}\right\}$
This elevation map is very noisy because the highest point does not represent the height of the object. When the robot obtains the highest point near the object, the height would be low, while it would be high when the robot is far away from the object.

4.3.2 Mean of data points

$m_{ij} = mean_k \left\{p_{k_z,\{i,j\}}\right\}$
This cannot represent a map properly because the density of data points is high at a low level and low at a high level due to the tilted ladars.

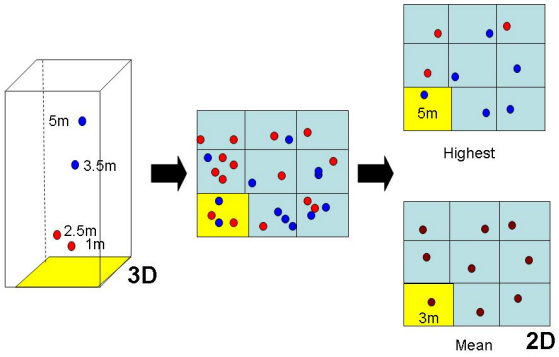


Figure 4: Elevation map using highest points or mean of points

4.3.3 Number of points

We count the number of data points in each voxel i, j, l. Then, we use the voxel with the maximum number of points as the elevation of the cell. In our case, this method cannot be applied because we obtain more data points on lower parts of objects than upper parts because of the fact that the ladars are tilted.
$l = \left\lceil \frac{p_{k_z}}{\beta} \right\rceil$
$p_n \Longrightarrow \{i, j, l\}$
A voxel $\{i, j, l\}$ has points $P_{k,q}$
$m_{ij} = argmax \left\{ countpoint_l \left\{ i, j, l \right\} \right\}$

4.3.4 Robust elevation map

We want to select a point below which α percent of points in the cell are included. If we use this method, the resulting map has less noise than the map created using the highest points. This method is very simple to use and more robust than the others; therefore we adopt this map as the elevation map.
$P_k = descend\ sort_k \left\{ p_{k_z, \{i,j\}} \right\}$
$m_{ij} = p_{\lceil \alpha * K \rceil}$
where $\{i, j\}$ has K points.

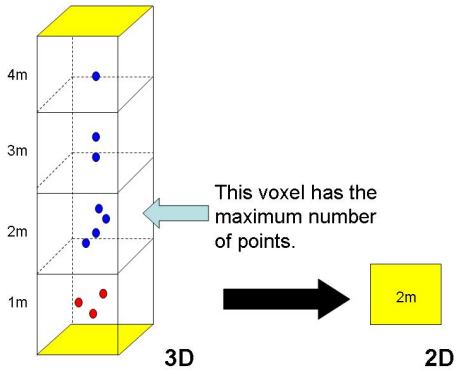


Figure 5: Creation of an elevation map using a voxel with the maximum number of points

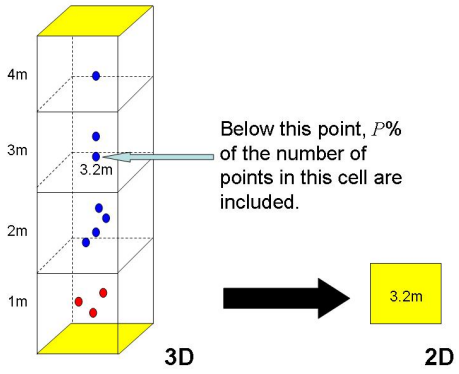


Figure 6: Creation of an elevation map with a point below which p percent of points are included

4.4 Feature map

People and cars are the main moving obstacles. Most of the people and cars are less than 2 m in length. Therefore, we consider a space that is greater than 2 m. Then we create an elevation map. We want to use these data as features. Thus, we select a space such as, 2 - 3 m, 2 - 4 m or 3 - 4m. Empirically, the space from 2 to 3 m above the ground would be suitable for features as shown in Fig.**??**. If we use the upper space

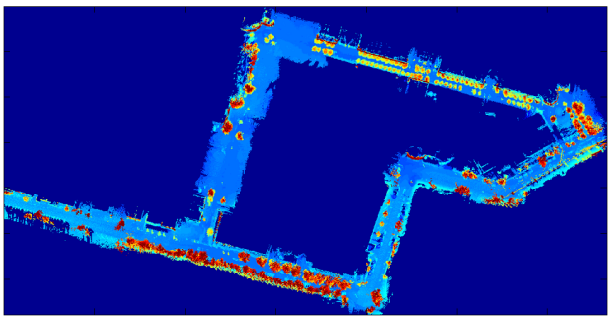


Figure 7: Elevation Map

such as 5 - 6 m, we obtain fewer points than with 2 - 3 m. After we extract the points in the space above 2 m and below 3 m, we can use a constant for representing the cell. The resulting map will be a occupancy map that has only two values, a constant or zero. Using a constant is a reasonable approach because considering the mean or the highest point might be noisy.

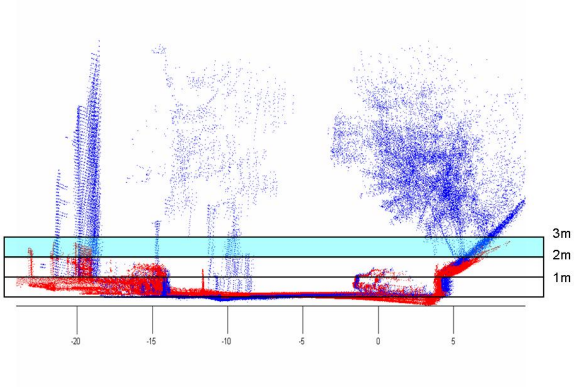


Figure 8: Extracting the space from 2 m to 3 m

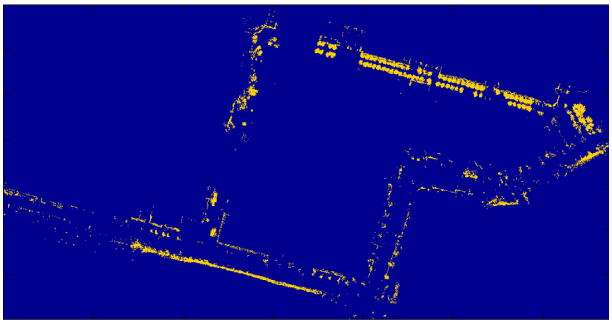


Figure 9: Feature Map

4.5 Landmarks

We can extract landmarks and remove overhang points by a simple method [1]. Usually landmarks could be tree trunks, telephone poles, and walls of buildings. These landmarks usually stand straight. We have the sweeping laser sensor that collects vertical data. This vertical data is suitable for extracting these landmarks. We slightly change the method in [1] to be apply it to our data. We use a constant for H_{max}. A 3D point $p_{i,j}$ is an overhang point if there is at least one point $p_{i,k}$ in the same vertical scan. $p_{i,k}$ is below $p_{i,j}$ and is at a larger distance to the sensor.

$$p_{ik,r} > p_{ij,r} + R_t \tag{1}$$
$$0 \leq k < j \tag{2}$$

where points $p_{i,j}$ and $p_{i,k}$ are in the i th scan, and R_t is the minimum overhang distance.

A 3D point $p_{i,j}$ is a landmark point if there is at least one point $p_{i,k}$ in the same vertical scan. $p_{i,k}$ is below $p_{i,j}$ and $p_{i,j}$ is not an overhang point.

$$H_t < p_{ij,z} - p_{ik,z} < H_{max}, \tag{3}$$
$$\left| \frac{p_{ij,r} - p_{ik,r}}{p_{ij,z} - p_{ik,z}} \right| < tan(\alpha_t) \tag{4}$$
$$0 \leq k < j \tag{5}$$
$$\tag{6}$$

where H_t is the minimum height of a landmark, H_{max} is the maximum difference between $p_{i,j}$ and $p_{i,k}$, and α_t is the maximum angle misalignment,

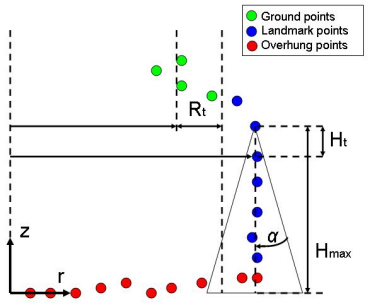


Figure 10: Definition of landmark and overhang points

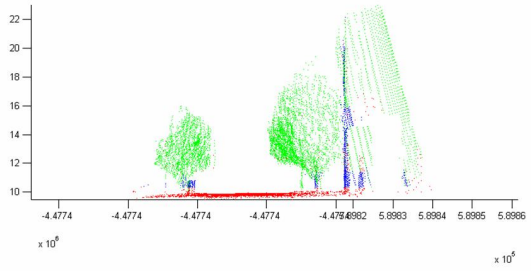


Figure 11: Tree trunks and building walls were extracted and overhang points were removed. Green points are removed overhang points, blue points are landmark points, and red points are other points.

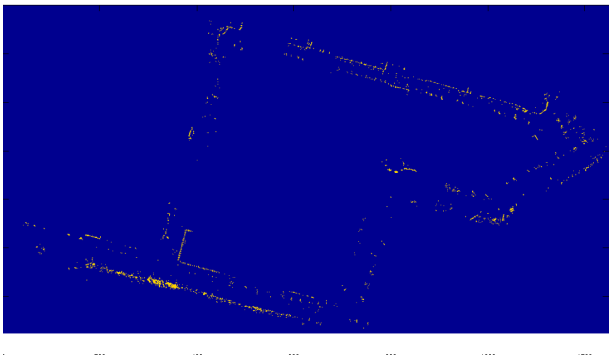


Figure 12: Landmark Map

5 Localization

5.1 Particle filters

There are three types of localization problems; tracking (local localization), global localization, and kidnapping. In the tracking problem, we estimate relative poses. Kalman filter is the most widely used localization algorithm. It approximates beliefs using mean and covariance. Extended Kalman filter solves non-linearities. An advantage of Kalman is its efficiency. Due to the assumption of unimodal posteriors, Kalman filters can only solve the tracking problem. If our estimate is completely wrong due to large noise of sensors or dead reckoning, it is difficult for the algorithm to track it again. In the global localization problem, we estimate a pose in the global coordinate. During an operation, if the robot is kidnapped and the pose is replaced, it is callded as the kidnapping problem. Kidnapping problem occurs when perception systems suddenly does not work and robot moves for certain time and then the sensors start to work again. In our case, we do not exactly know the initial pose of the vehicle. We want to localize the vehicle in the global coordinate. By comparing the known map with the local map created during the operation, we can obtain the global position in the known map. Particle filter is the best choice for localization in our case. Particle filters are Bayes filters that use random samples for posterior estimation. The filter can perform global localization even when we lose the vehicle pose again. The drawback of particle filters is the computational cost involved. By reducing the dimensionality of the pose and the number of particles, we can perform real time localization. In facts, this is a tradeoff between accuracy and computational cost. If we use more particles, it is highly possible that we can obtain a more accurate result. The advantage of the use of particle filters is the representation of arbitrary probability densities; thus, they can solve the global localization problem. The efficiency of particle filterer depends on the number

of particles. To reduce computational cost, real-time particle filters have been studied [5]. In our case, we focus on a small area and lower dimensionality of state space; therefore, particle filters can be applied in real-time operations. The pseudo code of a particle filter is shown in Fig. 1. First, we need to create a local map using the data obtained from ladars and odometry. Then, we generate particles from previous particles using their weights. Next, we propagate particles based on the uncertainty model and odometry and compute new weights by comparing the local map with a sub map created at the location of each particle in the global map. To determine the weight of each particle, we compute correlations between these maps. We can also use imaginary rays to compute the weight.

5.1.1 Recovery from failures

When the vehicle seems to get lost, we sometimes need to recover from the failures. To determine whether the vehicle got lost, we consider the likelihood. If the likelihood is really small, this means that even if particles converge, the vehicle might get lost because the known and the local maps are very different. It is possible that the estimate is correct even if the likelihood is very small. Failure occurs when there is no particle around the real vehicle pose. Therefore, we need to expand or replace the particles. We reset some particles among N particles. The reason why we do not reset all particle is to consider the case where the estimate is correct but the likelihood is low. We decide how big area we need to expand based on the standard deviation of particle clouds.

5.1.2 Uncertainty

The encoder and the gyro have noises as shown in Fig. 13. We need an uncertainty model for propagating the particles. A good uncertainty model improves the performance of our particle filters. We collected data for constructing an uncertainty model using the vehicle with GPS and odometry. Then we computed errors between GPS data. We computed errors at different speeds. Then, we computed the mean of absolute error and the mean of error of different speeds. Mean of error is a bias of this model. Small noises for propagating particles pose a problem that the particles cannot catch up when odometry gets large noises. Therefore, we select the largest noise among noises at different speeds for uncertainty model.

$$Errors_{\lceil \frac{ds^O}{interval_s} \rceil} = (ds^G - ds^O) \tag{7}$$

$$\sigma_v = \frac{std(Errors_v)}{v} \tag{8}$$

$$\sigma(ds) = max(\sigma_v) \tag{9}$$

$$Errors_{\lceil \frac{d\theta^O}{interval_\theta} \rceil} = (d\theta^G - d\theta^O) \tag{10}$$

$$\sigma_\omega = \frac{std(Errors_\omega)}{\omega} \tag{11}$$

$$\sigma(d\theta) = max(\sigma_\omega) \tag{12}$$

Table 1: Localization algorithm

$Algorithm Particle filter$
Inputs
M^L: Local map
M^G: Global map
y_t: Current observation
x_t: Current state
u_t: Control measurement
N: Number of particles
X_t: particles

$[m^L, M^L] := GetLocalMap(M^L, x_t, y_t, u_t)$ //Obtain a local map
for $n = 1, \cdots, N$ **do**
  $x_t^{G[n]} := MotionModel(u_t, x_t^{G[n]})$ //Propagate a particle
  $[m^G, M^G] := GetLocalMap(M^G, x_t^{G[n]})$ //Obtain a local map
  $w_t^{[n]} := CompareMaps(m^G, m^L)$ //Compute weight
end do

$X_t :=< empty >$
$x_t^* = 0;$
for $n = 1, \cdots, N$ **do**
  $w_t^{[n]} = \frac{w_t^{[n]}}{\sum w_t^{[n]}}$
  $X_t = X_t \cup \left\{ x_t^{G[n]}, w_t^{[n]} \right\}$ // Insert a particle into particle set
  $x_t^* = x_t^* + w_t^{[n]} x_t^{G[n]}$ //Compute weighted mean of particles
end do

if $max(w_t^{[n]} \sum w_t^{[n]}) > \kappa$ //If the maximum likelihood is greater than threshold
  $X_t := GenerateParticles(X_t)$ //Sample N particles based on weights
end

if $LostLocation() == 1$ //This function checks whether a robot lost its pose.
  $X_t := ResetParticles(X_t)$ //Reset Particles
end

Return X_t, x_t^* //Return new particles and estimate pose

,where N is the number of data, ds^G is the distance of travel in a second measured with GPS, ds^O is the distance of travel in a second measured with odometry, $d\theta^G$ is the relative orientation in a second measured with GPS, $d\theta^O$ is the relative in a second orientation measured with odometry, function $std()$ computes standard deviation, v is the velocity of the vehicle, and ω is the angular velocity. In this case, we compute one-second data, and the distance of travel and the relative orientation correspond to the velocity and the angular velocity respectively.

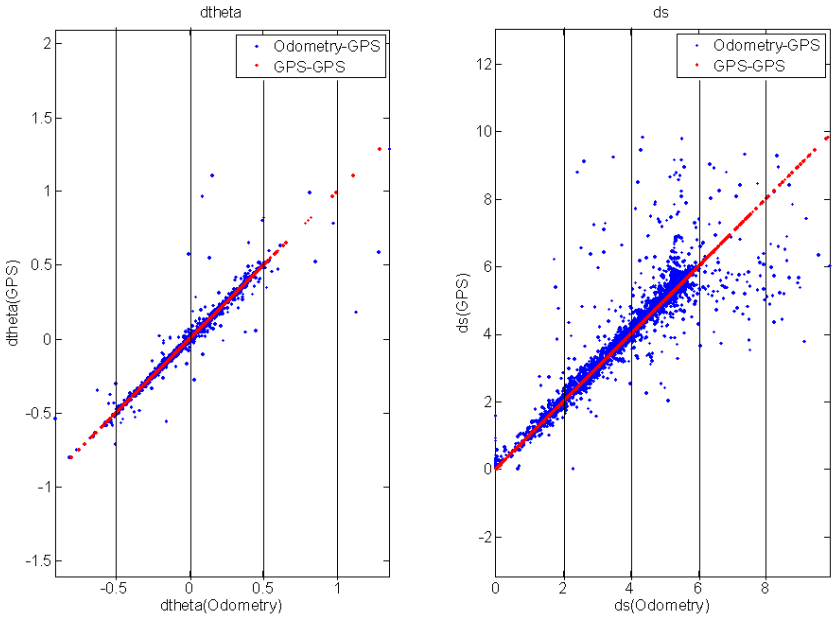


Figure 13: Difference between GPS data and Odometry data. We compute a model using the intervals 0.5 radian and 2 m.

5.2 Perception and comparing maps

When we compute the likelihood mentioned above, we need to select an appropriate perception system. There are some perception systems such as map matching, likelihood field, feature based-method and scan matching etc. We use map matching and scan matching to compute weights.

5.2.1 Map matching

We can compute correlations of these two maps [13]. The sensor measurement model compares the local map m_{local} with the global map m, such that the more similar m

and m_{local}, the larger $p(m_{local}|x_t, m)$. Since the local map is represented based on the robot pose x_t, this comparison requires that the sub map extracted from the global map is transformed into the estimated robot coordinate. The transformed sub map at the estimated pose $\hat{x}$ in the global map is compared with the local map using the map correlation function as follows:

$$\rho_{m,m_{local},x_t}(\hat{x}) = \frac{\sum_{x,y}(m_{x,y}(\hat{x}) - \bar{m}) \cdot (m_{x,y,local}(x_t) - \bar{m})}{\sqrt{\sum_{x,y}(m_{x,y}(\hat{x}) - \bar{m})^2 \sum_{x,y}(m_{x,y,local}(x_t) - \bar{m})^2}} \tag{13}$$

where $m_{x,y,local}(x_t)$ is the cell (x, y) in the local map at x_t, and $m_{x,y}$ is the cell (x, y) in the sub maps at $\hat{x}$. $\bar{m}$ is the average map value, which is expressed as

$$\bar{m} = \frac{1}{2N} \sum_{x,y} (m_{x,y} + m_{x,y,local}) \tag{14}$$

,where N denotes the number of overlapped cells between the local and sub map in the global map. The correlation ρ_{m,m_{local},x_t} takes the value from -1 to +1. This value is interpreted as follows:

$$p(m_{local}|x_t, m) = max(\rho_{m,m_{local},x_t}, 0) \tag{15}$$

5.2.2 Scan matching

After we extract the features, we can compute imaginary 2D rays as shown in Fig.**??**. These imaginary rays facilitate the use of indoor-like localization method since standard indoor robots have 2D horizontally fixed ladars. Using imaginary 2D rays, we obtain a distance and a orientation to each feature. We compute a likelihood by comparing the distances and the orientations. The difference is that this localization method can utilize 3D data but uses fewer data.

$$p(z_t^k|x_t, m) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(z_i^k - \bar{z}_i^k)^2}{\sigma^2}} \tag{16}$$

$$p(z_t|x_t, m) = \prod_{k=1}^{K} p(z_t^k|x_t, m) \tag{17}$$

where k is the k th measurement in a scan.

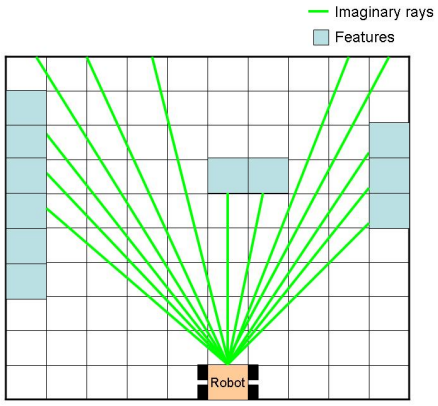


Figure 14: Imaginary 2D rays

6 Experimental results

6.1 Experiment

Our vehicle has two SICK laser sensors. Fixed laser sensor has 180° field of view with 1° resolution and collects data at 75 Hz. Sweeping laser sensor has 90° field of view with 0.5° resolution and collects data at 75 Hz. The vehicle has a encoder and a gyro. They provide information about relative yaw angle and travel of distance at 100 Hz. We collected data on Carnegie Mellon University campus on a weekend morning. There were few people and cars. We used GPS and laser data for creating a map, and then used odometry data and laser data for localization. We drove four laps around the campus at a speed of about 5 m/s in the counter clockwise direction and another four laps in the clockwise direction. First lap in both directions were used for creating a known map. Other laps were for testing our localization algorithms. We performed the localization off-line using the collected data. We didn't use GPS data for localization. We have done five combinations of algorithms as follows.
(1) Elevation map - map matching
(2) Feature map - map matching
(3) Feature map - scan matching
(4) Landmarks - map matching
(5) Landmarks - scan matching
The results depend on parameters of mapping and localizatoin, and therefore we used well-tuned parameters selected empirically. We supposed that the initial position is unknown but know that vehicle is in a certain small area such as about 50 m by 50 m. The distance of travel is about 4 km. We use 300 particles for each algorithm.

6.2 Results

Each algorithms showed that it can localize the vehicle even though the initial location is unknown. If we use map matching, it takes more time than with scan matching. In particulary, the computational cost of map matching depends on the resolution of the map. Using Landmarks also needs more time than using elevation or feature map. As shown in Fig. 21, 22, 23,24, and 25, weighted mean of particles shows the best result in terms of position but not orientation. Trajectory generated from particle with the largest weight or particle with the best history is very noisy compared to that generated from weighted mean. Therefore, we use weighted mean to estimate the pose of the vehicle.
Table 2 show the mean and the standard deviation of the estimated poses. Table 3 show the mean and the standard deviation of the estimated poses not using bad matching data. Using feature map and scan matching resulted in the best performance although all algorithms gave similar errors. These results also show that elevation map should be used for map matching, and feature map and landmarks should be used for scan matching. Map matching performed better results in the orientation of the vechile while scan matching performed better results in the position of the vhicle. Fig. 16, 17, 18, 19 and 20 show trajectory of each result. Problems occures when the vehicle moves in open areas. This is because particles have orientation errors and if we propagate

without resampling, orientation errors affects the performance of localization. Thus, the estimated position of vehicle was off the path in the open spaces, however, we recovered from failures using resetting. Figures31, 32, 33, 34, and 35 show cross and long track error. The cross track error is the distance from the closest path. The long track error is the distance from the closest position along the path. The long track errors are larger than the cross track error, which means that the distance error originates mainly from the long track error. The vehicle can view only about 20 m in front of the vehicle. Thus, although the vehicle easily knows the distance from the sides of the roads, it has difficulty in knowing its position along the road if the environment has similar features.

Plane	Distance Error		Orientation Error	
	Mean (m)	Std (m)	Mean (degree)	Std (degree)
Odometry	7.298	4.3472	3.227	4.3873
Elevation map - map matching	2.4604	1.6736	3.4873	9.7366
Feature map - map matching	2.6571	1.4978	3.7361	10.5098
Feature map - scan matching	1.804	1.2191	2.7178	4.2572
Landmarks - map matching	2.8778	1.5104	3.0178	4.8493
Landmarks - scan matching	2.2457	1.8486	3.8109	13.3221

Table 2: Error of distance of travel and orientation

Plane	Distance Error		Orientation Error	
	Mean (m)	Std (m)	Mean (degree)	Std (degree)
Odometry	7.298	4.3472	3.227	4.3873
Elevation map - map matching	2.016	1.3153	2.12	7.4853
Feature map - map matching	2.2803	1.2614	2.3258	9.6596
Feature map - scan matching	1.6957	1.0977	2.7178	4.2572
Landmarks - map matching	2.5206	1.3424	1.749	3.1905
Landmarks - scan matching	2.0679	1.5865	4.419	15.23771

Table 3: Error of distance of travel and orientation without bad matchings

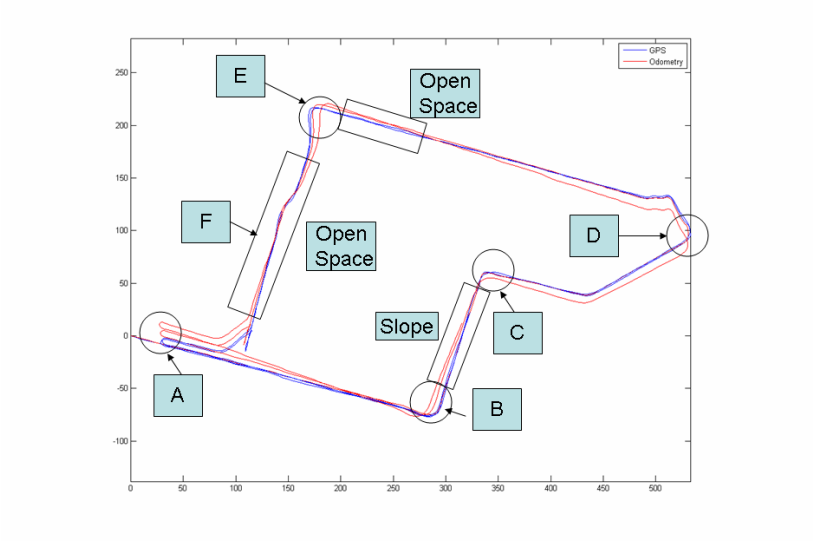


Figure 15: Odometry and GPS. A - F are area names.

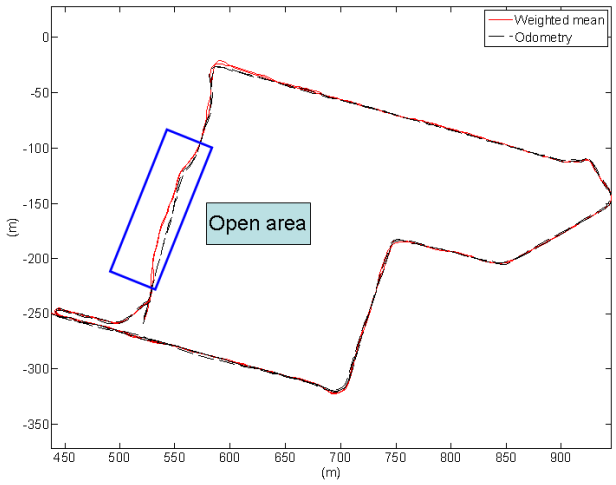


Figure 16: Trajectory obtained using Elevation map and map matching. We got large errors when the veihlce drives in the open space.

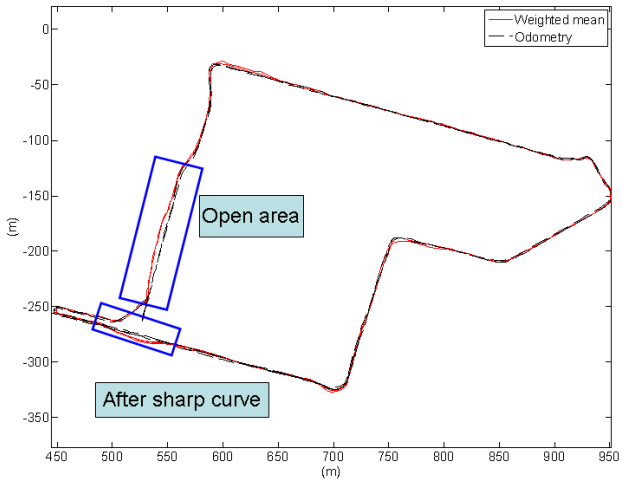


Figure 17: Traectory obtained using feature map and map matching. We got large errors in the open space. After the vehicle made a sharp turn, we also got large errors.

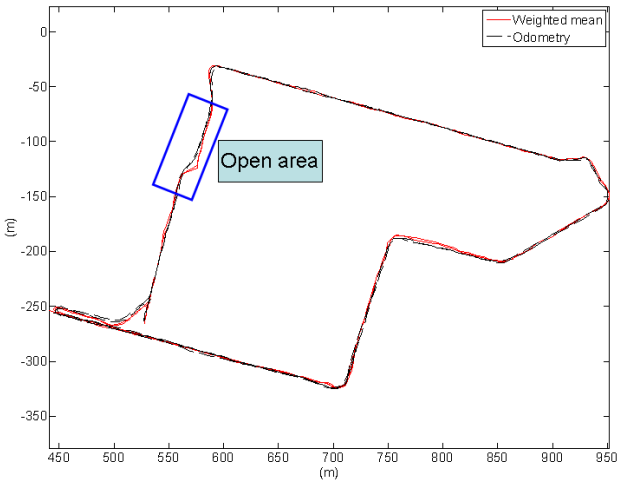


Figure 18: Trajectory obtained using feature map and scan matching. We got large errors in the open space.

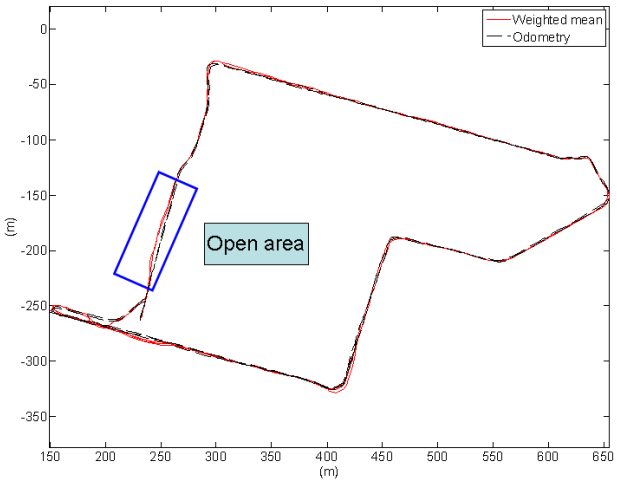


Figure 19: Trajectory obtained using landmarks and map matching. We got large errors in the open space.

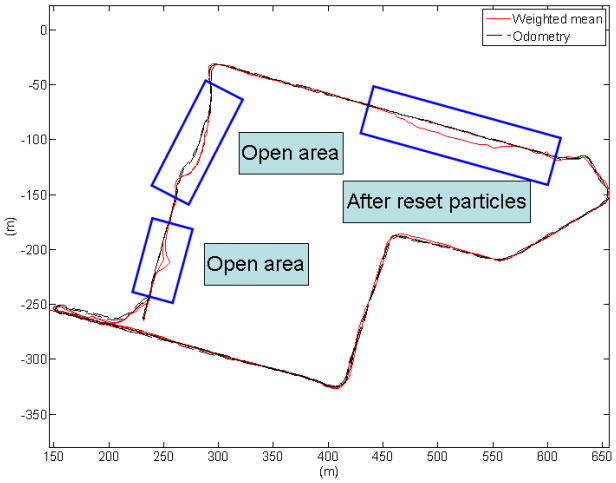


Figure 20: Trajectory obtained using landmarks and scan matching. We got large errors in the open space. Several seconds after we reset partilces, we got errors.

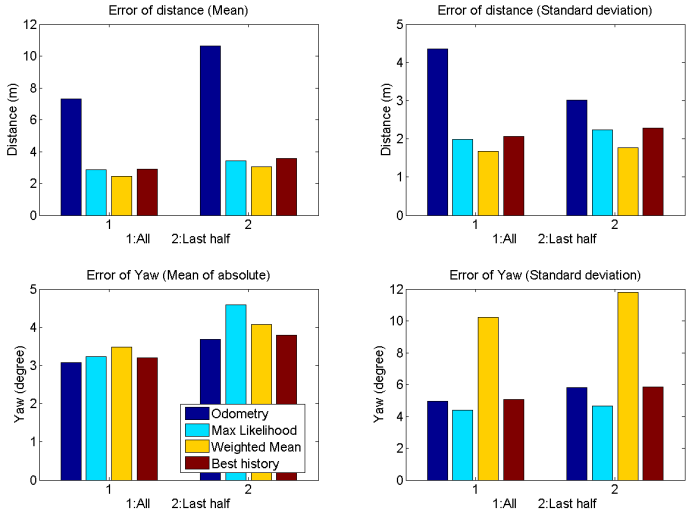


Figure 21: Statistics obtained using elevation map and map matching

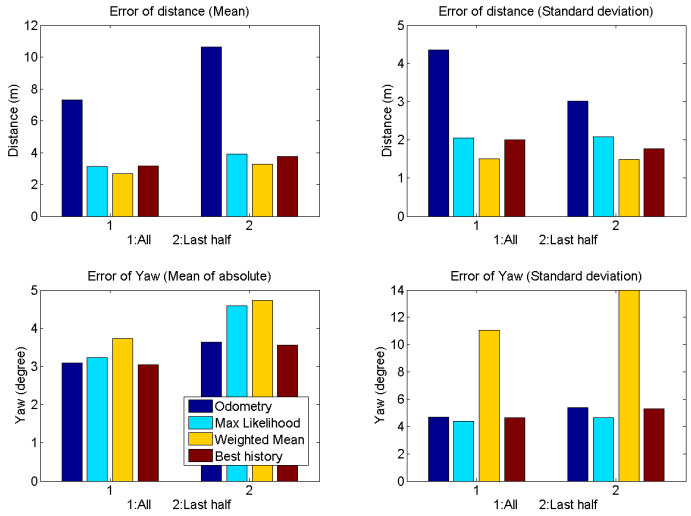


Figure 22: Statistics obtained using feature map and map matching

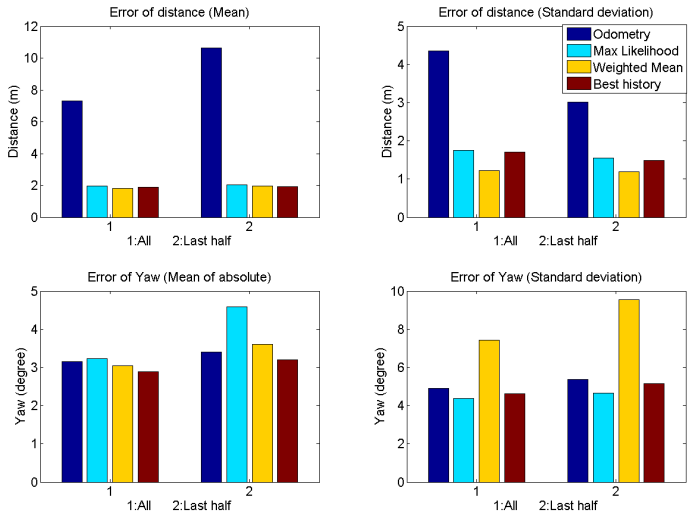


Figure 23: Statistics obtained using feature map and scan matching

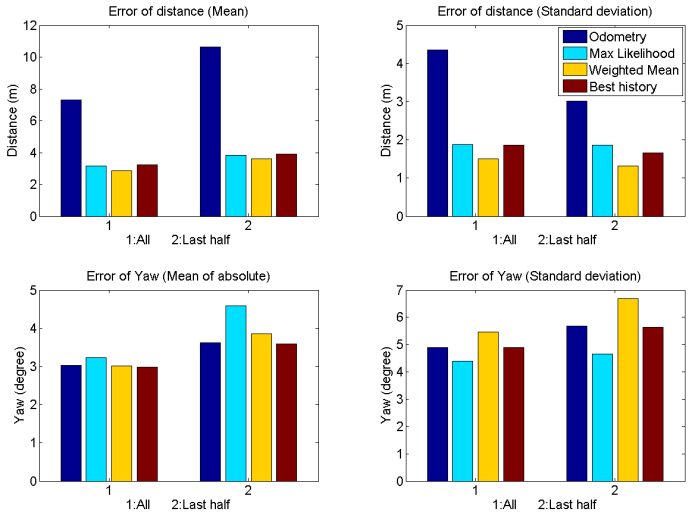


Figure 24: Statistics obtained using landmarks and map matching

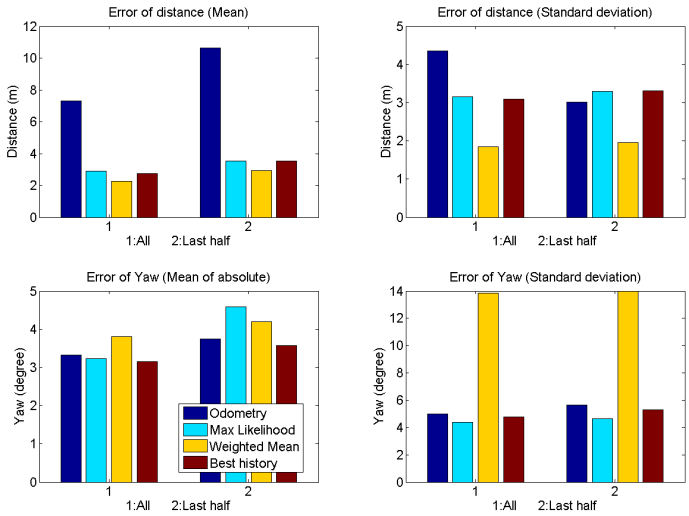


Figure 25: Statistics obtained using landmarks and scan matching

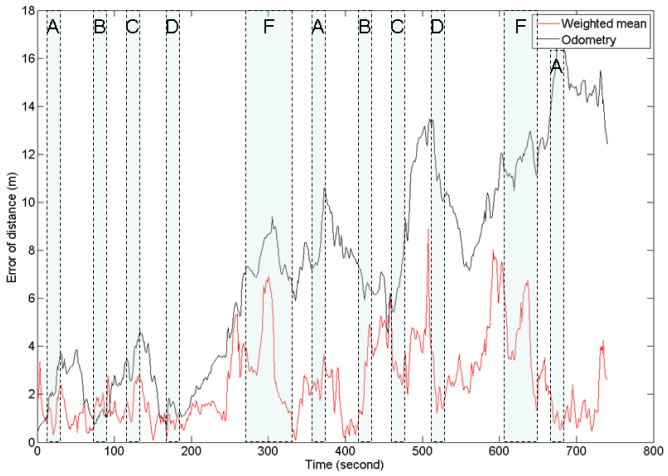


Figure 26: Error of Distance obtained using elevation map and map matching. A - F represent areas shown in Fig.15.

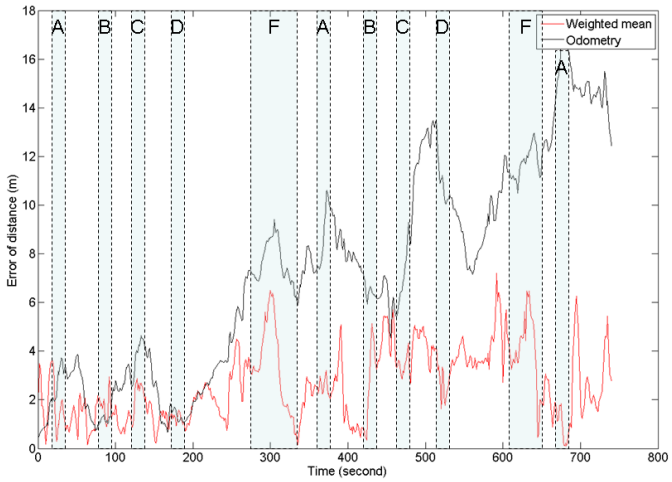


Figure 27: Error of Distance obtained using feature map and map matching. A - F represent areas shown in Fig.15.

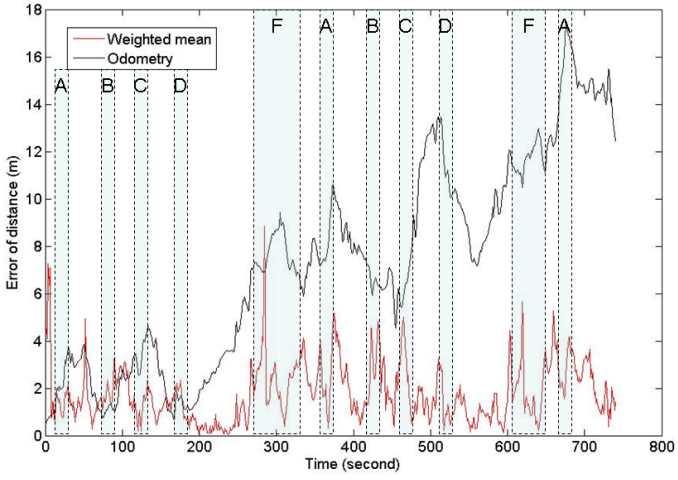


Figure 28: Error of Distance obtained using feature map and scan matching. A - F represent areas shown in Fig.15.

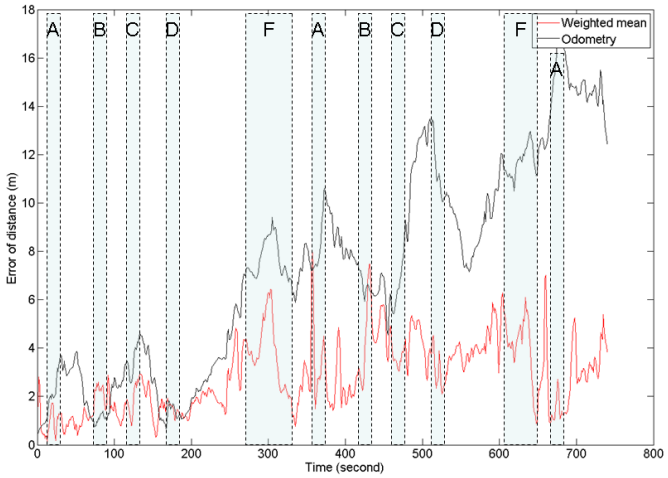


Figure 29: Error of Distance obtained using landmarks and map matching. A - F represent areas shown in Fig.15.

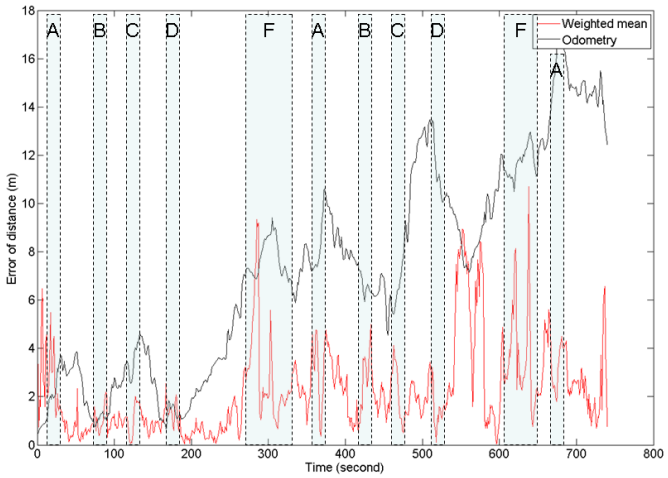


Figure 30: Error of Distance obtained using landmarks and scan matching. A - F represent areas shown in Fig.15.

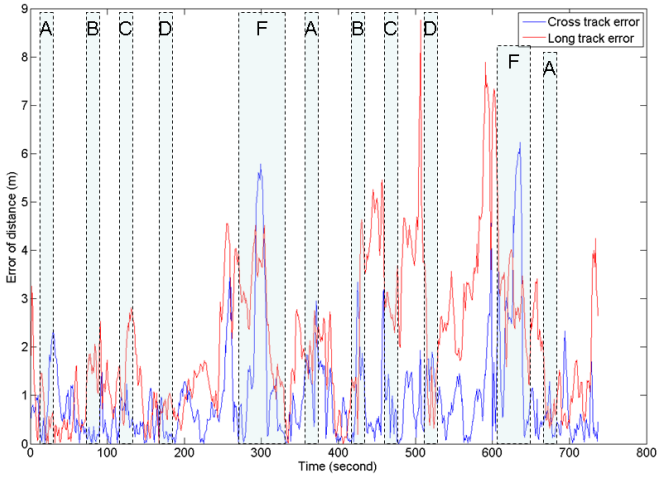


Figure 31: Cross and Long track errors obtained using elevation map and map matching. A - F represent areas shown in Fig.15.

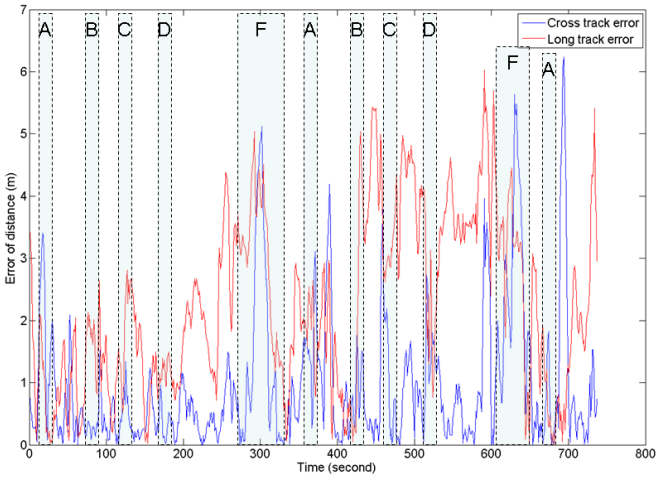


Figure 32: Cross and Long track errors obtained using feature map and map matching. A - F represent areas shown in Fig.15.

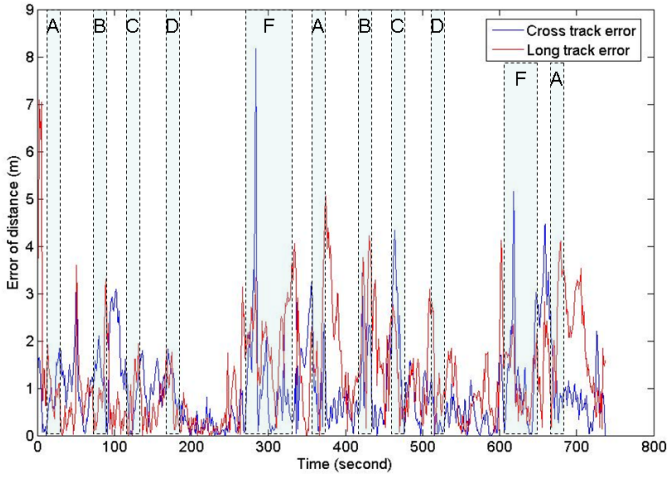


Figure 33: Cross and Long track errors obtained using feature map and scan matching. A - F represent areas shown in Fig.15.

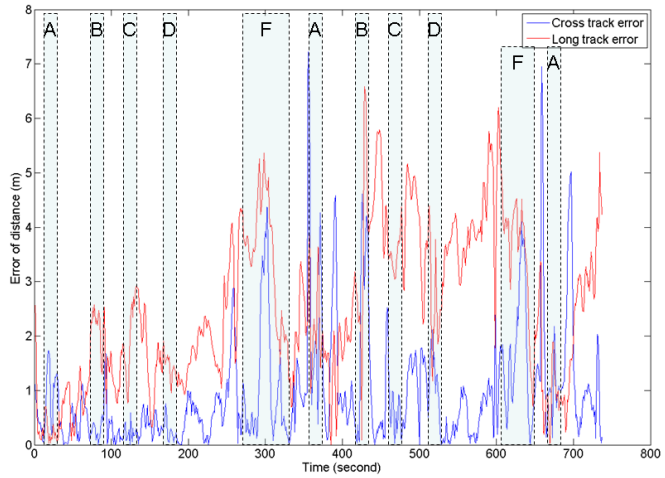


Figure 34: Cross and Long track errors obtained using landmarks and map matching. A - F represent areas shown in Fig.15.

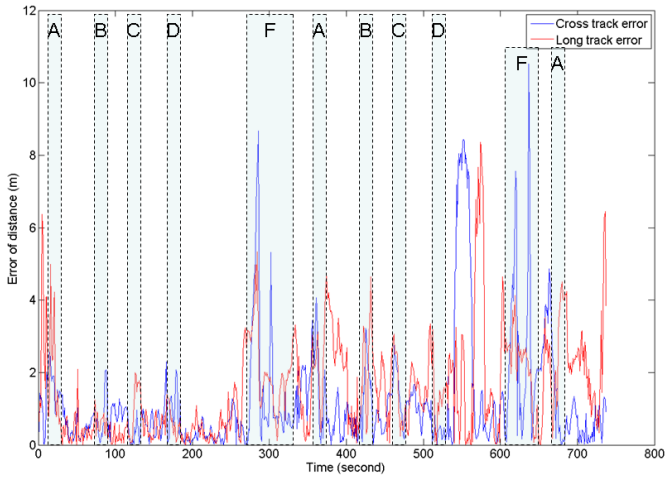


Figure 35: Cross and Long track errors obtained using landmarks and scan matching. A - F represent areas shown in Fig.15.

7 Conclusions and future works

7.1 Conclusions

We showed three map representations and two comaprison methods for a particle filter.Then,We compared five localization algorithms that use the benefits of 3D data in creating a 2D map. We reduced a large amount of 3D data obtained from ladars by converting them into 2D and preserved their features. We do not need to have thousands of particles, but just hundreds for localization and these facts make the algorithm compute in real time. Even though there are hills and open space in the campus and odometry goes off as the vehicle moves, our localization algorithm can still localize the vehicle. Even if we reduce the information by removing dynamic objects and low height objects, it can still localize the vehicle and extracting landmarks is also applicable in this situation. Particulary, using feature map and scan matching results in the best performance in this situation. Resetting particles and ignoring low quality matching result made the algorithm robust.

7.2 Future works

We collected data using the vehicle; however we localized the vehicle off-line. We are going to implement this algorithm on the vehicle and then use obstacle detection and avoidance, path planning and tracking, and localization simultaneously. We tested our algorithms just in the campus. It is important to test these algorithms in different environments. To improve the performance of localization, we need to consider a combination of localization algorithms such as using particle filters for detecting the initial pose and using kalman filter for tracking.

References

[1] C. Brenneke, O. Wulf, B. Wagner, "Using 3D Laser Range Data for SLAM in Outdoor Environments", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.188-193, Las Vegas, USA, 2003*

[2] O. Wulf, C. Brenneke, B. Wagner, "Colored 2D Maps for Robot Navigation with 3D Sensor Data", *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, September, 2004*

[3] S. Thrun, D. Fox, W. Burgard, F. Dellaert, "Robust Monte Carlo Localization for Mobile Robots",*Artificial Intelligence Journal, 2001*

[4] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo Localization for Mobile Robots", *IEEE International Conference on Robotics and Automation (ICRA99), May, 1999*

[5] C. Kwok, D. Fox, M. Meila, "Adaptive real-time particle filters for robot localization", *Proceedings of the IEEE international Conference on Robotics and Robotics and Automation,(ICRA) 2003*

[6] C. Kwok, D. Fox, "Real-time particle filters", *Proceedings of the IEEE, Vol.92, No.3, 2004*

[7] S. Lenser, M. Veloso, "Sensor Resetting Localization for Poorly Modeled Mobile Robots", *Proceedings of the IEEE International Conference on Robotics and Automation, 2000*

[8] R. Ueda, T. Arai, K. Asanuma, K. Umeda, H. Osumi, "Recovery Methods for Fatal Estimation Errors on Monte Carlo Localization", *Journal of Robotics Society of Japan Vol.23 No.4, 2005*

[9] M. Bosse, J. Roberts, "Histogram Matching and Global Initialization for Laser-only SLAM in Large Unstructured Environments", *IEEE International Conference on Robotics and Automation, April, 2007*

[10] D. Huber, M. Hebert, "A New Approach to 3-D Terrain Mapping", *Proceedings of the 1999 IEEE/RSJ international Conference on Intelligent Robots and Systems, 1999*

[11] C. Olson, L. Matthies, "Maximum Likelihood Rover Localization by Matching Range Maps", *Proceedings of the IEEE International Conference on Robotics and Automation, 1998*

[12] S.A. Roth, B. Hamner, S. Singh, M. Hwangbo, "Results in Combined Route Traversal and Collision Avoidance", *International Conference on Field and Service Robotics (FSR '05), July, 2005*

[13] S.Thrun, W.Burgard, D.Fox, "Probabilistic Robotics", *The MIT Press*

[14] K. Lingemann, H. Surmann, A. Nuchter, J. Hertzberg, “Indoor and outdoor localization for fast mobile robots”, *Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’04), 2004*