# Robot Search and Rescue
# A Comparison of 3D Mapping Techniques

by
Maria Guirguis
S.B., E.E.C.S., M.I.T., 2009

Submitted to the Department of Electrical Engineering and Computer Science
In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
Massachusetts Institute of Technology
May 2010

Copyright 2010 Maria Guirguis. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole and in part in any medium now known or hereafter created.

Author ____________________

Department of Electrical Engineering and Computer Science
May 21, 2010

Certified by ____________________

Cynthia Breazeal, Associate Professor of Media Arts and Sciences
Thesis Supervisor

Certified by ____________________

Philipp Robbel, PhD candidate
Thesis Co-Supervisor

Accepted by ____________________

Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

# Robot Search and Rescue
# A Comparison of 3D Mapping Techniques

by
Maria Guirguis
egyptoz@mit.edu

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 2010

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

**Abstract**

Modern robots are involved in sophisticated manipulations of their environment, and for that they need extensive knowledge of their surroundings. 3D mapping allows for the creation of such complex maps, and here we explore some of the options available for the creation of 3D maps. We consider using 2D and 3D sensors to see how helpful the extra information is.

## Supervisors:

Cynthia Breazeal, Associate Professor of Media Arts and Sciences
Philipp Robbel, PhD candidate
MIT Media Lab
Personal Robots Group

# Contents

# List of Figures

## 1 Introduction

Urban Search and Rescue has been a topic of much interest in recent years, and there is a lot of active research in applying robotics in that area. Robots are supplied with various types of sensors to aid the exploration of their surrounding, and are utilized for data collection and map creation in areas that are too dangerous or otherwise inaccessible to humans. A decision is made as to which sensors are useful and how they can be combined to maximize the information gathered, and the robot can be set free to roam. Because real sensors generally have errors, it is usually good to combine inputs from multiple ones so the error can be checked and minimized.

This task is becoming easier as the technology used in robot sensors advances, making it easier for robots to learn about and interact with their environment robustly. In order to make decisions in realtime, the processing of the collected data needs to happen quickly and efficiently to make the most use of that information. By using combinations of faster and more efficient sensors, we can make robots a viable choice as helpers in search and rescue operations. The task is now to make the robots as autonomous and self-reliant as possible in carrying out their operations.

## 2 Project Background

This project is part of a larger project dealing with team-based robotic search and rescue. The goal is to be able to send in a team of ground and aerial robots that coordinate amongst them the tasks of searching for people and leading them to safety. The ground robot used is the Mobile, Dextrous, Social (MDS) Robot, which is equipped with a Hokuyo laser scanner, a SwissRanger ToF camera, and stereo cameras. The goal of my exploration is to compare two different approaches to 3D map creation- one which creates a map using

2D-based localization (x,y, heading, along with readings from the laser scanner), and one which utilizes the full 6 dimensions (x, y, z, yaw, pitch, roll, together with 3D range readings) for localization and mapping. Both approaches plot the same set of 3D data but each has a different way of localizing the robot to fit that data into a map. The exploration is carried out using the Urban Search and Rescue Simulator (USARSim), where a model of the robot and all its sensors has been deployed for data collection and testing.

## 3 Previous Work

The idea of mapping is as old as humanity. Maps help humans navigate the world and find things easily. Creating maps, however, is a very difficult task if done manually, and it needs exact measurements. Once robots could move around, it was the logical next step to supply them with distance sensors and employ the data thus collected for map creation. Accordingly, mapping has been a very active and central area in robotics research for years. 2D mapping has been extensively studied, but 3D mapping has not been pursued as assiduously. This is due to its being a much more complicated problem, as the increase in dimension introduces new degrees of freedom in the movement of the robot, which makes it more difficult to maintain an accurate location estimate.

Simultaneous Localization and Mapping, is a class of algorithms used to automatically create maps using mobile robots. It involves alternating the tasks of localization and mapping, calculating location on a map based on an internal belief of location and inputs form a displacement sensor, then re-evaluating the location based on the measured map and sensor inputs, until a map of the desired area is created. Usually SLAM is done with 2-dimensional distance sensors that provide readings referring to locations of nearest obstacles along a slice of the world. 2D sensors have been studied profusely in the context of SLAM, and many excellent techniques exist for coming up with very good planar maps. This is due to the ability to acquire very high quality information from relatively inexpensive sensors.

Those algorithms, however, assume that the robot is traveling on flat ground and that all the readings are coplanar, that the map being created is a horizontal slice through the world. This assumption doesn't hold for search and rescue situations in disaster areas, as there is debris and rubble that could cause the robot's movements to not be flat, affecting the sensor's data. Also, the presence of this debris makes it more difficult to accurately estimate the robot location because it can cause incorrect estimates of the movement due to the more probable slippage of the wheels. Additionally, it is useful to have information about the wall for the entire height of the room- where a door ends, for example, or whether there are windows- that cannot be gathered from a horizontal slice of the room. For that, it is more helpful to create and use a 3D map. The goal is to create reliable global maps quickly (online, without stopping the robot). This provides for more details that can be exploited by people using the maps. This has become more feasible in recent years due to the availability of high-quality 3-Dimensional distance sensors. Unfortunately, the high quality comes at a price:

- Those sensors are generally much more expensive than their counterparts used in planar mapping.
- Due to the high volume of data collected, the frequency of scans is much lower than for 2D maps, and the robot has to run slower, even stopping when a scan is being done.
- They tend to produce a larger amount of data due to their nature, which makes it more difficult to process.

We will start out by discussing some 2D mapping algorithms and how we extend them to create 3D maps, examples of 3D sensors and a discussion of how they work, as well as full 6D SLAM algorithms.

## 3.1 2D SLAM

2-Dimensional maps are what we normally think of as maps- planar line drawings describing where important geometrical objects or features are in the world. Available sensors produce point data as a result of using beams to scan the environment, so any mapping technique will have to start with such data. Some examples of 1 or 2D scanners include sonar sensors, laser scanners, and IR sensors. As the robot moves, they collect data along a given slice of the world, at the height where the sensor is affixed to the robot. One of many algorithms can be applied to this collected data- online or offline- to create a map. Some approaches to 2D SLAM are scan matching and curve fitting, particle filters, and occupancy grids [7].

Scan matching finds corresponding parts of consecutive scans then attempts to minimize the cumulative or average distance between them. Once two scans are matched, the transformation from one to the other is calculated and used to improve the localization estimate. It is important to have a reasonable initial guess for the relative transformation, as this makes it more likely that the scan matching would give the best match, and that the optimization algorithm can calculate the best transformation.

Also, probabilistic methods can be applied to the problem of 2D SLAM, as it allows for more flexibility in localizing the robot. The stochastic model employed depends on the uncertainty in the robot's movement and sensor measurements, anticipating the errors and trying to compensate for them in creating good maps. Some probabilistic methods used include Maximum Likelihood Estimation and Kalman Filters.

Occupancy grid mapping is based on a probabilistic approach. It creates a grid for the world, and fills each square with the likelihood of that square being empty of obstacles. That likelihood is continuously re-evaluated based on sensor input from odometry and distance scanners.

The last SLAM method we discuss is particle filters, which is employed in the 2D localization algorithm, GMapping [4]. This algorithm maintains multiple "particles" that are associated with different possible states of the world (maps and robot location). At each step, we sample from a distribution of possible next locations given the measured odometry displacement, then narrow down the sampled possibilities using the input from the scanner and its associated distribution. This algorithm is good, but could easily lose the "correct" particle during the resampling step and not be able to find its way back to the correct map.

Grisetti, Stachniss and Burgard use Rao-Blackwellized particle filters to create a robust sampling method featuring a highly accurate distribution for generating potential particles (proposal distribution) and adaptive resampling of the particles to reduce the risk of particle depletion. The initial proposal distribution at a given time step is based on the previous particle distribution, with sampling based on the probabilistic model of the odometry estimate, and scan matching to give more weight to the more likely particles based on the range scanner input.

Rao- Blackwellized particle filter for SLAM is used to estimate the joint posterior $p(x_{1:t}, m|z_{1:t}, u_{1:t-1})$ about the map $m$ and the entire trajectory $x_{1:t} = x_1, \ldots, x_t$ of the robot. This estimation is performed given the observations $z_{1:t} = z_1, \ldots, z_t$ and the odometry measurements $u_{1:t-1} = u_1, \ldots, u_{t-1}$ obtained by the mobile robot. The Rao-Blackwellized particle filter for SLAM makes use of the following factorization

$$p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = p(m|x_{1:t}, z_{1:t}) \cdot p(x_{1:t}|z_{1:t}, u_{1:t-1}) \tag{1}$$

which assumes the independence of the two factors. This factorization allows us to first estimate only the trajectory of the robot and then to compute the map given that trajectory. This technique offers an efficient computation and is often referred to as Rao-Blackwellization. It especially points to the reliance of map creation on the pose estimate.

Typically, Eq. 1 can be calculated efficiently since the posterior $p(m|x_{1:t}, z_{1:t})$ over maps can easily be computed analytically using $x_{1:t}$ and $z_{1:t}$, which are known.

To estimate the posterior $p(x_{1:t}|z_{1:t}, u_{1:t-1})$ over the potential trajectories, a particle filter can be used. Each particle represents a potential trajectory of the robot and a map that is associated with that trajectory. The maps are built from the observations and the trajectory represented by the corresponding particle. This guarantees that the generated map has very high probability, since all of the poses along the entire robots trajectory are considered in its creation.

This approach takes into account the accuracy of the laser range finder used and favors locations that are more likely based on the previous observation when creating the proposal distribution after taking a step, to provide for more accurate and methodical sampling. If a particular sensor on the robot has very high accuracy, then that sensor's measurements are exploited to filter the samples and choose the best particles.

To create a 2D map, the MDS robot uses the Hokuyo laster scanner (Figure 1) to collect 2D data for localization using Rao-Blackwellized Particels. We later plot 3D point data collected using the SwissRanger together with the calculated coordinate frame to create a full 3D map using our 2D range scans.

## 3.2 3D Sensors

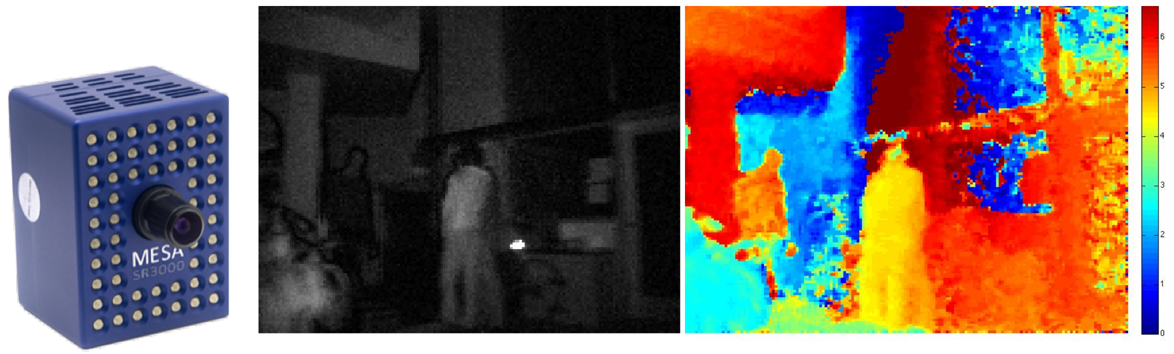
Many sensors can be used for collecting 3D range data. The sensors traditionally used for collecting 3-Dimensional distance measurements are LADAR scanners, which place a 2D

Figure 1: Hokuyo laser scanner.

Laser sensor on a rotating axis to expand the distance measurements into 3D. The robot needs to be stationary for the entire time of the scan, making it a very slow process, and the configuration of the scanner gravely affects the quality of the data.

Other means of collecting 3D range data include CCD array cameras, which can calculate distances using stereo vision or patterned flashes that assign each point a unique grey code for ID that can easily be read, and triangulation-based camera-beam systems that calculate the distance to the object based on a geometric formula using the relative location of the camera, beam origin, and beam endpoint in the environment.

The simplest 3D range scanners, however, are Time of Flight cameras, which send out several beams and collect them, calculating the distance based on time delay or phase shift. An example of ToF, phase shift camera is the SwissRanger 3000 which we use for this exploration. The SwissRanger is a cheap, high frequency, low resolution 3D sensor. This very small camera, shown in Figure 2 below, generates measurements with a resolution of $176 \times 144$ pixels at 20 fps, making it very fast but very low resolution compared with other sensors used in machine vision (which can give up to $1000 \times 1000$ points per scan). The Field of View of this camera is $47.5^o \times 39.6^o$, which is much narrower than laser scanners



(a) SwissRanger (b) Intensity Reading from SwissRanger (c) Distance Reading from SwissRanger

Figure 2: The SwissRanger 3000 Camera along with an intensity image and a range image. Note the grainy quality of these snapshots.

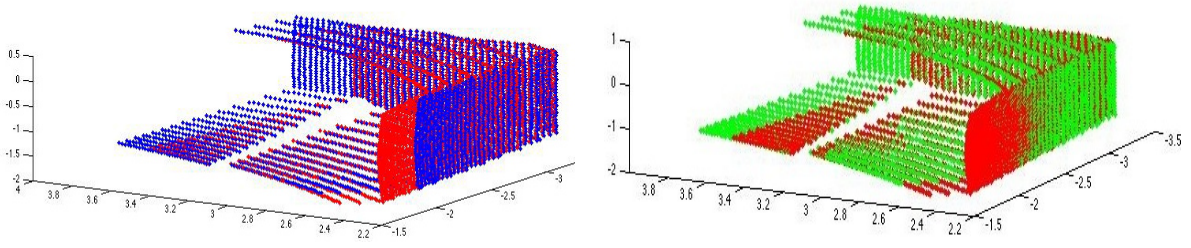
(which scan 180$^o$). Each frame is constructed by sending out a beam of infrared light and waiting for it to bounce back. Once the beam is recollected, the distance is calculated using the time and phase shift from the original signal, and the intensity of the returned signal is calculated from the ratio of returned to initial beams. These two readings, an example of which is shown in Figure 2, are returned for each measurement. As you can see, the distance values are in the range [0,7] meters. [2] has a great discussion of various sensors, including the SwissRanger, along with things to note when using those sensors.

## 3.3 6D SLAM

Due to the nature of 3D maps which would be attempt to fill up all of space, grid-based maps are very computationally expensive to generate, and are generally not considered. 3D maps usually use point clouds or 3D mesh surfaces for calculation and display.

### 3.3.1 Scan Registration

Inherently imprecise sensors provide inexact data, creating the need for registration. Scan registration is an important step in the generation of 3D maps based on 6D localization, as it is the process of merging all the collected data into 1 unified coordinate frame. It can be done by complete scan matching or by feature selection and matching. Feature-based localization is good to use in support of another SLAM method, but would be lacking if used by itself. A common technique for scan registration is the Iterative Closest Point (ICP). It calculates-via an iterative method- a transformation in 3-Dimensional space which provides the best transformation which matches two data sets. Those data sets can be 3D world coordinates of walls or other features in the world, collected by any available means- such as distance sensors and cameras, or any two collections of features whose distance can be calculated. Because ICP is independent of the representation, we can use- in addition to 3D points- lines and surfaces, and even CAD meshes [1].



(a) Ground truth data

(b) ICP matching, $e_{\mathrm{rms}}$= 0.0237

Figure 3: ICP match is very close to the ground truth

The ICP Algorithm was invented by Besl and Mckay [1]. It assumes a model $M$ and a set of 3D data $D$, and seeks to find the transformation $(R, t)$ such that

$$E(R,t) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \|m_i - (Rd_j + t)\|^2 \tag{2}$$

where $w_{i,j}$ is an indicator, equal to 1 if $m_i = d_j$ and 0 otherwise. This equation shows we only consider points that are considered near matches to begin with, and all points that have no potential near matches do not contribute to the error calculation. Due to its iterative nature, where the error monotonically decreases with every step, ICP guarantees the minimum error correspondence between the pair of scans, given the maximum number of iterations and stop conditions.

There are many optimization techniques used for representing the transformation between two scans for the duration of the iterations. Some examples include:

- Separating the Rotation and translation: finding the distance between the two means, removing it, and then finding the best rotation.
- Using SVD to find the rotation matrix by factoring the matrix of the correlation between M and D.
- Using orthonormal matrices to find the rotation matrix.
- Linearizing the displacement by approximating it as helical motion
- Using unit quaternions, an extension into 3D of complex numbers, to calculate the rotation matrix.

$$R = \begin{pmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y - 2q_0q_z & 2q_xq_z + 2q_0q_y \\ 2q_xq_y + 2q_0q_z & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z - 2q_0q_x \\ 2q_xq_z - 2q_0q_y & 2q_yq_z + 2q_0q_x & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix} \tag{3}$$

where

$$\begin{pmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) a_x \\ \sin\left(\frac{\theta}{2}\right) a_y \\ \sin\left(\frac{\theta}{2}\right) a_z \end{pmatrix} \tag{4}$$

and $\theta$ is the rotation angle around the rotation axis $a = (a_x, a_y, a_z)$. Note that saving the data as a unit quaternion is much more efficient than the full rotation matrix, as it only necessitates storing 4 items as opposed to 9.

At the end of a registration step, ICP stops when it has plateaued off, and so it is useless to run it again looking for improvements. If the maximum number of iterations, however, is not enough for convergence (ie. the convergence is very slow), we might see incomplete registration. The transformation calculated from the registration of a new scan is applied to the odometry estimate to improve the localization estimate at that time step.

However efficient the registration method is, it assumes that it already has pairs of corresponding points matched up and it tries to minimize the distance between them. Actually finding those pairs is a much more extensive job.

### 3.3.2 Calculating Closest Points

Some methods for speeding up the search for closest points involve breaking up the space into equally-sized grid cells and starting the search in the cell where the point is, extending to nearby cells if nothing is found. This presumes a uniform distribution of the data, which is usually not the case[6].

The one we use here is K-D tree, a type of generalized binary tree. This method divides the space to buckets, each containing a certain number b of points, separated by axis-aligned hyperplane decision boundaries for each of K dimensions and D "split dimensions".

It is arranged so that the buckets all have approximately the same number of points, and the search algorithm takes $O(KN \log N)$ time at worst. Additionally, randomized methods, parallelizing, and caching can also be employed to speed up the algorithm, but the latter comes at the price of more memory usage or increased requirements for the machine running the algorithm.

Another way to streamline the nearest neighbor search is by decreasing the number of points we use for the scan matching. This is achieved by subsampling, which can be done randomly, but produces much better results if the geometry of the surrounding environment is taken into account and choosing more useful points to keep [3].

It is important to note that it is almost never the case that each single point in $D$ will find a match in $M$, so trying to force all points to be matched will produce unsatisfactory results. One might be tempted to use the $w_{i.j}$'s as weights as opposed to indicator variables, but that has been shown to not produce significantly different results. In SLAM6D, a threshold value, $d_{max}$ is used as a maximum point-to-point distance in the initialization of the closest points calculation. This helps disregard points that are too distant to begin with and helps improve the match for largely non-overlapping scans.

Finally, as the map is being created sometimes we are met with a scan that matches one we've seen previously but has been slightly displaced due to the accumulation of error over time. If such a pair was found, ICP can be used once again to register the new scan, then the error calculated can be back propagated into pervious scans to make the change seamless.

## 4 Methods

The National Institute for Standards and Technology has created life size arenas for testing robots' ability in Urban Search and Rescue. To simplify algorithm development and robot

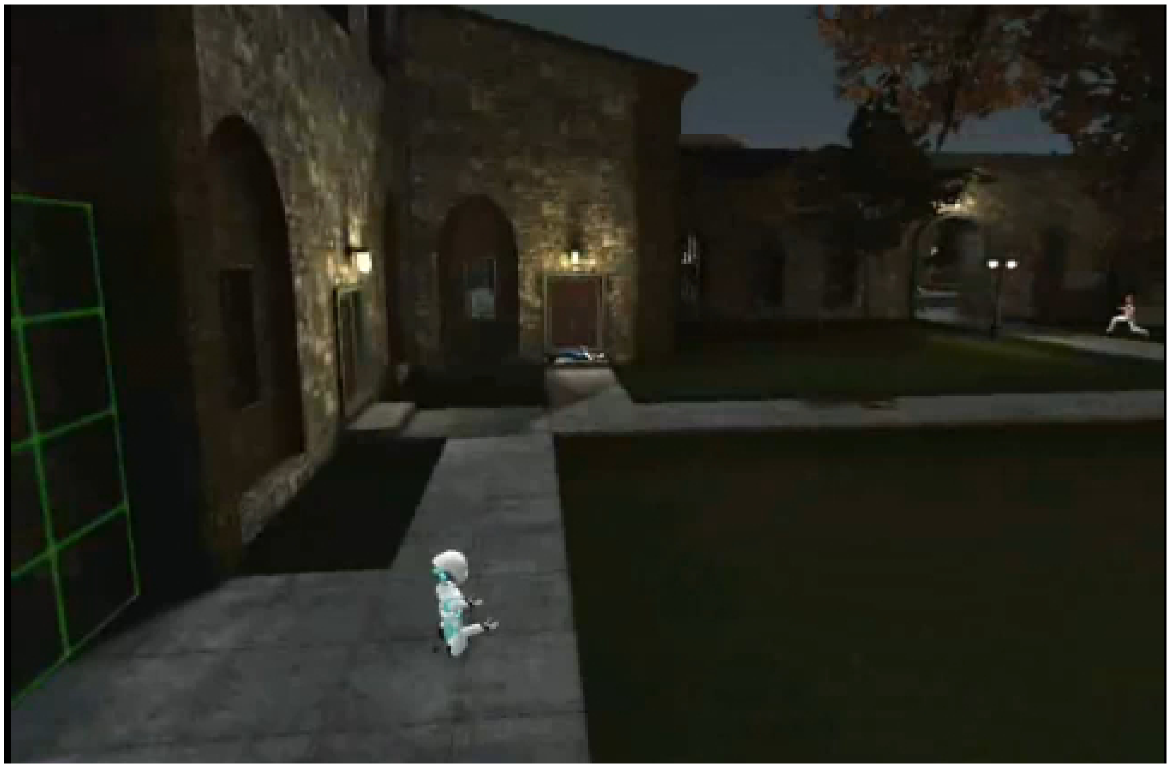


Figure 4: Screenshot of the MDS robot in the USARSim Environment.

testing, the Urban Search and Rescue Simulator (USARSim) has been developed, which contains simulations of those arenas for testing simulated versions of the robots in the same arenas. The simulator is built on top of the Unreal Tournament Video game engine, which provides accurate 3D graphics as well as a physics engine for modeling body dynamics and gravity. As you can see in Figure 4, this allows for realistic rendering of robots to learn about their interaction with the environment and with other robots without actually using the real robots.

We use USARSim to simulate the MDS robot along with several sensors on board for the sake of data collection. Our model robot has on board a simulated Hokuyo scanner and a SwissRanger 3000 ToF camera, with built-in gaussian measurement errors and configurations set to match the real sensors. The robot is run in one of the maps

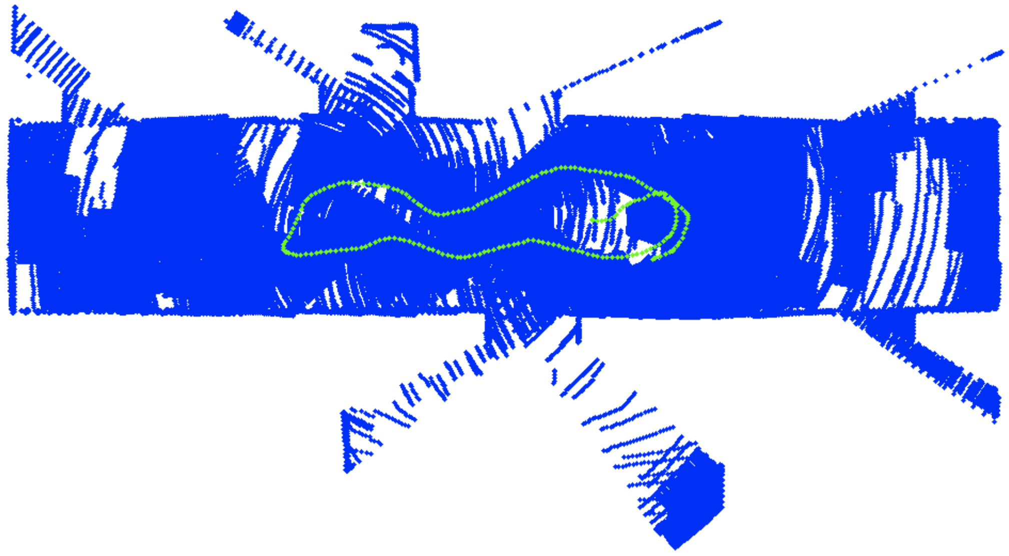


Figure 5: Top view of the groundtruth map overlaid with path followed.

provided with the USARSim, and Figure 5 above shows the groundtruth map with the path followed overlaid on it. The robot was driven around a loop in a corridor, with some of the scans reaching into side rooms, but only one or two per room which is not sufficient to get a more full image of each of the rooms.

The odometry estimate and laser reading were recorded every 0.2s, while the 3D information was sampled at 1hz. The goal is to explore whether 2D localization is enough to build a nice 3D map, or is 6D SLAM needed. Note that to enable the use of 2D SLAM, we necessarily had to have the robot moving on planar ground.

## 4.1 Mapping with GMapping

GMapping is an implementation of the Rao-Blackwellized particle filter which was created by Grisetti, Stachniss and Burgard [4], and which learns 2D maps from odometry together with the data collected by the Hokuyo laser scanner. It was run online, while the robot is running in simulation, and output the list of poses $x_{1:t} = x_1, \ldots, x_t$ for the trajectory followed by the robot as calculated using the 2D SLAM algorithm. It also outputs a map, but we did not utilize the 2D map.

After the run, the trajectory information calculated in GMapping is used to plot the 3D range data collected by the SwissRanger to create a 3D map. No additional processing of either the trajectory or the range information is done, so this map completely depends on the 2D SLAM localization estimates.

## 4.2 Mapping with SLAM6D

To create the 3D map using 6D SLAM, we use the very comprehensive SLAM6D package developed by Nuechter et al. [5] we use K-D trees to speed up the calculation of closest point pairs and a $d_{max} = 0.1$ to make sure we only consider the relevant points The maximum number of iterations per registration was set at 50, and the loop closing calculation using unit quaternions.

Once the pairs of closest points are calculated, the Iterative Closest Point algorithm is used to register consecutive sets of data into one universal frame using the odometry as initial estimate of the relative displacement. ICP calculates the appropriate transformation that allows the robot to calculate its revised pose at each step. Thus the guess for the robot's world location is employed in the calculation and is revised based on its outcome. Each run of ICP provides a reasonable registration of the scan into the global framework, but errors accumulate over time. As the goal is to create a globally-accurate registration of many readings as the robot moves over time, we use loop detection and closing, which looks for pairs of non-sequential scans that are physically close and finds the best transformations to minimize the error, then unit quaternions are employed to interpolate the error calculated so as to spread it out over all intervening readings.

Table 1: Average and Maximum RMS Error for each of the maps produced

| | Average RMS Error | Max RMS Error |
|---|---|---|
| Odometry | 0.2286 | 0.6483 |
| GMapping | 0.3561 | 1.0309 |
| SLAM6D Before Closed Loop | 0.7644 | 2.1031 |
| SLAM6D After Closed Loop | 0.5572 | 1.7434 |

The SLAM6D algorithm is particularly efficient in the calculations it does with large point clouds, and does not create redundant information so as to streamline its execution, instead only modifying the poses in the trajectory. Thus it avoids extra memory requirements by only saving the pose of the robot and using the same set of 3D data

## 4.3 Map Evaluation

For evaluation, we compare the Root Mean Square error of each scan in each of GMapping and SLAM6D against the saved ground truth values. The RMS error for a given scan $s$ versus the corresponding one in the groundtruth map $g$ is

$$e_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} r_i^2} \tag{5}$$

where N is the number of points in the scan and $r_i^2 = (x_{i_s} - x_{i_g})^2 + (y_{i_s} - y_{i_g})^2 + (z_{i_s} - z_{i_g})^2$ is the Euclidean distance metric for 3D. Figure 6 shows a plot of the RMS error over all scans for pure odometry, GMapping and 6D SLAM.

To have a more discrete comparison between the various mapping techniques, we also consider the average and maximum RMS Error for each map as can be seen in table 1.

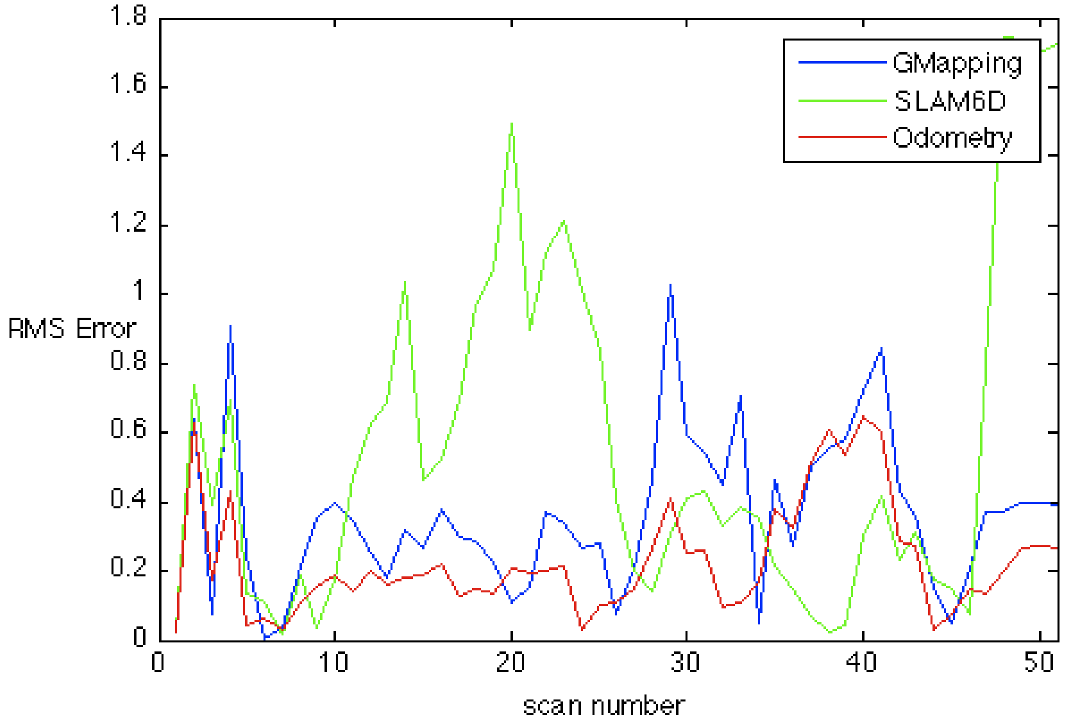


Figure 6: RMS Error.

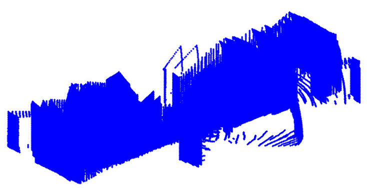
## 5 Discussion

Because of the nature of this experiment, we had to limit the environment to a flat ground so that the 2D SLAM algorithm can work. This run was brief and in a very structured environment, so the odometry estimate by itself was not bad.

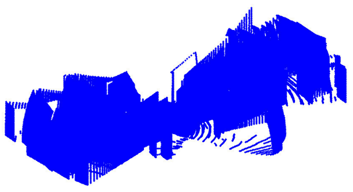
Looking at the produced maps in Figure 7, we can see that both SLAM maps exhibit large errors due to being badly aligned with the true map.

### 5.1 GMapping 3D Map

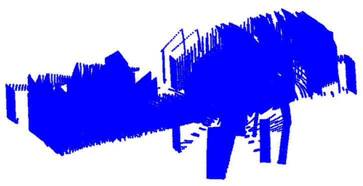
The map produced by GMapping follows very closely the odometry estimates. It is aligned to the groundtruth map and therefore the error is roughly constant. GMapping provides a very efficient way to calculate the map using a 2D grid and Rao-Blackwellized particles



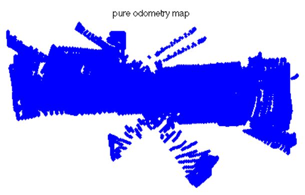
(a) Plain Odometry Map



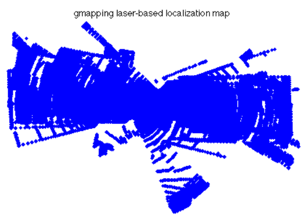
(b) GMapping-based Localization Map



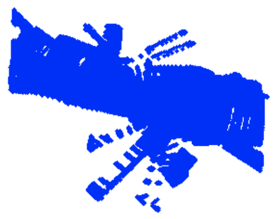
(c) 6D SLAM ICP Map



(d) Plain Odometry Map top view



(e) GMapping-based Localization Map top view



(f) 6D SLAM ICP Map top view

Figure 7: Maps Created.

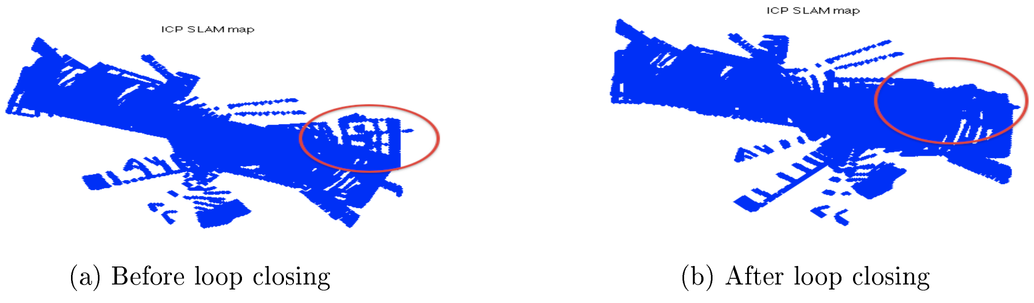
as discussed above. Due to the nature of the particle filter, we only track the localization of the best particle at each time step. This best particle may have changed during the algorithm execution so even though it finds the highest probability path overall it is not necessarily the most likely at each time step, as one we see in Figure 7e

We don't see much variance from the odometry due to the fact that the simulation wasn't run long enough for odometry to deviate much from ground truth.

## 5.2 SLAM6D Map

Early on in the 6D SLAM run, around scan 10, we see a large jump in the data due to a quick rotation. This causes the algorithm to lose its bearings for most of the run. A chance for loop closing is encountered in scan 45, and we see a corresponding dip in the RMS error when it closes a loop, returning to initial error rate, and achieves a rate lower than both GMapping and odometry for the 20 steps prior to that point due to interpolation.

Figure 8 shows the effect of loop closing on the map; the circled part in both sub-images is where the loop was detected and closed. This shows that loop closing is an important part of 6D SLAM, particularly because a single incorrect registration skews the remainder of the readings, as can be seen in 8a.



(a) Before loop closing

(b) After loop closing

Figure 8: The top view of the point-cloud rendering of the map from our run. 8a is the map before applying loop closing, and 8b is after.

Because the simulator renders data more slowly than actual robot, we were forced to use lower frequency when collecting 3D sensor data, and the results would probably be better if collected at higher resolution. Additionally, recall that the 3D data was collected at 1hz, whereas the odometry and laser data at 5hz, and that the SwissRanger can take up to 20 fps. However, when considering the speed of the 6D SLAM algorithm, which takes approximately 0.738s to register a pair of scans when using the K-D trees to optimize the search for closest points, the information from the SwissRanger can't all be used toward coming up with a better map. Therefore the speed of the robot has to be monitored and capped so as to collect good information. It does not have to make a complete stop for each scan, but should not exceed a certain speed to make sure that the scans still have significant overlap so as to not register scans incorrectly. In order to improve the resolution or speed, we would need to utilize subsampling techniques for the data, which are discussed below.

ICP converges to the local minimum, so if the starting point is a good estimate that works well, but it cannot deal with too large of an error (significantly larger than data precision) unless there is a distinct feature, such as 3D corner or some kind of protrusion, to help the fit. For now, it is the best way employed in 3D mapping, and is often reinforced by other checks. ICP is a robust method that is easily applicable and extendible to many areas, including scan matching for 2D SLAM.

Because ICP is very sensitive to starting conditions, the initial relative displacement estimate has to be reasonably close to it to come up with a good result. In this case we've found that odometry reading is a good enough starting point, given the threshold $d_{max}$ was set at 0.1.

The resolution used in the simulation is even lower than that of the SwissRanger ($88 \times 73$ vs. $176 \times 144$), and we've seen that it produced a satisfactory map comparable to the one from 2D SLAM. Thus the full resolution sensor would likely produce good enough results for a good map without being too computationally intensive. However, it has to be carefully configured and calibrated to set an integration time and a maximum distance past which the values are clipped.

## 5.3 Comparison

The SLAM6D map is a better match, but it's tilted off center. This is because the registration mismatch causes all the following scans to be registered in the slightly altered reference frame, and the loop closing step fit the scans into that altered frame. Accordingly, the frequency of the scans has to be based on the speed of the robot, so as to assure having overlapping scans to facilitate finding closest points and the optimal transformation. On the other hand, the particle filter approach used for 2D SLAM is more robust to such errors by continuously resampling the space and giving higher weight by taking the sensor input into consideration.

Both algorithms start off with the odometry estimates as initial localization estimates, but even though ICP is guided by odometry, it really deviates. On the other hand, to nature of GMapping, it follows the odometry readings very closely, whereas ICP has a better chance of splitting off as was seen, which in this case was for the worst, but allows it to disregard incorrect location sensor data and go purely by how well consecutive scans match one another. Particle filter places more trust on the sensors, whereas SLAM6D looks for more general feature similarities.

Looking at the RMS Error values from Table 1, we can see, as expected that odometry produced the best results. Also, we can see that the loop closing step made a significant improvement to the RMS Error values of the SLAM6D algorithm. Although comparable, the Error values for SLAM6D are somewhat larger than those for GMapping.

## 6 Conslusion

6D SLAM can be improved by increasing the sampling rate- taking 3D scans more frequently. This would make for a more significant amount of overlap between successive scans to help improve the ICP estimate. Other ways to improve 6D SLAM would be by subsampling the space and choosing the points more cleverly [3]. Some ways to choose subsamples cleverly is to include a set representative of the gradient of the data, or that has very distinctive $n_{th}$ derivatives. If more information is known about the area being looked at than the distances alone (e.g. more sensors are used), that additional information can be taken into consideration. For example, if the color or brightness of a pixel is known, that can be employed as a metric for filtering relevant points or aiding in the distance calculation together with the straightforward Euclidean distance.

We were able to show that 6D SLAM works roughly as well as traditional 2D SLAM in a flat environment, and more work will be needed to check its accuracy in more general environments.

Some extensions for this exploration to provide more comprehensive testing:

1. Run the robot for some time before collecting any data.. allowing the odometry estimates to drift.

2. Deploy the experiment in more diverse maps, including non-flat ground, to see the effect of differing types of environments and features.

3. Try different resolutions for the 3D scanner, different subsampling techniques for scan matching, different rates of data collection

## References

[1] P. J. Besl and N. D. Mckay, *A Method for Registration of 3-D Shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence. 14(2): pp. 239-256, February 1992.

[2] P. Einramhof, S. Olufs, and M. Vincze, *Experimental Evaluation of State of the Art 3D-Sensors for Mobile Robot Navigation.* In: 31st AAPR/OAGM Workshop (2007) .

[3] N. Gelfand et al., *Geometrically Stable Sampling for the ICP Algorithm*, Fourth International Conference on 3D Digital Imaging and Modeling (3DIM 2003), pp. 260-267.

[4] G. Grisetti, C. Stachniss and W. Burgard, *Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters*, IEEE Transactions on Robotics 23(1) (2007)

[5] A. Nüchter, *3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*. Springer-Verlag, 2009.

[6] S. Rusinkiewicz and M. Levoy, *Efficient Variants of the ICP Algorithm*, Third International Conference on 3D Digital Imaging and Modeling (3DIM 2001), pp. 145-152.

[7] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics*, MIT Press (2005).