Jacobs University Bremen

Robotics Project
Spring 2011
School of Engineering and Science

Motion Compensation for Single Beamed Sound Navigation and Ranging Sensor

---

Mohammad Faisal

mfaisal_abdul@hotmail.com
Electrical Engineering and Computer Science
Jacobs University Bremen
Germany

May 13, 2011
Department of Computer Science
Jacobs University Bremen
Prof. Dr. Andreas Birk

Contents

List of Figures

Abstract: Single beams SONAR works well with a stop and scan method; however, this is usually not feasible when dealing with AUVs. This paper looks at approaches to solve the motion compensation problem in data acquired from single beamed SONAR in a moving AUV. This paper also presents a novel approach for motion compensation for a single beamed SONAR sensor. The approach relies on high accuracy in terms of localization for each scan line acquired from the SONAR sensor of the AUV. The approach presented in this paper relies on IMU data and input voltage data for the thrusters to calculate the pose. The results have shown that a combination of the two leads to very good results in terms of localization, and therefore in motion compensation of acquired SONAR data.

1 Introduction

Less than 30% of the world is land without water. As the population and the technology have grown exponentially in the past two-three decades, much of this land has been occupied or explored. Soon much of the remaining unexplored land would also be covered with settlements, farmlands, or factories. This has increased important of putting the remaining 70% of the world, which is under water, to use.

SONAR sensors are the biggest asset in the market currently to make it possible to explore the seas. This is used to map the terrain which can then be used in establishing underwater pipes, study feasibility of underwater construction, and by oceanographers to access the terrain.

SONAR stands for *Sound Navigation and Ranging*. It works by sending a sound underwater, and measuring the time it takes for it to reflect and travel back. Since sound waves are much slower than light which is used in laser scanners, each sonar scan takes significantly longer to register. Therefore sonar is traditionally used in "stop and scan" mode. This is where the sonar sensor is held in a fixed location, while it completes a 360° scan. The sensor is then moved to a new location and a new scan is taken.

This is a very inefficient way to gather data as a lot of energy and time is wasted in stopping and starting again. Ideally sonar should be able to be used as it is moving, without distortion on the output data. While the sonar was used with boats and ships, this problem was solved using GPS to track when and where each sonar scan was taken. This information was then used to fix the distortion in the data caused by the movement of the scan. However, as *autonomous underwater vehicles* (**AUVs**) are getting cheaper and more accessible, this problem has cropped up again.

This project deals with methods to accommodate the motion in sonar scans obtained underwater using AUVs for the purpose of terrain mapping. Section 2 describes the expected time-line for the project for spring 2011 semester. Section 3 formally describes the problem we are aiming to solve during the course of this project. Section 4 mentions the different state of the art approaches which could be taken, and are being used, to fix the given problem. Section 5 describes the approach chosen in this project, and the details behind the choice. This is followed by implementation of the approach has results in section 6. Section 7 gives a summary of the paper, and high-lights what future work can be done.

2 Time-line

The expected time line is shown in figure 1 and in figure 2. The details of the different phases mentioned are given on the next page.

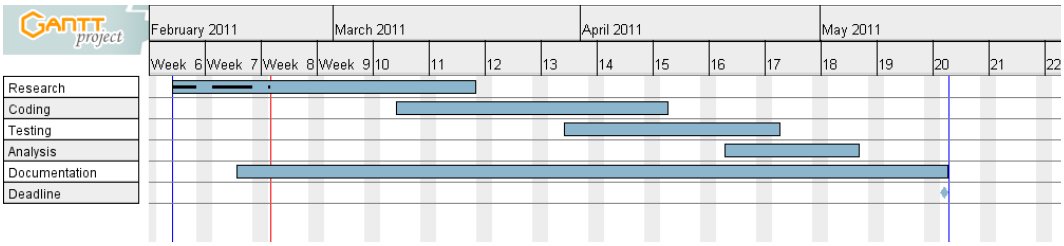


Figure 1: Expected Time-line

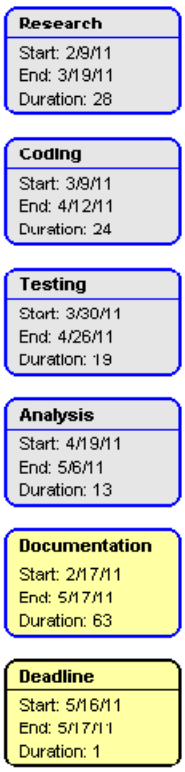


Figure 2: Expected Time-line

The **research** phase includes research about different methods for motion compensation in terrain mapping with sonar. These include general approaches, based on single sonar sensor as well as customized sonar hardware.

The **coding** phase includes implementation of the selected approach. This may or may not include actually use of new code. This includes implementation of the approach to an existing system such that the approach can be evaluated against normal conditions.

The **testing** phase is used to solve any problems encountered in the implementation phase. These can include ordering of any required hardware part, or debugging of the code, to make sure that the implemented approach is working as it is supposed to.

The **analysis** phase is used to compare the new approach against normal conditions. This could also include comparing multiple approaches if multiple approaches have been implemented. The goal is to find the advantages and disadvantages of the new approach and the final effect of the new method on the scan achieved through a sonar scan on a moving UAV.

The **documentation** phase represents the time-frame used to document all the different phases of the project. This includes the information discovered during research phased, the technical details of the implementation and testing, the troubleshoot information from testing, the techniques used in analysis, and finally documenting the results of the entire project.

3 Problem Definitions

SONAR stands for *Sound Navigation and Ranging*. A sonar device is used to detect objects in water using sound. It works by sending a sound underwater, and measuring the time it takes for it to reflect and travel back. This is illustrated in figure 3. Most common type of sensor used is Single beamed Sonar. Therefore multiples measurements are required to complete a 360° scan. This requires the use of a "stop and scan" approach. If the sensor is moving, the data gathered in a complete scan will be distorted by the movement. This is a very inefficient way to gather data as a lot of energy and time is wasted in stopping and starting again. Ideally sonar should be able to be used as it is moving, without distortion on the output data.

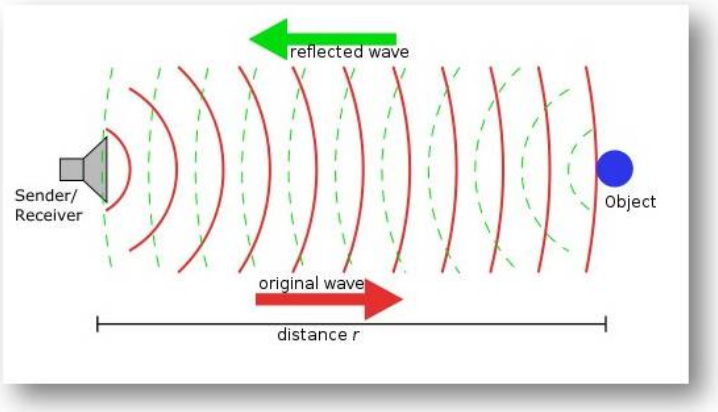


Figure 3: How SONAR works.

There are two different types of sonar sensors: Active and Passive. **Passive** sonar is a listening device, which does not produce any sound itself, but just listens to sounds in water. It is basically a receiver. **Active** sonar is a device which measures distance to an object by first producing a sound of a specific frequency and then listening for it, while measuring the time it takes for it to arrive back.

Single beamed Sonar has been around for a long time. This works by gathering distance measurements in a particular direction in a **scan line**. The original purpose of these was to detect objects underwater by sending a ping. However, these cover very small area with each ping and therefore require large time to cover a larger area. Furthermore, if the sensor is moving while it is making a complete circle for mapping, it leads to distortion. Therefore single beamed sonar has not been feasible in terms of time, and therefore money, for mapping large areas.

These are, nevertheless, most commonly used in fishing and marine surveying. These boats use differential GPS (Global Positioning System) to keep track of its location at each point. This information is then combined with the sonar data to compensate this movement and plot an accurate map of the terrain. An example of such operation is shown in figure 4.
However, GPS cannot be used underwater by a UAV. Hence a better approach is required for underwater mapping.

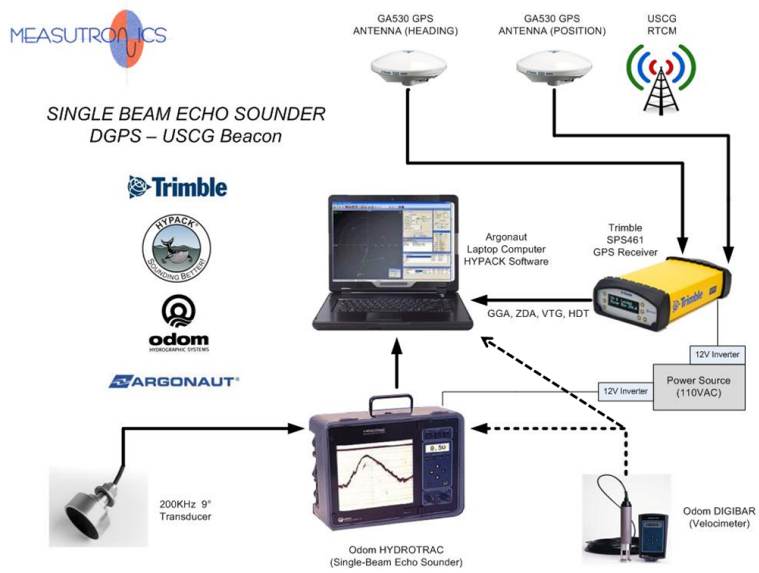


Figure 4: Using Single beamed Sonar with a GPS [1]

4 Existing Approaches

4.1 Multi beamed Sonar

As mentioned in section 3, one of the biggest problems with mapping the terrain underwater using single beam sonar is that to get an accurate map, the process is very time consuming. It takes the single beam sonar a lot of time to scan a large area since it does so one ping at a time, and it cannot send a new ping until the previous one is registered. A possible solution to solve this problem is the use of a Multi-beam SONAR. **Multi-beam** sonar is an instrument that can map a larger area, with a higher resolution, than single beamed sonar, using multiple beams [2]. It is basically an array of numerous single-beamed sonar. All these components of the array takes the scan at the same time. This way, in one scan, they can measure a continuous area which would take a single beamed sonar number of scans to measure. Since all of this happens in one scan, there is no distortion due to motion in plotting them. Subsequent scans can be joined together using feature detection such as sift features to create a global map [4]. An example of how multi-beamed Sonar works is shown in figure 5.
It is obvious from the figure 5 that multi-beam sonar is much better than single beamed sonar (being 8 times faster in the above image) and essentially solves the problem with motion compensation. Multi-beam sonar are actually up to 100 times faster than single beamed sonar, with SEA BEAM 2100 multi-beam sonar plotting 151 beams with one ping and allowing it to cover tens of kilometers at depths of a few kilometer [2]. However, this increased complexity and efficiency comes at a great price. Multi-beam sonar costs a lot more than single beamed sonar.

4.2 Side Scan Sonar

The single beamed sonar and the multi beam sonar both rely on using the difference in distance to the terrain to map the terrain. **Side Scan sonar** on the other hand, takes advantage of the sound absorbing and reflecting properties of the sea floor to map the terrain. Materials such as metals and rocks have higher reflection of acoustic pulses, whereas clay and silt does

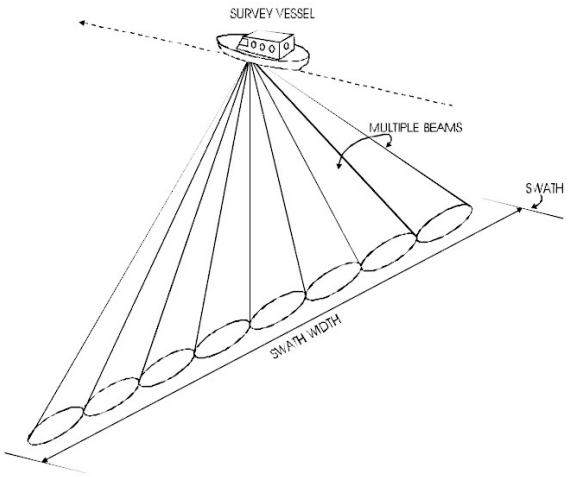


Figure 5: Multi beam Sonar operation [2]

not [2]. This creates difference in the strength of echoes, which is interpreted by the side scan sonar to map the terrain. The operation of side scan sonar is shown below:

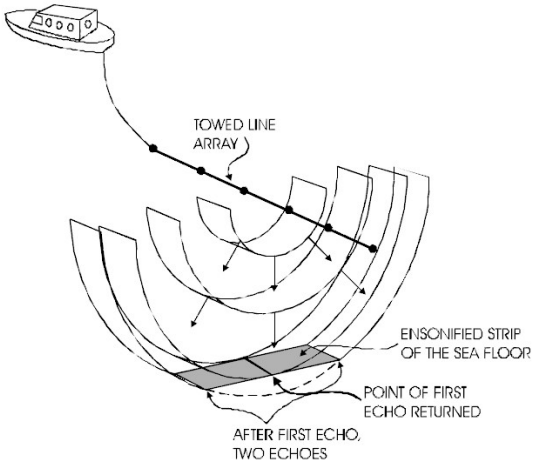


Figure 6: Side Scan Sonar Operation

However, the side scan sonar model is only efficient for flat surfaces. Since actual sea-bed is not always very flat, the mapping can fail. This is because a single pulse is used to send the data, with multiple receivers to listen to the data. To solve this, the idea of **Synthetic aperture sonar** (SAS) is used. This approach utilizes correlations between the signals obtained through the different receivers coming from the same illuminated zone. These receivers are placed equidistant along the length of the sonar array. This approach can be used by making the entire UAV into side scan sonar. This requires purchase of specialized equipment and costs much more than single beamed sonar.

4.3 Single beamed Sonar

As mentioned in section 3, single beamed sonar have been in used for a long time, and are being used in shipping industry and other industry for surveying and mapping purposes. This is done by using a GPS to relate the sensor data from the SONAR to the location it was taken

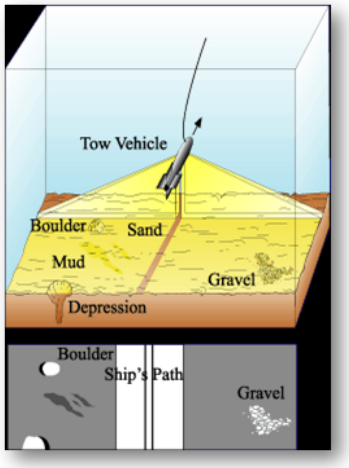


Figure 7: Side Scan Sonar Results

at. An example of such a system has been shown in figure 4. However, it is not possible to use the GPS underwater. Nevertheless, this approach relies on the same basic principles. Instead of GPS, different sensors and actuators can be used for localization of the UAV underwater to be able to keep track of its location at each instant when a sonar scan line is captured [5].

5 Our Approach

5.1 Reasons

One of the major goals behind this project is to come up with a solution which is financially and technically feasible in the span of 3 months. Since the Jacobs University Robotics group already has an AUV with a single beamed SONAR, it was not financially feasible to order custom hardware discussed in section 4.2 and 4.1. Therefore the approach mention in section 4.3 was used. Furthermore, choosing a software based solution, in conjunction with single beamed sonar, would make it easily implementable in different scenarios by larger number of people. The approach is cheap and easy to implement, and can still be used with multi-beamed sonar sensors.

5.2 Technical Details

The framework used in the implementation is ROS[1]. ROS provides operating system services such as hardware abstraction, low-level device control and package management. It is based on graph architecture where processing takes places in nodes. This node represents sensors and actuators, and different application software. The approach will be implemented as a ROS package with a node which will listen to sensor data, and publish pose estimates. This way it can be used in conjunction with already working data, with no modification to the existing code. The publish pose can be used in real-time to plot compensated sonar data. Additionally, the node also writes the pose estimates in a file which can be used later to plot the path, or view the sonar data.

5.3 Structure of approach

The approach will use a number of sensor readings from the AUV, and use them to calculate the pose estimate. The sensor data used are:

1. IMU
2. Visual Data
3. Input Voltage

Details on how each of these will work are given below:

5.3.1 IMU

IMU stands for *inertial measurement unit*. It is a device which uses accelerometers and gyroscopes to give acceleration, orientation and angular velocity of the device. The accelerometers are placed so that their measuring axes are orthogonal to each other, to for the three axes: x-axis, y-axis and z-axis [6]. It is this acceleration which is used to calculate the speed of the device, hence the AUV to which the device is attached. This is best illustrated in figure 8.

[1] http://www.ros.org/wiki/

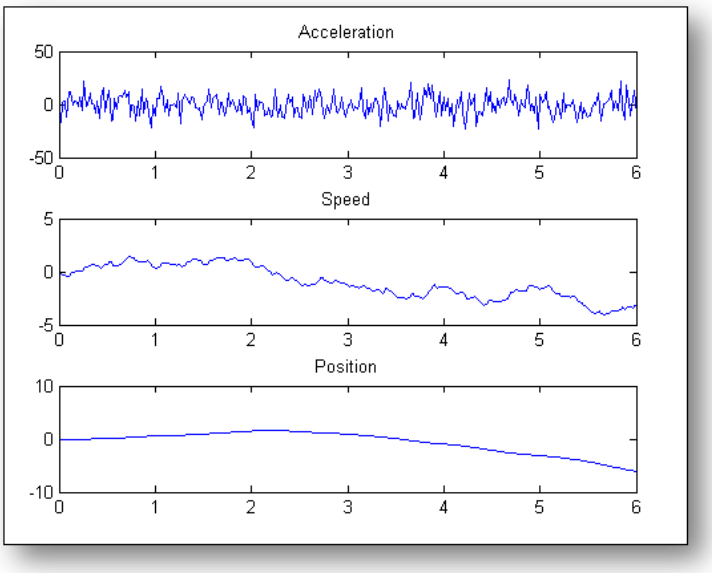


Figure 8: Top: Acceleration data (from IMU); Middle: Speed data (integrated acceleration); Bottom: Distance (integrated speed)

The top graph shows the acceleration in one axis. This data is integrated to get the velocity. Since integration of the value requires analog values, this integration is done by calculating the area under the curve. Suppose at t_0 and t_1, we have a_0 and a_1 acceleration respectively. The velocity gain during the interval $\delta t = t_1 - t_0$ is calculated as shown below:

$$change\_velocity = \frac{(a_0 + a_1)}{2} \times \delta t \tag{1}$$

where $\delta t = t_1 - t_0$. This gives us the velocity at t_1 as:

$$velocity_1 = change\_velocity + velocity_0 \tag{2}$$

This velocity is then integrated to get the position. Again, in absence of any analog data, we approximate the integration as follow:

$$change\_position = velocity_1 \times \delta t - \frac{(\frac{(a_0+a_1)}{2} \times \delta t^2)}{2} \tag{3}$$

where $\delta t = t_1 - t_0$. This gives the distance traveled between time t_0 and time t_1. To calculate the today distance traveled, we add the change in distance to the previous distance traveled:

$$position_1 = position_0 + change\_position \tag{4}$$

This way it is possible to track the position of the device in one axis.

It is worth mentioning at this time that the IMU gives the orientation of the device, relative to its own coordinate frame, in quaternion. These are transformed using built-in ROS functions.

5.3.2 Visual Odometry

Underwater cameras have been in use for decades.The iconic images of the Titanic from 1985 have demonstrated the usefulness of underwater cameras. Underwater cameras nowadays are cheap and easy to install, and after some calibration, are the best tool to visualize the robot's environment. One method to use in underwater localization is visual odometry. **Odometry** is the technique to determine change in position of an object, using the sensor data [7]. The simplest example of odometry is using a wheel. Suppose we know the diameter (d) of a wheel, and know that in a given time t, the wheel rotated x times. We can then calculate the distance traveled as:

$$change\_distance = x \times \pi \times d \tag{5}$$

From this, we can calculate the speed as:

$$speed = \frac{change\_distance}{t} \tag{6}$$

Visual Odometry (VO) is a special kind of odometry. Instead of rotation of the wheel, the position and orientation of an object is calculated using number of associated camera images [8]. Visual odometry has already used numerous times for localization, most notably on the NASA's Mars Exploration Rovers Spirit and Opportunity [7].

The algorithm for the process is quite simple and is shown in figure 9.

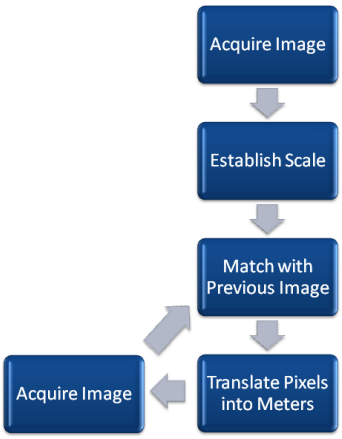


Figure 9: Visual Odometry Algorithm

The first two steps are done in the start of the setup. The reference image is used to create a scale. This is done by using an image with an object for which the dimensions are known. This establishes a pixel to distance scale. After this, an image is acquired and matched against the previous image. This results in transformation between two images in term of pixels. This is then scales back to distance and to get the change in directions in the frame of the camera. An example of this is shown below:

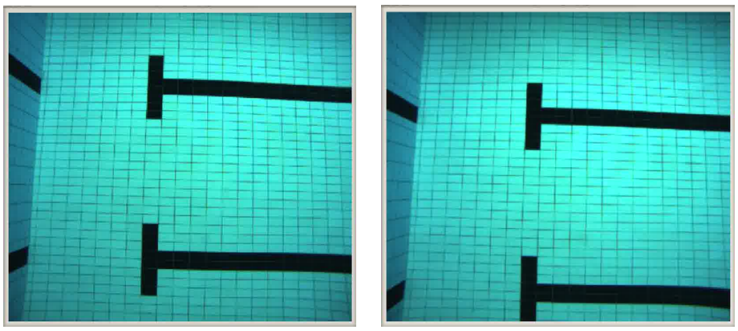


Figure 10: Two consecutive images. Left: Old Image; Right: Current Image

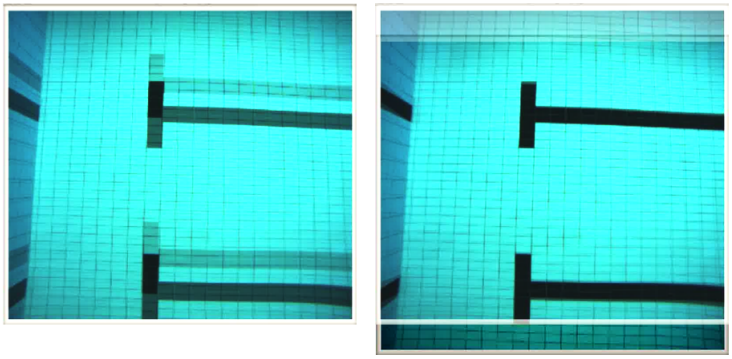


Figure 11: Left: Old and New Images overlapped; Right: Old Image transformed to match New Image

Figure 10 shows two consecutive images acquired as the robot is moving. The old image is on the left, and the new image is on the right. Figure 11 shows the overlapped old and new images on the left. On the right we can see the transformed new image to fit the old image. The transformation in the pixels observed in the right image of figure 11 can then be used to calculate the distance moved between the two images, using the scale established from the reference image.

5.3.3 Input Voltage

As mentioned in section 5.3.2, **Odometry** is the technique to determine change in position of an object, using the sensor data [8]. Instead of sensor data, we can also use the actuator data for this purpose. Coming back to the wheel example explained earlier, if we know that applying a certain voltage v would rotate the motor, and through it the wheel x times, we can calculate the distance traveled if we have the diameter d of the wheel. The change in distance for voltage v is:

$$x = v \times constant \tag{7}$$

$$change\_distance = x \times \pi \times d \tag{8}$$

This was a very simple example. Using the same principles in our problem requires a lot more complex mathematics and physical ideas. The basic idea is very simple and illustrated in figure 12.

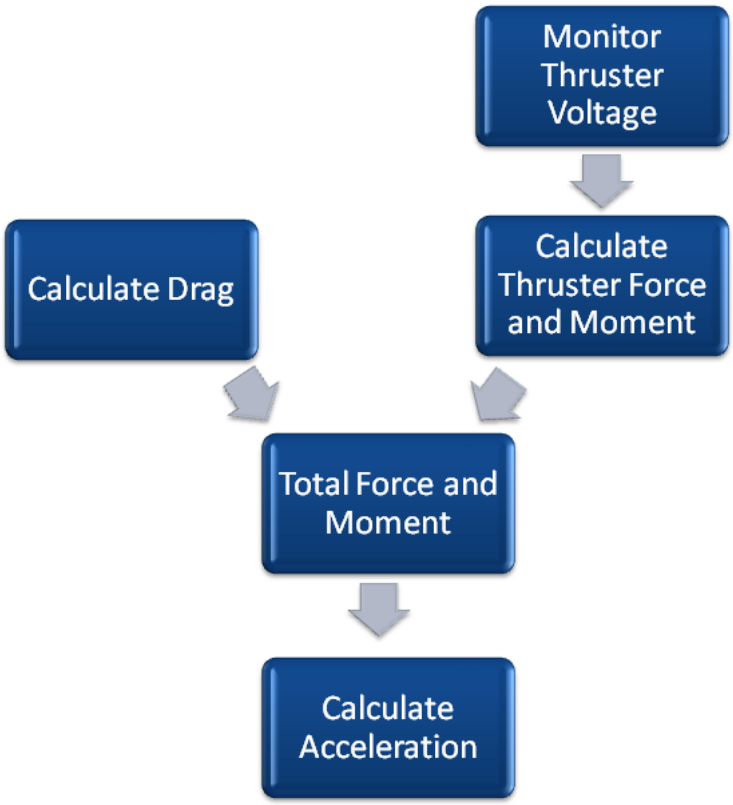


Figure 12: Algorithm for Localization using Thruster Model

We have a UAV with 8 thrusters. We monitor the voltage applied at each thruster to get the total thruster force and moment applied to the UAV. We add drag due to water, and buoyancy to the total force calculation to calculate the resultant force and moment on the UAV. This can then be used to calculate the acceleration on the robot.

Before we look at each part separately, it is important to define the terms which will be used in the following description. /bf Force is defined as the power exerted upon an object, usually to cause a change in speed, direction or shape [9]. It is a vector quantity which has both direction and magnitude. Contributions of force from different sources can be summed to give the net force at any given point [10]. **Moment** us defined as the tendency to produce motion, especially rotational motion, about a point or an axis [11].

Once we have defined these values, each component mentioned above can be calculated. The first one is the total thrust from each thruster. This can be related to the input voltage V using the following equations:

$$thrust = (c_0 + (c_1 \times V)) \tag{9}$$

$$moment = \frac{c_q}{c_t} \times thrust \tag{10}$$

The moment is then modified for each thruster based on its position, to get the overall moment exerted by the thruster on the UAV. Some relative values between thrust and voltage are shown in figure 13. These values are taken from the datasheet of the thrusters being used in the Jacobs UAV [12].

Direction	Voltage(DC)	Current Draw (A)	Thrust (kg)	Power (W)
Forward	12.0	1.91	0.75	22.92
	14.0	2.45	1.00	34.3
	16.0	2.78	1.20	44.48
	18.0	3.32	1.43	59.76
	20.0	3.78	1.65	75.6
	22.0	4.20	1.85	92.4
Reverse	12.0	1.92	0.68	23.04
	14.0	2.42	0.92	33.88
	16.0	2.89	1.15	46.24
	18.0	3.30	1.38	59.4
	20.0	3.75	1.60	75
	22.0	4.23	1.80	93.06

Figure 13: Thruster-Voltage Relationship of the SeaBotix BTD150

However, the following values of the constants for positive and negative voltage were used for the thrusters in the project.

	Positive Voltage	Negative Voltage
c_0	-1.0212	-1.2146
c_1	0.8272	0.8068
c_q	6.9882e-06	5.9056e-06
c_t	5.3558e-04	4.3392e-04

The next component is the drag. This was calculated using the fluid dynamic drag equation shown below:

$$F_{drag} = \frac{1}{2} \times A \times C_D \times v^2 \times \rho \tag{11}$$

where A is the cross-sectional area of the UAV, C_D is the drag coefficient for the UAV, ρ is the density of the liquid and v is the speed of the robot. The values used in the implementation is given below:

Variable	Value
A	$length \times width\ m^2$
C_D	2.1
ρ	1000 kg/m^3

The third component is the buoyancy. This is the force that a fluid exerts on a less dense object [10]. This can be calculated as follows:

$$B = \frac{2gm\rho_f V}{m + \rho_f V} \tag{12}$$

where g is force of gravity, m is mass of the UAV, V is the submerged volume and ρ is the specific density of the liquid.

All these forces can be combined by adding and subtracting them since force is a vector quantity. The resultant force can then be used to calculate the acceleration of the UAV in all 3 axis using $F = m \times a$.

5.4 Global Pose Calculation

However, in our example, the robot is moving in two axes: the x-axis and the y-axis. Before we start calculating the position of the robot, it is important to establish a global coordinate frame, relative to which all the measurements are made. This is necessary because the coordinate frame of the IMU, the camera and the thrusters is fixed to the robot, which may change its orientation. This is better explained through figure 14.

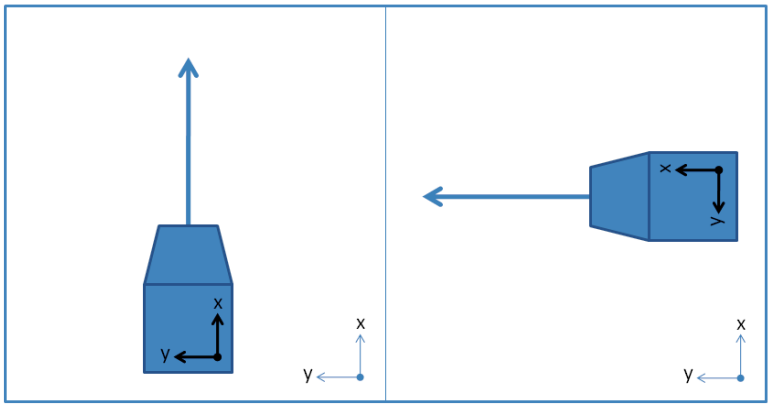


Figure 14: Left: Robot facing straight in the global frame. Right: Robot facing left in the global frame.

In the left image, the robot is facing straight in the global frame, and the x axis of the robot (of the IMU) is aligned with the x axis of global coordinate frame. In the right image, the robot is facing left, aligning its x axis with the y axis of the global coordinate frame. If the robot moves forward in the scenario in the right, we are actually moving left instead of straight, (which the coordinate frame of the IMU suggests. This will give us bad results if we do not keep everything in a global coordinate frame.

As far as the position of the global frame is concerned, the choice is quite simple and obvious. We can have the robot start at position (0, 0). As far as the orientation of the global frame is concerned, since our main goal is eliminate the error associated with sonar data, it makes sense to align the global coordinate frame with the starting position of the robot. This leads us to have an angle offset θ_{ori} which contains the original direction of the robot.

Once we know the starting position of the robot (x_{global_old}, y_{global_old}) in the global coordinate frame, we can now transform the distance traveled by the robot in the local frame, into the global coordinate frame. Let the distance traveled in the local frame be (δx_{local}, δy_{local}) in the direction θ_{yaw}. Now we can transform this into the global distance traveled easily using the following equations:

$$x_{global\_new} = x_{global\_old} + (sin(\theta_{yaw} - \theta_{ori}) \times \delta x_{local}) + (cos(\theta_{yaw} - \theta_{ori}) \times \delta y_{local}) \quad (13)$$

$$y_{global\_new} = y_{global\_old} + (sin(\theta_{yaw} - \theta_{ori}) \times \delta y_{local}) + (cos(\theta_{yaw} - \theta_{ori}) \times \delta x_{local}) \quad (14)$$

Now we have the new location of the robot in the global frame. These equations are used in all transformations in the project, converting the IMU values and the voltage values into the global coordinate frame.

6 Implementation and Results

6.1 Data

All the data used in this project was acquired in the Sportbad Bremen-Nord. The first few data sets were taken in December 2010 and were without Voltage readings. The later datasets were acquired in April 2011, and contained Sonar, IMU, Images and the Voltage readings.

The table below shows information about the different datasets.

Name	Purpose	Approximate Movement (x,y) [m]	Duration [s]	360° *Sonar*
data01	Calibration	0	20	YES
data02	Straight Movement	(23, 4)	90	YES
data03	Straight and Sideways Movementt	(10, 4)	60	YES
data11	Calibration	(0,0)	12	NO
data12	Calibration	(1,0)	40	NO
data13	Calibration	(0,0)	20	NO
data14	Rotation	(0,0)	27	NO
data15	Straight Movement	(12,1)	51	NO
data16	Straight and Turn	(3,4)	40	NO
data20	Calibration	(0,0)	20	NO

The approximate movement is a guess based on visual acquisition of data. In some data sets, 360° sonar scans were used, whereas in the other data sets, 270° sonar scans were used. The dataset data20 was taken while the robot was placed out of the water, on a table.

6.2 IMU

Off the three sensors mentioned, IMU had the most potential because it was a single sensor which could represents all the motion of the robot, from linear acceleration, to orientation, to angular velocity. It also requires the least amount of conversion between coordinates since it is only one sensor. It was therefore the very first sensor to be worked on.

After working on the first three data sets, the results were very mixed. The results produced were accurate in terms of movement, but were always magnified. An example is shown below: As it can be seen, the forward movement of the UAV is well illustrated, but it is also quite magnified, with the robot moving almost twice as much as it actually moved. In addition, in the data01 data set, which was supposed to be of the robot being still, movement was noticed. Because of this, when new data samples were collected in April, three different data sets were taken for calibration. All six datasets taken in April showed overestimation of the distance traveled as well. Since IMU tend to always over-estimate the data, the idea that it might be random noise was dismissed. Subsequently, another dataset was taken with the UAV out of the water and sitting on a still table. The following results were observed:
It can be seen in figure 16(a) that the value of acceleration in x-axis has an offset of -0.166 whereas figure 16(b) shows that acceleration in y-axis has an offset of -0.032. The code was then adjusted to take into account these offsets. The subsequent results were much better as can be seen below:

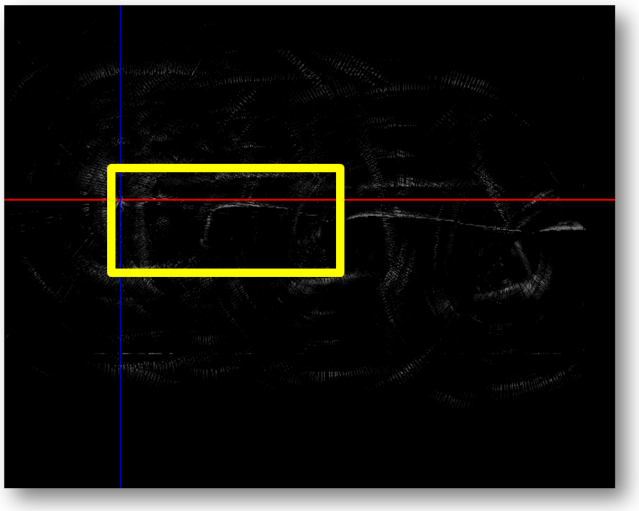
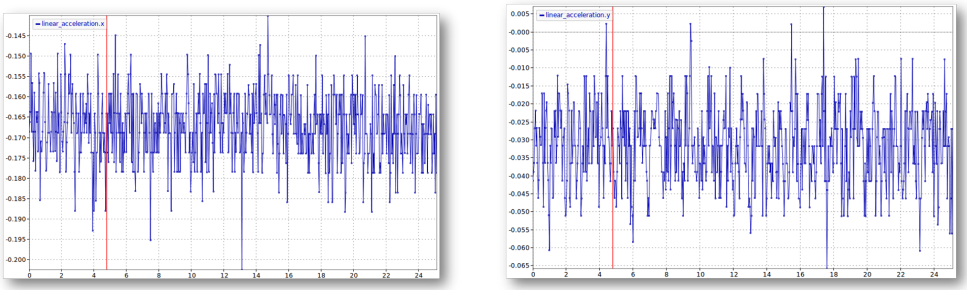


Figure 15: IMU Initial Results + Pool



(a) Linear Acceleration - X Axis

(b) Linear Acceleration - Y Axis

Figure 16: Linear Acceleration of the IMU

Here figure 17 shows the distance traveled by the UAV in x and y axis. Left figure shows that the robot moved more than 4 meters whereas the figure on the right gives a more realistic estimate of 1 meter. Figure 18 shows the effect of reducing the offset on the x-y plot of dataset data16. The red graph shows a more realistic path. One thing to notice is that IMU is not very effective in detecting deceleration. In the actual dataset, the UAV accelerates, slows down, turns, accelerates and then stops, however at the end of the dataset, the UAV seems to have a velocity which is greater than zero. Relying on IMU alone, the path calculated is very good, and the resultant Sonar image is very accurate as shown in figure 6.2.
Another important test was to check the rotation of the UAV, and the readings of the orientation taken from the IMU. Some results of this are shown below:
Figure 20(a) shows the UAV as it is in 0° direction. This is the direction it started from. Figure 20(a) shows the UAV as it is in 180° direction. This can be seen from the black line in the pictures. These pictures were taken on a sloop, and the direction of the sloop illustrated the change in direction observed. These results are highly accurate.

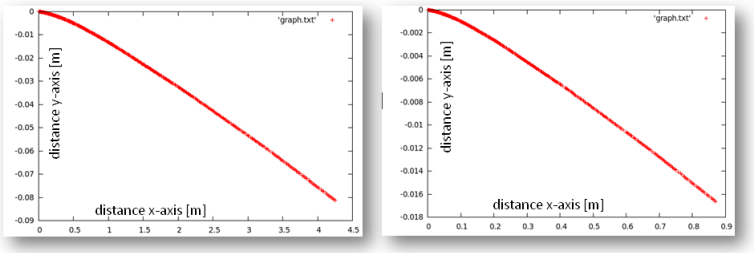


Figure 17: IMU Results with and without offset - data12

Figure 18: IMU Results with and without offset - data16

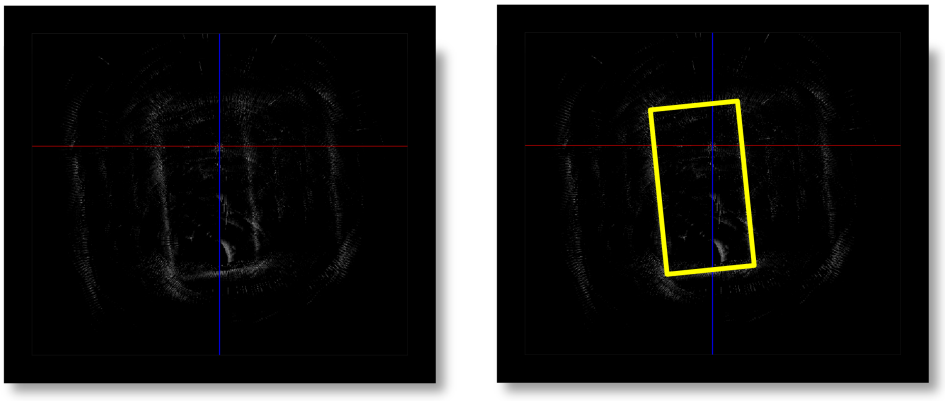
6.3 Visual Odometry

As described in section 5.3.2, visual odometry is a very important method for localization. It has shown a lot of potential, and has been used by NASA on their Mars Exploration Robots. The main principle behind matching of images used by NASA was detection and matching of features. The same principles were used in this project as well. Since ROS already has a package to work with visual odometry, the first implementation was tested with that package[2].

Two data sets were used to generate videos from which frames were extracted and saved in a folder. These images were then used to calculate the transformations in pixels. These values were then analyzed manually to check for consistency. The results were not very encouraging with often failure to find a transform or finding the wrong transform.

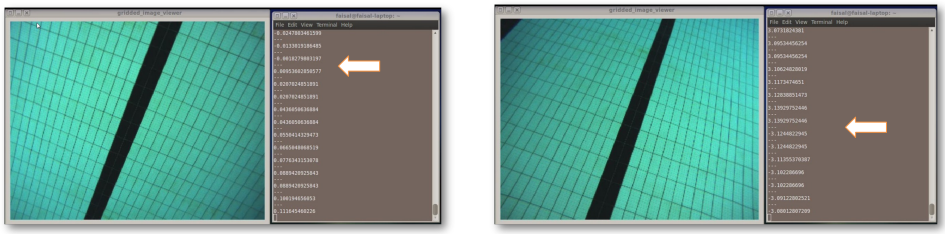
The next step was to use the Jacobs Robotics FMI Scan Matching algorithm to perform the matching between the two scans. The Fourier-Mellin transform (**FMI**) approach, mentioned in [13], was originally designed for scan matching of sonar data. It works by separating the translation estimate from rotational estimate. It works in laser scans by first transforming them into a matrix of size **M**. In our case, each picture is interpreted as the matrix M. A

[2] http://www.ros.org/wiki/visodo



(a) IMU Result (b) IMU Result + Pool

Figure 19: IMU Results 1



(a) 0° (b) 180°

Figure 20: IMU Results Rotation

spectrum is created from this matrix. The rotational estimate is calculated by geometrical re-sampling of the spectrum magnitude to find the matching peaks. The estimate is applied as a 2D shift in the spectrum of the scan and then the translational estimate is calculated. In figure 21 we can observe that dark blue spectrum is a rotated version of the red spectrum, meanwhile light blue spectrum shows a translated spectrum.

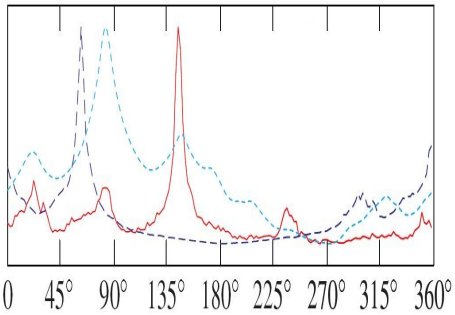


Figure 21: Laser Scan Spectrum [3]

Again, the results were very bad. The scan matcher failed in performing correct matching. The failure of both of these techniques to correctly match the images could be due to number of reasons. Few of them are mentioned below:

1. Variable Distance: One of the biggest problems faced with the first couple of data-sets used for the variable distance to the ground. This is shown in figure 22. Since the algorithm relies on a fixed scale to match pixels with distance, this causes problems when the scale is dynamically changing due to change in distance.

2. Lack of Distinct Features: Another major problem faced was lack of distinct features. The data-sets were taken in a swimming pool where the floor has a very constant design. This is shown in figure 23. The lack of distinct features makes it impossible to know if the movement between two consecutive images is in one direction or another.

3. Distortion and Reflection: Small differences in the pictures also caused a problem. In the data sets acquired in April, a strong reflection was observed in some images where the robot was above the black strips, which may have caused problems with matching images. The distortion observed in corners might also have made it harder to match these images.

4. Moving Objects: This one was present in only one data-set, but extending this approach to real world data would make this problem very significant. Objects such as animals, sea-plants constantly move and sway in the underwater currents which makes matching of consecutive images quite difficult.

Figure 22: Variable distance in the Pool

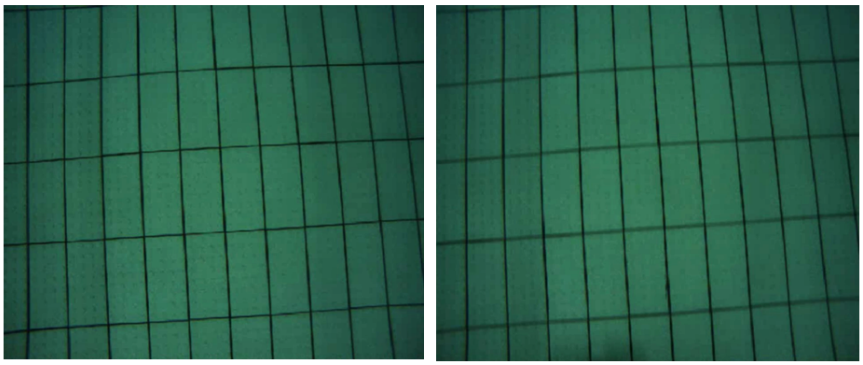


Figure 23: Lack of distinct features in the datasets

6.4 Voltage Input

Creating a model of all the forces acting on the UAV was one of the toughest tasks. This is because it required the use of number of parameters which were supposed to be calculated manually, or required a lot of coding (creating the voltage-thruster model with corresponding values in a switch statement). The main working was done in a loop which was called by a frequency of 30 Hz to calculate the forces at each point and then apply the corresponding acceleration. This acceleration was applied in the similar way as the IMU readings were applied. The main problem with this method was immediately realized when the first few datasets were tested: the parameters being used had to be calibrated, and there were a lot of parameters. Assuming the literature which was basis for many of the equations were correct, I tried changing the area of the drag, as originally the area of the central structure of the UAV was being considered, ignoring the outer frame. However, the thrusters, the camera, the lights and the sonar sensor are other obstacles present within this outer frame. To accommodate this increase in drag, the area of cross section in drag equation was changed. However, the output was still inaccurate. After further testing, and tweaking of the different parameters, a good model of the actual model of the UAV was still elusive. There are a number of reasons

why the theoretical model and parameters did not work. The very first one is the complex structure of the UAV, compared to the simple rectangular model which is used to calculate the cross sectional area. This makes it extremely hard to estimate the correct area. Furthermore, viscous drag (tendency of the fluid to stick to the surface of the UAV, and cause drag) is not considered. This could further add drag. It is also assumed in the theoretical part that the robot is moving in still water, which is an unrealistic assumption given the reflection of the water waves in an enclosed space such as a swimming pool. Lastly, the UAV was not fully submerged in the datasets, and about 10 % of it was protruding out of water. This may cause additional drag. It was therefore decided to simplify the model. The new simplified model

will have three constant: one for relating the input voltage to acceleration, one for relating drag to deceleration and a third one for relating negative voltage for deceleration. This made the tweaking of the parameters easier. The dataset data15 and data16 were used for tweaking of the parameters as it involves a long acceleration, deceleration till the robot is stopped and is long enough to observe drag. The results were very accurate in straight motion as can be observed in figure 24.
However, in turns, the voltage model did not perform well because the simple model does not take into account the position of the thrusters, and hence the different moment imparted by each on the UAV. This is most prominent in figure 25
As can be seen, when dealing with turns, the rotation is not considered properly.

6.5 Best of Both Worlds

As seen in figure 24, the simple voltage model is very nice representation of the real UAV in dealing with straight motion, and in figure 6.2, the IMU is great with orientation. The next idea was to combine the magnitude from the voltage, and the direction from the IMU. The results were, as expected, better than the individual results.
Figure 26 shows that the orientation and voltage combined show similar results to simplified voltage alone in the straight section.
Figure 27 shows that the orientation and voltage combined solve the scenario with IMU alone where the UAV does not decelerate. In both images, the UAV slows down and then stops.

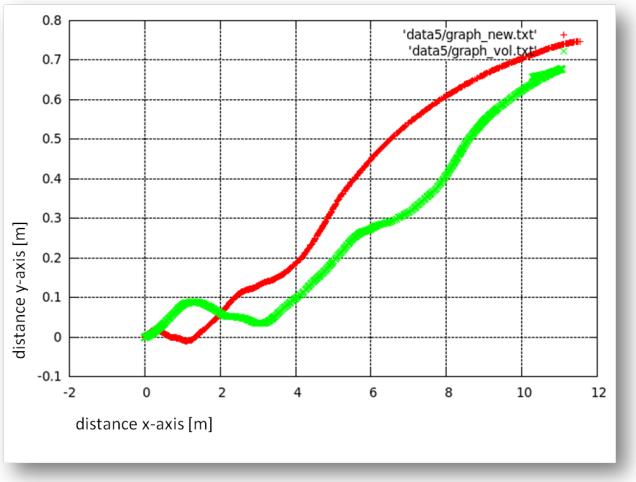


Figure 24: Simple Voltage Model - Straight

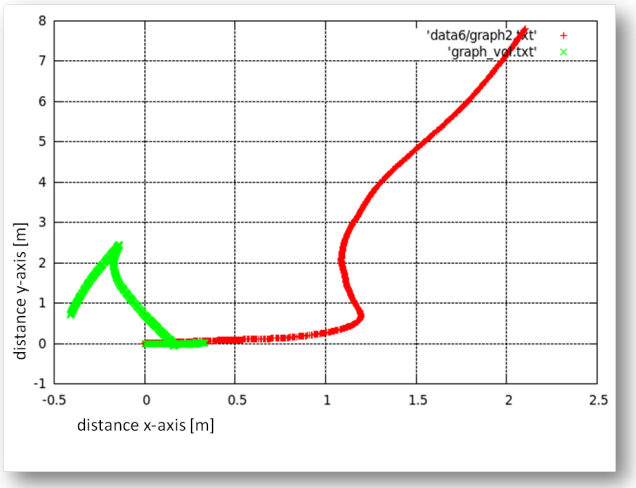


Figure 25: Simple Voltage Model - Turns

6.6 Structure of Solution

One of the best things about working in ROS is that all packages can work independent of each other, as long as the topics being subscribed and published are the same. This is where the ROS package created in this project will come in handy. There already exists a launch file which launches all the sensor and actuator drivers. Now all that is needed is to add the launch command of the created node in that file, and change the sonar viewer node to subscribe to "sonarpose" topic instead and the user would be able to view the motion compensated sonar as it arrives. In addition, the node is also writing the frame id of each sonar and the associated pose into a file which can then be read later in time to view the sonar as it was recorded.

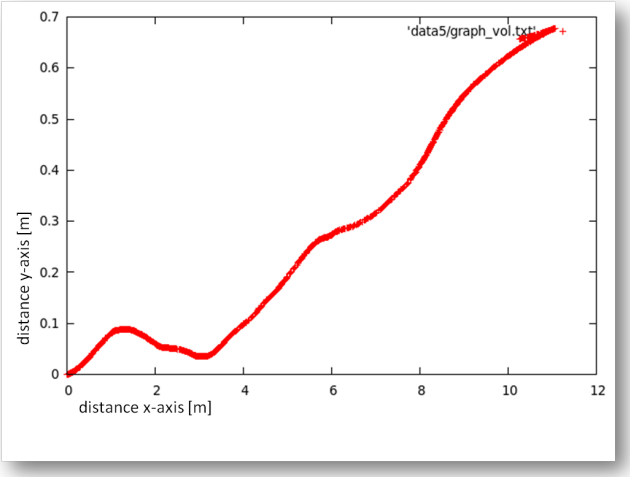


Figure 26: IMU + Voltage 1

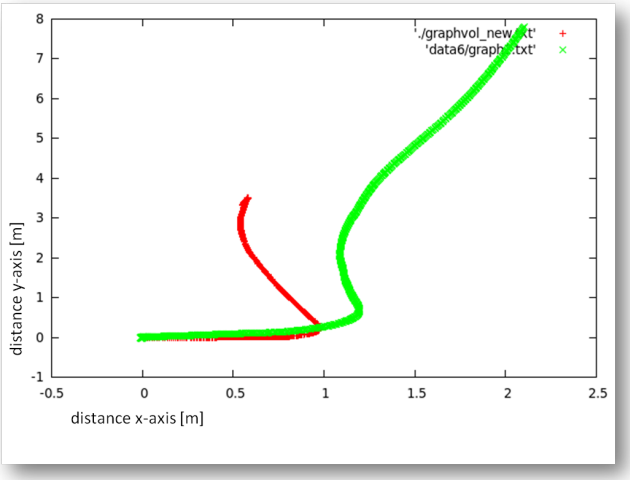


Figure 27: IMU + Voltage 2

6.7 Time and Memory Complexity

The testing was done on a Core 2 duo, 2.0 GHz laptop, running Ubuntu 10.10, with 3 GB RAM. The ROS version C-Turtle was used.

The ideal purpose, as suggested above, is to be able to view the sonar in real-time. The created node is fully able to do that. In addition, it also writes the frame id of all the sonar messages and the associated pose on to an output file. In addition it also writes its pose in a third file. Hence, the created node fulfills its requirement in terms of speed.

In terms of memory consumption, the bag file, the motion compensation node, the image viewer and the sonar viewers were all run at the same time while VLC, Google Earth and SLAM6D were running in the background. Everything ran quite well, suggesting that the node would be able to run on a computer with less memory without any problems. This

makes sense since at each cycle all values used are deleted and in total only about 20 double values are stored throughout the running of the code.

7 Discussion

As the results suggest, IMU alone is fairly good estimate for localization, and can be used for motion compensation in single beamed Sonar. It works really well when dealing with change in orientation, and when the velocity is not changing much. This is especially useful as mapping the terrain requires the UAV to be moving at a constant velocity over the interested area. However, when the UAV was accelerating and especially when the UAV was decelerating, the velocity did not change as much as it should have, leading to over estimation of velocity.
Visual Odometry is not a feasible solution for motion compensation. One of the biggest reasons for this is the fact that when the sea bed is quite far away from the UAV, the surface would not be visible. Furthermore, even in good conditions in clear water, in a swimming pool, there were numerous factors which lead to failure of visual odometry as a source of localization.
The Voltage model has, by far, the most potential for accurate localization underwater. However, the detailed model for the voltage has a lot of parameters which have to be adjusted to fix make the model realistic in real world. However, a simply model can be used to estimate the motion of the UAV. This model shows good results in straight lines, but because of the simplification of the thruster model, does not represent turning very accurately.
To solve this problem, a combination of IMU and simplified voltage model was used to utilize the advantages of both. The results were much better than the results of either of them alone, being able to work in all kinds of scenarios, such as turns, accelerations, deceleration and constant velocity movements.
The created ROS node works well by integrating seamlessly into existing code, and can publish the corrected pose of the UAV as well as write the pose on a file. It works in real-time and can be used in real-time for localization of the UAV.
Further work can be done in improving the voltage model, by adjusting the parameters to better represent the actual UAV[3].

[3]Associated presentation and code could be downloaded at http://www.mfaisalabdullah.com/2011/03/30/roboticsproject2011/ from mid-June 2011 onwards.

References

[1] "Measutronics." http://measutronics.wordpress.com/systems/, May 2011.

[2] L-3 Communications SeaBeam Instruments, East Walpole, MA 02032-1155, *Multibeam Sonar Theory of Operation*, 2011.

[3] A. C. Luca, L. Iocchi, and G. Grisetti, "Scan matching in the hough domain," in *in Proc. of Intern. Conference on Robotics and Automation (ICRA 05*, 2005.

[4] M. Fiala, "Structure from motion using sift features and the ph transform with panoramic imagery," in *Computer and Robot Vision, 2005. Proceedings. The 2nd Canadian Conference on*, pp. 506 – 513, may 2005.

[5] R. H. A, M. A. A, and Y. D. B, "Motion compensation on synthetic aperture sonar images."

[6] K. A. D., "Inertial navigation : Forty years of evolution," in *General electric company*, vol. 13, pp. 140–149, 1998.

[7] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics, Special Issue on Space Robotics*, vol. 24, p. 2007, 2007.

[8] D. Nistr, O. Naroditsky, and J. Bergen, "Visual odometry," pp. 652–659, 2004.

[9] "Dictionary.com unabridged," May 2011.

[10] "The american heritage stedman's medical dictionary," May 2011.

[11] "Collins english dictionary - complete and unabridged 10th edition," May 2011.

[12] D. K. Pathak, "Dynamic modeling, identification, and control of the jacobs auv," tech. rep., Jacobs University Bremen - Robotics Department, August 2010.

[13] A. B. H. Blow, M. Pfingsthorn, "Using robust spectral registration for scan matching of sonar range data," 2010.

[14] H. Moravec and A. E. Elfes, "High resolution maps from wide angle sonar," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pp. 116 – 121, March 1985.

[15] J. Borenstein, H. R. Everett, and L. Feng, *Navigating Mobile Robots: Systems and Techniques*. Natick, MA, USA: A. K. Peters, Ltd., 1996.

[16] W. Rencken, "Concurrent localisation and map building for mobile robots using ultrasonic sensors," in *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 3, pp. 2192 –2197 vol.3, July 1993.

[17] F. Dellaert, W. Burgard, D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, pp. 2 vol. (xxiii+637+663), 1999.