DRAFT COPY

Printed November 14, 2013 21:41

FEATURE BASED ADAPTIVE MOTION MODEL

by

Rohan Bhargava

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

at

Dalhousie University
Halifax, Nova Scotia
October 2013

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "FEATURE BASED ADAPTIVE MOTION MODEL" by Rohan Bhargava in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated: October 1, 2013

Supervisors: _____
Dr. Thomas Trappenberg

_____
Dr. Mae Sato

Readers: _____
D. Odaprof

_____
A. External

DALHOUSIE UNIVERSITY

DATE: October 1, 2013

AUTHOR: Rohan Bhargava

TITLE: FEATURE BASED ADAPTIVE MOTION MODEL

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: M.C.Sc.   CONVOCATION: January   YEAR: 2014

_____
Signature of Author

Table of Contents

Abstract

We present a method to learn and adapt the motion model. The motion model can be influenced by environmental properties and is a crucial part of the navigation system.Examples are the drift that is accounted in the motion model can be different for carpets and tiles. The AUV can have a change in their motion model when they are moving from fresh water to sea water. Our algorithm is based on the Expectation Maximization Framework which help us to learn the right parameters for the model. The Expectation Step is completed by particle filtering and smoothing. The Maximization step involves finding the parameters for the model.We use side sonar images to extract landmarks which can gives us position estimates to help us evolve our motion model.This leads to a better position estimate of the robot. We found that the our learning motion model adapted well to the change in parameters and had low localization error compared to static motion model. This algorithm eliminates the need for laborious and hand-tuning calibration process. The main significance of learning the motion model is to have better navigation algorithms and also its a step towards robots being able to adapt to environment without human intervention. We validate our approach by recovering a good motion model when the density,temperature of the water is changed in our simulations.

Acknowledgements

Thanks to all the little people who make me look tall.

Chapter 1

Introduction

1.1 Motivation

The core of human environment interaction is the ability of a person to know its position in surrounding environment. The process of estimating the robots position and orientation in the world is termed as Localization [20]. A common approach to determine the location of a robot is through the use of Global Positioning System (GPS), a segries of satellites in low earth orbit that use differential positioning to determine a location for a receiver. Another approach is to provide a prior map of the environment and with help of sensors a robot perceives the world and localizes itself in it. An additional challenge to the localization is when sensing of the external environment is impossible or incomplete due to unreliability and inaccessibility of the sensors. In this case the localization estimate can be updated using self-generated cues such as acceleration and velocity.

An example for such a scenario from underwater robotics is maintaining a pose estimate of an Autonomous Underwater Vehicle, such as Hugin 4500 (Figure 1.1). In AUV the pose estimation relies on Inertial Navigation Systems (INS) which gives an estimate of the velocity, position and orientation. All INS systems suffer from drift i.e. small errors in measurement of acceleration and angular velocity are integrated into progressively larger error. To compensate for the drift systems such as Doppler Velocity Log (DVL), surface GPS etc. are used [12] [14]. Hegrenaes et al [**?**] pointed out that there are situations where these systems fail or readings from these sensors need to be discarded due to poor quality. An example of such a situation is the non-feasibility of the vehicle to surface. In such situations they proposed to use the self-generated velocity estimates to aid INS systems.

The estimates can be difficult to acquire and maintain due to uncertainties in way a robot interacts and senses its environment. These uncertainties can arise due to noisy and incomplete sensing of the environment, uncertain movements of a robot
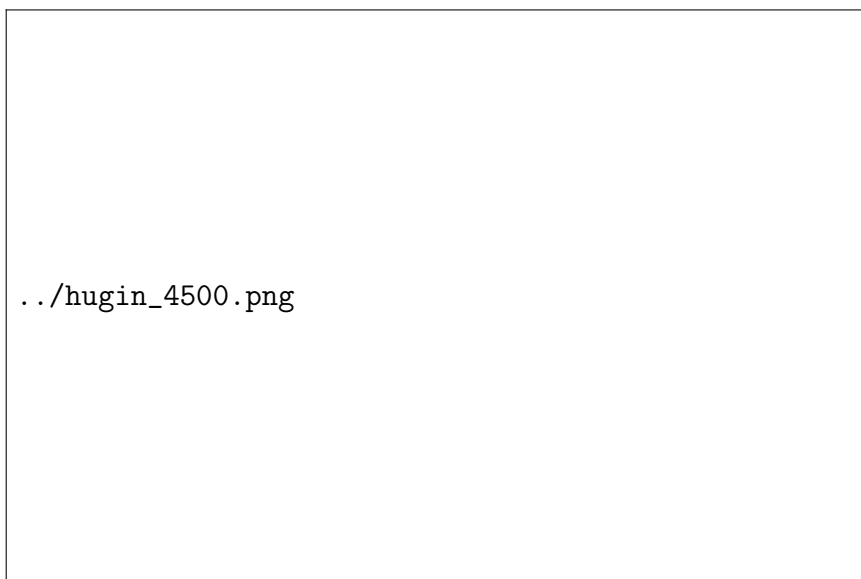
../hugin_4500.png

Figure 1.1: Hugin 4500 autonomous underwater vehicles. When submerged, the vehicle uses dead reckoning, incorporating DVL and compass input to maintain an estimate on current positioning.

in the environment and changes in the environment itself. To address these issues Sebastian Thrun [20] represented the motion and sensor models probabilistically. He defines it as instead of relying on single best guess as to what might be the case, probabilistic algorithms represent the information by probability distributions over a whole space of guesses. To update the state of a robot probabilistically there are algorithms such as Kalman Filter [11], Particle Filter [8] etc. which are based on Bayes filter.

In probabilistic robotics the motion and sensors models are represented by a distribution which is defined by its parameters. The process of determining parameters to kinematic model is termed as calibration [?] [21]. Generally the parameters to the models are hand tuned and are derived by conducting calibration experiments. Such calibration methods are impractical for two reasons. Firstly these processes are labour intensive and require prior information about the environment and robot. Secondly, changes in robot (e.g.-: general wear and tear) and environment (e.g.-: moving from fresh water to sea water). The changes in underwater environments can be due to change in density, temperature etc. of water. These changes require recalibration of

a robot while it is in operation which in most cases is not possible to do. The inaccuracies in the models will effect results for higher level tasks such as path planning [13], Simultaneous Localization and Mapping (SLAM) [20] [9] etc.

Roy and Thrun [17] proposed an online calibration method for land robots which can be performed without human intervention. They approached the calibration process as maximum likelihood estimation problem which gives an estimate of parameters for the given data. The calibration parameters are iteratively estimated by comparing pair of subsequent sensor readings. The algorithm proposed worked well for systematic drifts and the results showed the position error reduced by approximately by 83

Alizar and Parr [6] continued the work further by estimating non-systematic drifts. They proposed an algorithm to learn the right parameters of a motion model for a land robot using Expectation Maximization Framework [3]. It is an unsupervised machine learning technique that alternated between the expectation and maximization step. In the Expectation Step it creates an expectation of the log-likelihood using the current estimate of the parameters and the Maximization step which computes parameters maximizing the expected log-likelihood found in the Expectation Step. They were able to learn accurate motion models with very little user input.

Yapp [22] took Alizar and Parr [6] work further and proposed an algorithm to learn the motion and sensor models for land robots. They used the same Expectation Maximization framework to learn the right parameters for the models. To calculate the like trajectory of a robot both the algorithms implemented particle filtering [15] [2] and smoothing [4]. The algorithm started with an estimate of initial parameters and iteratively optimized the parameters based on the data collected during robots operation. The algorithm assumed that a prior map of the environment is provided.

In my thesis I specifically deal with water environments such as oceans, rivers etc. which are highly dynamic. The algorithm proposed here is different from the work done before in two ways. Firstly the algorithm is meant to learn the right parameters for an AUVs motion model. Secondly, the algorithm can learn motion model for unknown environments by generating landmarks using side sonar images and therefor doesnt have to rely on static maps. Sound Navigation and Ranging (SONAR) is a technique based on sound propagation used for detecting objects underwater. Side

scan sonar is a specific type of sonar used to image the topography of a sea floor. The SONAR sensor is the only imaging tool that can work at high depth.

My algorithm uses EM algorithm to calculate the most likely parameters for a data set. The trajectories are calculated by implementing particle filtering and smoothing. Particle Filters are chosen because they can mode non-linear transformations as well as have no restrictions in model. Particle smoothing was performed because Russel and Norving [refer] pointed out that the state of the system is better estimated by smoothing as it incorporates more information that just filtering.

The remainder of the thesis is structured as follows. Chapter 2 will explain the motion model for AUV as well as give an insight on particle filtering and smoothing. Chapter 3 will give an overview of Expectation Maximization and show how this framework is used to adapt parameters for a motion model. Chapter 4 will give explain how landmarks are extracted from side sonar images. The reliability of the landmarks are shown by extracting motion information and the algorithm to do that is described in Chapter 4. Chapter 5 consist results of a simulated experiment to show the effectiveness of the algorithm. This chapter also includes the comparison of motion estimation using side sonar images to DVL.

1.2   Contributions

The main contributions of the algorithm are-:

1) **Adaptive Motion Model**-: The motion model for AUVs adapt to changing environment. This automated process of calibration of the robots lead to no hand tuning of the models and gives us an online process which can be preformed during the robot's mission.

2) **Motion Estimation from Side Sonar Images**-: We present an approach to estimate the movement from side sonar images which can be coupled with existing motion model and can improve localization. It can be easily be performed on-board as limited amount of interest points are used which lead to lesser computation and memory usage.

Chapter 2

Background

In the chapter we start by discussing how motion models are probabilistically represented as well as give an insight about motion models for AUV. This helps us in understanding on how motion model captures the probabilistic movements of robots. We then discuss a probabilistic state estimation algorithm such as particle filter [15] [2] which is at the heart of my algorithm as well as many other robotics systems. Lastly we discuss about particle smoothing [4] [5] which gives an estimate of ground truth by calculating the distribution of past states with taking into account all the evidence up to present.

2.1   Motion Model

A motion model is responsible for capturing the relationship between the control input and the change in robot's configuration. Thrun [20] models the motion of a robot probabilistically because the same control inputs will never reproduce the same motion. A good motion model will capture the errors such as drift that are encountered during the motion of the robot. The motion model is a necessary ingredient of many algorithms such as localization,mapping etc.

Let $X = (x, y, \theta)$ be the initial pose of the robot in x-y space. Mathematically the motion model can be described as $P(X'|X, u)$, where X' is the pose after executing the motion command u. Based on the control input Thrun [20] divided the motion model in two classes 1) Odometry based motion model 2) Velocity based motion model.

The first class of motion models are used for robots equipped with wheel encoders. Odometry is generally obtained by integrating wheel encoders information and is more accurate than velocity. Velocity based models calculate the new position based on velocities and time elapsed. These models are implemented for Autonomous Underwater Vehicle(AUV) and Unmanned Aerial Vehicles(UAV). Both odometry as well as

velocity suffer from drift and slippage therefore the same control commands will not generally produce the same motion and the motion model $P(X'|X,u)$ is represented as probability distribution.

The velocity motion model proposed by Thrun [20] assumes that robot can be controlled through two velocities a rotational and translational velocity. The translational velocity at time t is denoted by v_t and rotational velocity by w_t. Hence the control input u_t can be represented by

$$u_t = (\begin{smallmatrix} v_t \\ w_t \end{smallmatrix})$$

The assumption is that positive rotational velocities w_t induce a counterclockwise rotation whereas positive translational velocities v_t correspond to forward motion. The set of equations to compute the next state of a robot for a velocity motion model are

$$x_t = x_{t-1} + V_{t-1}/W_{t-1}\sin(\theta_{t-1}) + V_{t-1}/W_{t-1}\cos(\theta_{t-1} + W_{t-1}\delta t) \qquad (2.1)$$

$$y_t = y_{t-1} + V_{t-1}/W_{t-1}\cos(\theta_{t-1}) - V_{t-1}/W_{t-1}\sin(\theta_{t-1} + W_{t-1}\delta t) \qquad (2.2)$$

$$\theta_t = \theta_{t-1} + W_{t-1}\delta t \qquad (2.3)$$

In an AUV a velocity motion model is implemented and to represent AUV's motion, 6 independent coordinates are necessary to determine the position and orientation of the rigid body. The notations used for marine vehicles are described in Table 2.1.

The pose of AUV can be represented as $s = (x, y, z, \theta, \phi, \psi)$. The first three coordinates correspond to the position along the x,y,z axes while the last three coordinates describe the orientation. Fossen [19] in his book describes the motion of a marine vehicle in 6 DOF using two coordinate systems as shown in Figure 2.1. X_0, Y_0, Z_0 represent the moving coordinate frame and is called as body-fixed reference frame. The earth-fixed reference frame is denoted by X, Y, Z. The origin of the body-reference frame is denoted by O and is chosen to coincide with the center of gravity denoted by CG.

DOF		forces and moments	linear and angular vel.	positions and Euler angles
1	motions in the x-direction (surge)	X	u	x
2	motions in the y-direction (sway)	Y	v	y
3	motions in the z-direction (heave)	Z	w	z
4	rotation about the x-axis (roll)	K	p	ϕ
5	rotation about the y-axis (pitch)	M	q	θ
6	rotation about the z-axis (heave)	N	r	ψ

Table 2.1: Notation used for marine vehicles. Table from [19]

To estimate the position of an AUV we need to calculate the velocity at which the AUV is currently moving. The velocity can be computed in two ways-: 1) Static Motion model 2) Dynamic Motion model

Hegrenaes [10] points that a way to implement a simple static motion model as table look-up based on experimental data.

$$u_r = f(n_s) \tag{2.4}$$

u_r, n_s are the water relative linear velocity in x direction and control system set point respectively. In a similar manner an expression can be established for v_r.

Another way to implement the motion model is through dynamics. The 6 Degrees of Freedom (DOF) rigid body equations of motion described by Fossen [19] are

$$X = m[\dot{u} - vr + wq - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(pr + \dot{q})] \tag{2.5}$$

$$Y = m[\dot{v} - wp + ur - y_G(r^2 + p^2) + z_G(qr - \dot{p}) + x_G(qp + \dot{r})] \tag{2.6}$$
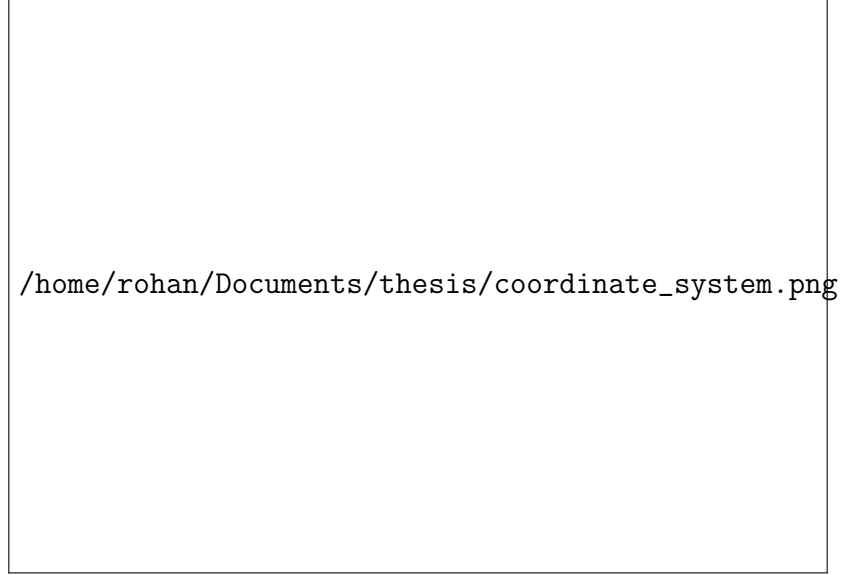
Figure 2.1: Body-fixed and earth-fixed reference frames. Figure from [19]

$$Z = m[\dot{w} - uq + vp - z_G(p^2 + q^2) + x_G(rp - \dot{q}) + y_G(rq + \dot{p})] \qquad (2.7)$$

$$K = I_x\dot{p} + (I_z - I_y)qr - (\dot{r} + pq)I_{xz} + (r^2 - q^2)I_{yz} + (pr - \dot{q})I_{xy} + m[y_G(\dot{w} - uq + vp) - Z_g(\dot{v} - wp + ur)] \qquad (2.8)$$

$$M = I_y\dot{q} + (I_x - I_z)rp - (\dot{p} + qr)I_{xy} + (p^2 - r^2)I_{zx} + (qp - \dot{r})I_{yz} + m[y_G(\dot{u} - vr + wq) - z_g(\dot{w} - uq + vp)] \qquad (2.9)$$

$$N = I_z\dot{r} + (I_y - I_x)pq - (\dot{q} + rp)I_{yz} + (q^2 - p^2)I_{xy} + (rq - \dot{p})I_{zx} + m[x_G(\dot{v} - wp + ur) - y_g(\dot{u} - vr + wq)] \qquad (2.10)$$

The equations described above can be expressed in a more compact form:

$$M_{RB}\dot{\mathcal{V}} + C_{RB}(\mathcal{V})\mathcal{V} = \tau_{RB} \qquad (2.11)$$

Here $\mathcal{V} = [u, v, w, p, q, r]^T$ is the body fixed linear and angular velocity and $\tau_{RB} = [X, Y, Z, K, M, N]$ is generalized vector of external forces and moments. M_{RB} is the rigid body inertia matrix and C_{RB} is Coriolis and centripetal matrix.

The right hand side of the vector 2.11 represents the external forces and moments acting on the vehicle. Fossen [19] classifies the forces into 1) Radiation-induced forces 2) Environmental Forces 3) Propulsion Forces

τ_{RB} can be represented as the sum of these forces.

$$\tau_{RB} = \tau_H + \tau_E + \tau \tag{2.12}$$

Here τ_H is the radiation induced forces and moments, τ_E is used to describe the environmental forces and moments and τ is the propulsion forces and moments. Equations 2.11 and equation 2.12 can be combined to yield the following representation of 6 DOF dynamic equations of motion:

$$M\mathcal{V}^{\cdot} + C(\mathcal{V})\mathcal{V} + D(\mathcal{V})\mathcal{V} + g(\eta) = \tau_E + \tau \tag{2.13}$$

where

$M \triangleq M_{RB} + M_A \; ; \; C(\mathcal{V}) \triangleq C_{RB}(\mathcal{V} + C_A(\mathcal{V})$

M_A is the added inertia matrix $C_A(\mathcal{V})$ is the matrix of hydrodynamic Coriolis and centripetal terms. $g(\eta)$ is the restoring force.

Lammas [12] pointed out that navigation equation of an underwater vehicle is-:

$$\mathcal{V}^{\cdot} = M^{-1}(\tau - C(\mathcal{V})\mathcal{V} - D(\mathcal{V})\mathcal{V} - g(\eta)) \tag{2.14}$$
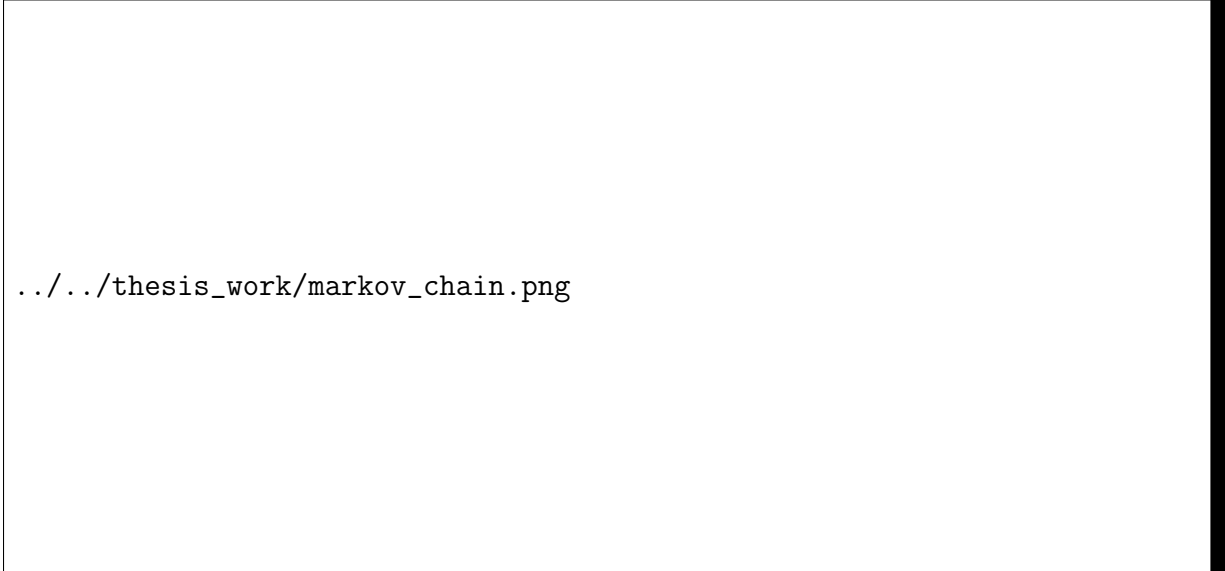
$\mathcal{V}^{\cdot}$ can be integrated with time to get velocity.

In the static model the velocity is calculated from a lookup table. In the other model we are computing forces and moments on the fly but the parameters to these forces are considered to be static. The parameters such as density, temperature etc of water can change with time and lead to an inaccurate estimate of velcoity in both the models. Hence the velocity needs to be adapted and Chapter **??** explains how it is done in my algorithm.

2.2 Particle Filter

Particle Filter is a state estimation algorithm based on a sampling method for approximating a distribution. Thrun [20] defines particle fitler as an alternative non-parametric implementation of the Bayes filter. It also can be called as a Sequential

../../thesis_work/markov_chain.png

Figure 2.2: A temporal Bayesian model with hidden states x_t, observations z_t and controls u_t. Figure taken from [20]

Monte Carlo(SMC) algorithm. The first attempt to use SMC was seen in simulations of growing polymers by M.N Rosenbluth and A.W. Rosenbluth [16]. Gordon et al. [8] provided the first true implementation of sequential monte carlo algorihtm.

Thrun [20] stated that the key idea behind particle filter is to represent the posterior bel(x_t) by a set of random state samples drawn from this posterior. Instead of representing the distribution by a parametric form particle filter represents a distribution by a set of samples drawn from this distribution. In Figure 2.3 the belief if represented by a set of particles. The representation is a approximation but it is nonparametric and therefore there are advantages of using particle filters as an alternative to Extended Kalman Filter and Unscented Kalman Filter. Particle Filters can represent a broader space of distributions for example non-Gaussian and can model non linear transformations of random variables. In Figure 2.3 particle filters are shown to model non-linear transformations.

The objective of particle filters is to estimate the state of the system given the observation variables. They are designed for Hidden Markov Models(Fig 2.2, where the system consists of hidden and observed variables. In this model the state x_t is the hidden random variable as it is not directly observed. The state at time t is only dependent upon the state at time $t-1$ and external influences such as control u_t. The

/home/rohan/Documents/thesis_work/particle_filter_non_gaussian_thru
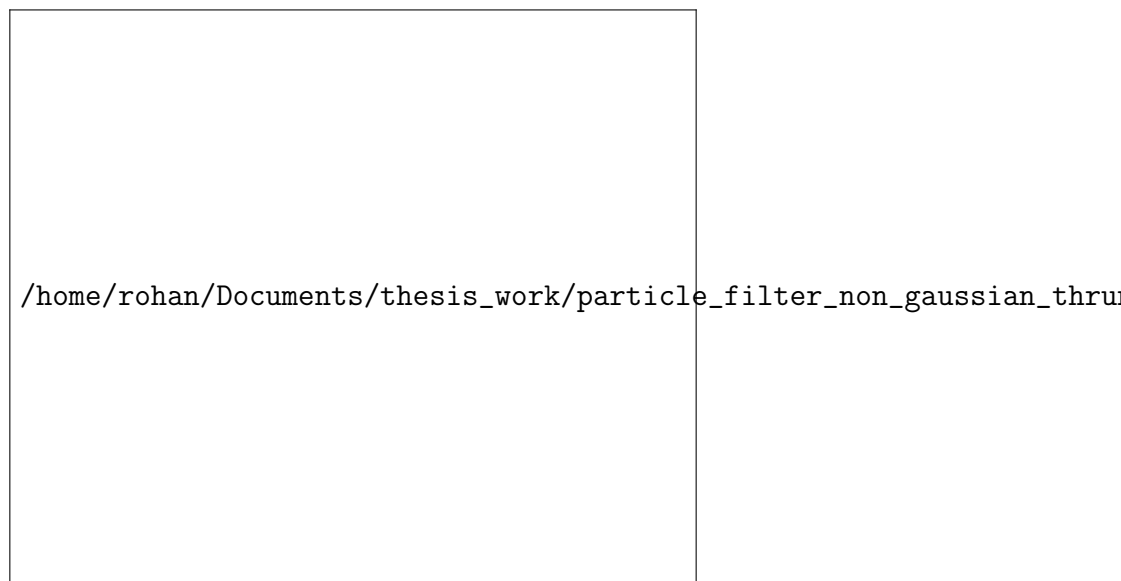
Figure 2.3: The "particle" representation used by particle filters. The lower upper right graphs shows samples drawn from a Gaussian random variable, X. These samples are passed through the nonlinear function shown in the upper right graph. The resulting samples are distributed according to the random variable Y. Figure taken from [20]

measurement z_t depends on the state at time t. The knowledge about the influence of the control on the system can be used to calculate a new expected location and the measurement can be combined in a Bayesian way.

The algorithm for particle filters is described below-:

Input: X_{t-1}: particle set

u_t: most recent control

z_t: most recent measurement

Output: X_t:particle set

begin

> **for** *m=1 to M do* **do**
> | sample x_t^m $p(x_t|u_t, x_{t-1}^m)$ $w_t^m = p(z_t|x_t^m)$ $X_t^- = X_t^- + (x_t^m, w_t^m)$
> **end**
>
> **for** *m=1 to M do* **do**
> | draw i with probability $\propto w_t^{[i]}$
> | add $x_t^{[i]}$ to X_t
> **end**
>
> return X_t

end

Algorithm 1: Particle Filter Algorithm. Algorithm taken from [20]

In algorithm 1 each particle x_t^m is instantiation of the state at time t. The first step is to generate a hypothetical state x_t^m for time t based on previous state x_{t-1}^m and control u_t. The particles are samples from the state transition distribution $p(x_t|u_t, x_{t-1})$. The importance factor for each particle x_t^m is calculated and denoted by w_t^m. Importance factor is defined as the probability of measurement z_t under the particle x_t^m. Thus importance factor are used to incorporate the measurements into the particle set. In practice, the number of particles used are a large number(e.g.-:1000).

The key part of the algorithm is the re-sampling step in particle filter algorithm. The algorithm draws M particles with replacement from a temporary particle set X_t^-. The probability of drawing the particles is given by the importance factor. The re-sampling step is a probabilistic implementation of the Darwinian idea of survival of the fittest. It refocuses the particle set to regions in state space with high posterior probability.

Particle Filters is an integral part of my algorithm to learn the right parameters of the motion model and the way it is used is explained in 3.3.2.
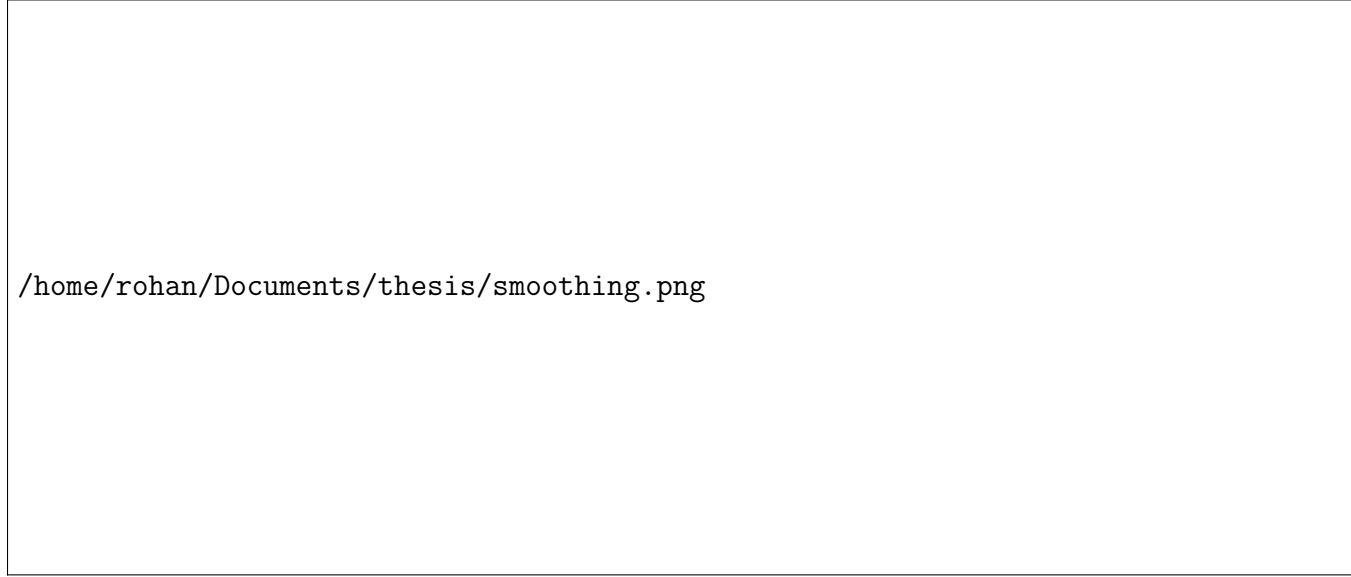
Figure 2.4: Smoothing computes $P(X_k|e_{1:t})$, the posterior distribution of the state at some past time k given a complete sequence of observations from 1 to t. Figure taken from [18]

.

2.3   Particle smoothing

The particle filter algorithm as described before is the first step in the Expectation process. The next algorithm that completes the Expectation Step is the particle smoothing. Doucet [5] in his paper stated that filtering based on observations received up to the current time is used to estimate the distribution of the current state of an Hidden Markov Model (HMM) whereas smoothing is used to estimate distribution of state at a particular time given all the observations up to some later time. Russel and Norving [18] showed that the state of the system is better estimated by smoothing as it incorporates more information than just filtering. We use particle smoothing algorithm proposed by Teddy N Yap and Christian R. Shelton [22] which was based on the technique presented by Docuet et al. [4] and Godsill et al. [7].

Particle smoothing is carried out in order to generate samples from the entire joint smoothing density $p(x_{0:T}|u_{1:T}, z_{1:T})$. The equations described by Yapp [22] are

$$p(x_{0:T}|u_{1:T}, z_{1:T} = \prod_{t=0}^{T} p(x_t|x_{t+1:T}, u_{1:T}, z_{1:T}) \qquad (2.15)$$

where,

$$p(x_t|x_{t+1:T}, u_{1:T}, z_{1:T}) = p(x_t|x_{t+1}, u_{1:t+1}, z_{1:t}) \tag{2.16}$$

$$= \frac{p(x_{t+1}|x_t, u_{1:t+1}, z_{1:t})p(x_t|u_{1:t+1}, z_{1:t})}{p(x_{t+1}|u_{1:t+1}, z_{1:t})} \tag{2.17}$$

$$= \frac{p(x_{t+1}|x_t, u_{t+1})p(x_t|u_{1:t}, z_{1:t})}{p(x_{t+1}|u_{1:t+1}, z_{1:t})} \tag{2.18}$$

$$\alpha \quad p(x_{t+1}|x_t, u_{t+1})p(x_t|u_{1:t}, z_{1:t}) \tag{2.19}$$

This technique assumes that particle filtering has been carried out on the data set which results in a set of particles X_t with their corresponding weights. The states can be generated recursively backwards in time given the future states using equation 2.19. $p(x_t|u_{1:t}, z_{1:t})$ can be obtained by performing particle filtering. $p(x_t|x_{t+1:T}, u_{1:T}, z_{1:T})$ can be approximated by particles with modified importance weights $w_{t|t+1}^{[i]}\alpha w_t^{[i]}p(x_{t+1}|x_t^{[i]}, u_{t+1})$ given filtered particles $x_t^{[i]}$.

Algorithm 2 shows the step involved to sample from the entire joint smoothing density.

Input: $X_t, t = 0, 1, ..., T$: particle approximations to the posterior pdfs

$\qquad p(x_t|c_{1:t}, s_{1:t}), t = 0, 1, .., T$

$c_{1:T} = (c_1, c_2, ..., c_T)$: set of controls from time 1 to time T

Output: $x'_{0:T} = (x'_0, x'_1, ..., x'_T)$: a sample from the entire joint smoothing

$\qquad$ density $p(x_{0:T}|c_{1:T}, s_{1:T})$

begin

$\quad$ draw i with probability $\propto w_T^{[i]}$ $x'_T \leftarrow x_T^{[i]}$

$\quad$ **for** $t \leftarrow T - 1$ *down to 0 do* **do**

$\qquad$ **for** $i \leftarrow 1 to N_s$ *do* **do**

$\qquad \quad \mid$ $w_{t|t+1}^{[i]} \leftarrow w_t^{[i]}p(x'_{t+1}|x_t^{[i]}, u_{t+1})$

$\qquad$ **end**

$\qquad$ draw i with probability $\propto w_{t|t+1}^{[i]}$

$\qquad$ $x' \leftarrow x_t^{[i]}$

$\quad$ **end**

end

Algorithm 2: Sample the entire joint smoothing density $p(x_{0:T}|c_{1:T}, s_{1:T})$

In the first step of the algorithm a particle is drawn with probability proportional to the filtered weight of the particles. The next step is to move a time step back and

modify the weights of the particles by calculating the new smoothed weights. The new smoothed weights are the product of state transition probability $p(x'_{t+1}|x_t^{[i]}, u_{t+1})$ and the weight of the particle $w_t^{[i]}$. The next step in the algorithm is to draw particles with probability proportional to new smoothed weights $w_{t|t+1}^{[i]}$. The sequence of particles x' drawn from joint smoothing density $p(x_{0:T}|c_{1:T}, s_{1:T})$ from time 0 to time T form a sampled trajectory $x'_{0:T} \triangleq (x'_0, x'_1, ..., x'_T)$.

Chapter 3

Learning the motion model

3.1 Introduction

3.1.1 Work done so far on learning motion model

3.1.2 General Architecture

Figure 3.1: Block Diagram of the framework
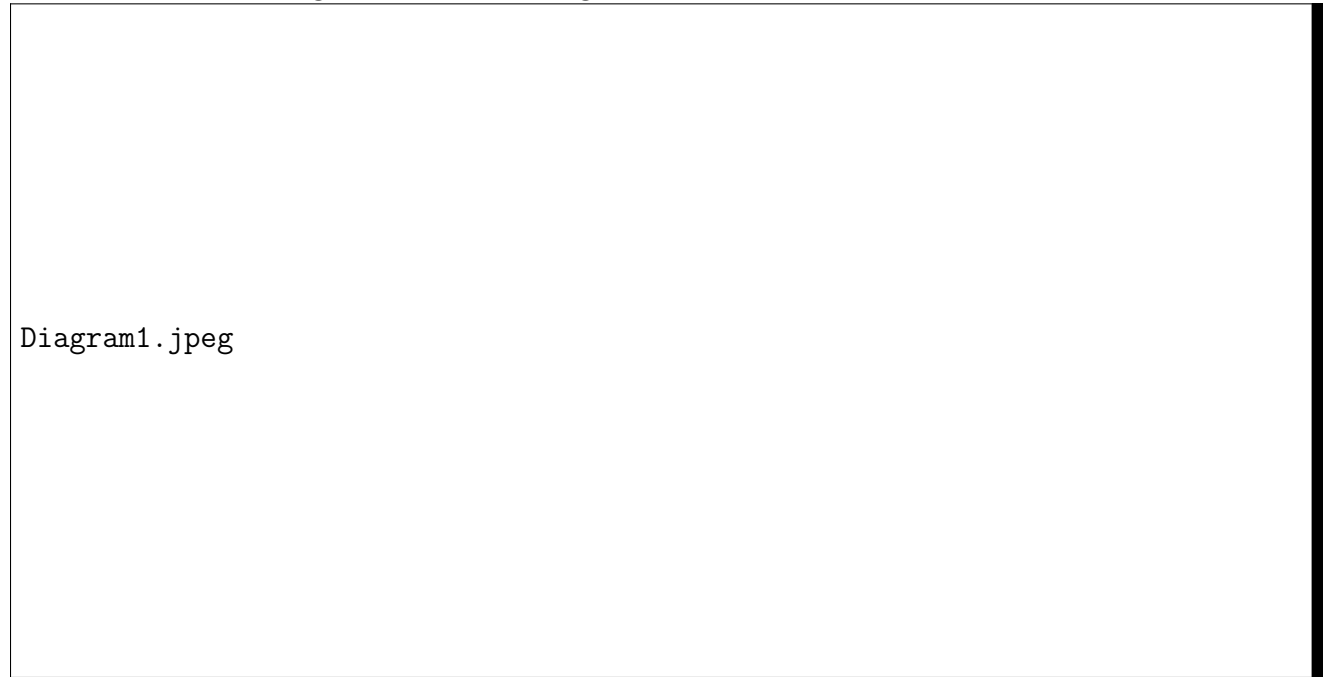
```
Diagram1.jpeg
```

Figure 3.1.2 is a block diagram describing the whole system. The first step is to initialize the motion model with a set of parameters.

Given the prior distribution $p(x_{t-1})$ of the robot's pose at time t-1,the motion model $p(x_t|x_{t-1}, u_t)$,the sensor model $p(z_t|x_t)$,the control commands $u_{1:t}$ and the sensor measurements $z_{1:T}$ we perform particle filtering using Algorithm 1 from time t=1 to time t=T. After we perform filtering we have a set of particles with importance

factors describing the distribution over the pose of the robot. To complete the Expectation Step particle smoothing is performed recursively backwards in time. A set of trajectories are generated by repeating Algorithm 2.

Based on the trajectories and the reported translational and rotational motion we calculate errors at each time step. We perform newton conjugate gradient on the errors to give us the best estimate of the parameters. This completes the Maximization Step.

The learned parameters are re-assigned to the motion model which helps in adapting the model. The whole process is repeated at each time step so that we can dynamically learn the right parameters. Various algorithm such as SLAM,Kalman Filter are dependent upon the accuracy of the motion model and adapting our motion models is the most basic step towards it. We found that the adaptive motion model performs better than the static and the results are shown in the later chapters.

3.2 Particle Filtering and Smoothing

3.2.1 Particle Filtering

3.2.2 Particle Smoothing

3.3 Adapting Motion Model

3.3.1 Expectation Maximization

3.3.2 Parameter Estimation

Velocity as distribution As stated in earlier chapters as our environment changes the estimate for a control input becomes erroneous. The control input to a AUV's motion model is velocity and to adapt our motion motion we assume it as a Gaussian distribution. This chapter describes an algorithm to adapt the parameters for a distribution representing velocity. To adapt parameters of a distribution we use a Expectation Maximization [3] which is an unsupervised machine learning technique.

Expectation Maximization is an iterative process of finding maximum likelihood of parameters of a model which depend upon hidden variables. We use the EM algorithm as described by Christopher M. Bishop in his book [1]. We have a joint distribution $p(X, Z|\theta)$ where X are the observed variables and Z are the latent variables governed

by parameters θ. The goal is to maximize the likelihood function $p(X|\theta)$ with respect to θ. The steps taken in EM algorithm are described below -:

    1. Have an initial estimate of the parameters θ^{old}

2. **E Step:** Evaluate $p(Z|X, \theta^{old})$

3. **M Step:** Evaluate $\theta^{new} = argmax_\theta L(\theta, \theta^{old})$

        where $L(\theta, \theta^{old}) = \Sigma_Z p(Z|X, \theta_{old}) \log p(X, Z|\theta)$

4. Check for the convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied then let

        $\theta \leftarrow \theta^{new}$

and return to step 2

There are various convergence techniques that can be applied in step 4. In our algorithm we use Newton Conjugate Gradient to estimate the right parameters.

    The first step of parameter estimation i.e. Expectation Step consists of particle filtering and smoothing as described in Chapters and respectively. The output of this step is a set of robot trajectories which are treated as ground truth. At each time step we calculate the error between the distance given by trajectory and the actual distance moved.

$$\epsilon_{T_t}^{[j]} = (\theta_{t+1}^{'[j]} - \theta_t^{'[j]} - r_t^{''}) mod 2\pi$$

$$\epsilon_{D_t}^{[j]} = (x_{t+1}^{'[j]} - x_t^{'[j]}) cos(\theta_t^{'[j]} + \frac{r_t^{''} + \epsilon_{T_t}^{[j]}}{2}) + (y_{t+1}^{'[j]} - y_t^{'[j]}) sin(\theta_t^{'[j]} + \frac{r_t^{''} + \epsilon_{T_t}^{[j]}}{2}) - d_t^{''}$$

$\epsilon_{T_t}^{[j]}$ and $\epsilon_{D_t}^{[j]}$ are rotational and translational errors for t=0,1...T-1 for j^{th} sampled trajectory. $\epsilon_{W_t}^{[j]}$ and $\epsilon_{V_t}^{[j]}$ are the rotational and translation errors in velocity and is given by $\epsilon_{T_t}^{[j]}/\delta(t)$ and $\epsilon_{D_t}^{[j]}/\delta(t)$. $r_t^{''}$ and $d_t^{''}$ are the distance moved and are given by $r_t^{''} = w_t * \delta(t)$ and $d_t^{''} = v_t * \delta(t)$. w_t and v_t are the reported velocity. Based on the motion model described

$$\epsilon_{V_t}^{[j]} \sim \mathcal{N}(0, v_t^2 \sigma_{V_v}^2 + w_t^2 \sigma_{V_w}^2 + \sigma_{V_1}^2)$$

$$\epsilon_{W_t}^{[j]} \sim \mathcal{N}(0, v_t^2 \sigma_{w_v}^2 + w_t^2 \sigma_{V_w}^2 + \sigma_{W_1}^2)$$

The log likelihood functions are

$$\mathcal{L}(\sigma_{V_v}^2, \sigma_{V_w}^2, \sigma_{V_1}^2) = -\frac{1}{2} \sum_j \sum_{t=0}^{T-1} [\log 2\pi + \log(v_t^2 \sigma_{V_v}^2 + w_t^2 \sigma_{V_w}^2 + \sigma_{V_1}^2) + \frac{(\epsilon_{V_t}^{[j]})^2}{v_t^2 \sigma_{V_v}^2 + w_t^2 \sigma_{V_r}^2 + \sigma_{V_1}^2}$$

$$\mathcal{L}(\sigma_{W_v}^2, \sigma_{W_w}^2, \sigma_{W_1}^2) = -\frac{1}{2} \sum_j \sum_{t=0}^{T-1} [\log 2\pi + \log(v_t^2 \sigma_{W_v}^2 + w_t^2 \sigma_{W_w}^2 + \sigma_{W_1}^2) + \frac{(\epsilon_{W_t}^{[j]})^2}{v_t^{''2} \sigma_{W_v}^2 + w_t^{''2} \sigma_{W_w}^2 + \sigma_{W_1}^2}$$

In the Maximization Step we minimize the likelihood function. In this case we maximize the log likelihood function because of the minus sign with respect to the motion parameters. We use newton conjugate gradient ascent for finding the local maximum of the function. The gradient of the function is the taken as the first search direction while the next search direction are chosen in such a way that they are orthogonal to all previous search directions.

The output of the step described above is a set of parameters that maximize the function. We substitute these values in our motion model at every time step and the process is repeated throughout the robot's mission.

Chapter 4

# Landmarks extraction using Side Sonar Images

4.1 Introduction

4.1.1 Side Scan Sonar

4.1.2 Speeded Up Robust Features

4.2 Dynamic Landmarks

In the particle filtering algorithm the sensor model is responsible to assign weights to the particles.The sensor model describes the process by which sensor measurements are generated in the physical world.It is generally done by using some sort of references in the world aka landmarks. Austin and Elizar in their papers used static maps of known environments to generate references for their sensor model. The probability of having static maps for underwater environments is pretty low and thus lead us to use side sonar images as references for the sensor model. As you can see in the image there

Figure 4.1: Side Sonar Image

/home/rohan/Documents/thesis_work/visual_sonar/sonar_59/18.jpg
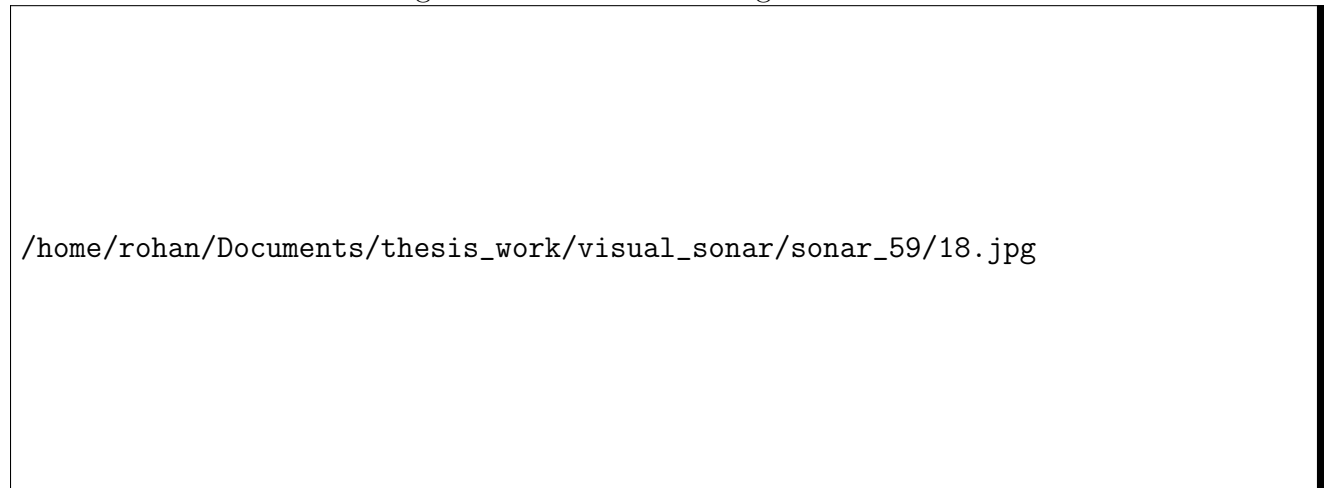
are lot of horizontal lines which we treat as noise. For us to use the side sonar images we performed a pre-processing step in which we used a median filter on the image.

This helped us in getting rid of major noises in terms of horizontal lines but also blurred the image. In order to generate some landmarks we ran feature extractions techniques like SURF. This algorithm helped us in detecting interest points in the image and are shown below in circles. Andrew Vardyrefer the paper) ran multiple image registration techniques such as phase matching (you need to mention more) on side sonar images and showed that SURF performs better than the rest .

Figure 4.2: SURF Keypoints

```
/home/rohan/Documents/thesis_work/visual_sonar/landmarks.png
```

These interest points can be treated as landmarks. We use a high Hessian threshold so that we have a maximum of 4 landmarks. The distance of the robot and the particles to these landmarks are used to assign weights to the particles.

The output of a side sonar is a ping of the surface and for an image to be created we need to combine pings over time. Andrew Vardy in his paper explains on how to combine pings to create an meaningful image. For our algorithm to run on the AUV we can use Andrew's algorithm as a black box and run our feature extraction on the output i.e. images of the seabed. This method allows us independence from a static map as well as helps in learning the motion model on the fly in a new environment.

4.3   Motion Estimation using side sonar images

In AUV the motion estimation is generally done through dead-reckoning. It is a process of calculating the current position based upon the previous position and the speed of vessel. The velocity of the AUV can be estimated by the acceleration measurement supplied by an Inertial Measurement Unit. Another way is to use a Doppler Velocity Log(DVL) that measures velocity in water by measuring the Doppler

effect on scattered sound waves. Dead reckoning may have significant errors as the velocity and direction must be accurately known at every time step.

In this algorithm we propose to get motion estimates using side sonar images. As stated before we run SURF on the images and generate some key points. These key points are matched in the next image using a KNN based matcher. The matched key points gives us an estimate on the movement of the vehicle. In the figure we show two consecutive images and in the first image we have the key points marked in circle. In the next image we have the matched key points marked in green circles. This

Figure 4.3: SURF Keypoints

/home/rohan/Documents/thesis_work/visual_sonar/surf_working.jpeg

estimate can be coupled with the previous estimate to calculate the current position. The visual input to the dead reckoning algorithm has its pros and cons.The main advantage of using a visual estimate is that it doesn't suffer from drift which is prime concern for underwater vehicles. The disadvantage lies in the fact that we don't have side sonar images available every time. We can solve the problem by combining the visual input with the velocity estimates. We can pass the visual motion estimate and

the DVL estimate to a Kalman Filter and use the output as an input to our dead-reckoning estimate. The second disadvantage is the computation power available on AUV. To specifically deal with the problems we use a high Hessian threshold to extract maximum of 4 landmarks so that feature matching is not computationally expensive.

We validate our algorithm on datasets consisting of side sonar images and the total distance the AUV moves.

Chapter 5

Results

5.1 Adapting Motion Model

We use a simulation to demonstrate the effectiveness of learning the motion model. The simulation mainly consists of particle filter SLAM. We use a motion model described in the above chapters. The sensor model basically measures the distance from the four static landmarks defined at the start of the experiment. The experiment runs over 100 time steps and at every 5th time step we change the parameters of the motion model. As described in the above chapters the parameters that we intended to learn are $\sigma^2_{D_d}, \sigma^2_{T_d}, \sigma^2_{D_1}, \sigma^2_{D_r}, \sigma^2_{T_r}, \sigma^2_{D_1}, \sigma^2_{T_1}$. In the first stage of the experiment at every 5th time step we change $\sigma^2_{D_d}$ or $\sigma^2_{T_r}$ in our motion model. The following table and plots describes the five experiments that were conducted in the simulation.

No.	σ_{D_d}	σ_{T_r}	$\sigma^*_{D_d}$	$\sigma^*_{T_r}$	Sensor Noise
1	0.05	0.05	0.2	0.05	2.0
2	0.05	0.05	0.2	0.05	5.0
4	0.05	0.05	0.5	0.05	2.0
5	0.05	0.05	0.5	0.05	5.0
6	0.05	0.05	0.05	0.2	5.0
7	0.05	0.05	0.05	0.5	5.0

$\sigma_{D_d}, \sigma_{T_r}$ are the parameters values that the motion model was initialized. These values are altered in order to simulate a change in the motion model and they are described by $\sigma^*_{D_d}, \sigma^*_{T_r}$. The sensor noise can be described as the confidence the robot has in its sensor model . The impact of the noise on the localization error can be seen in the following plots.

Figure 1 and Figure 2 are plots of the localization error with different sensor noises. We can see in both the cases the learned motion model performed better than the static motion model. Another important point is that the average error is less
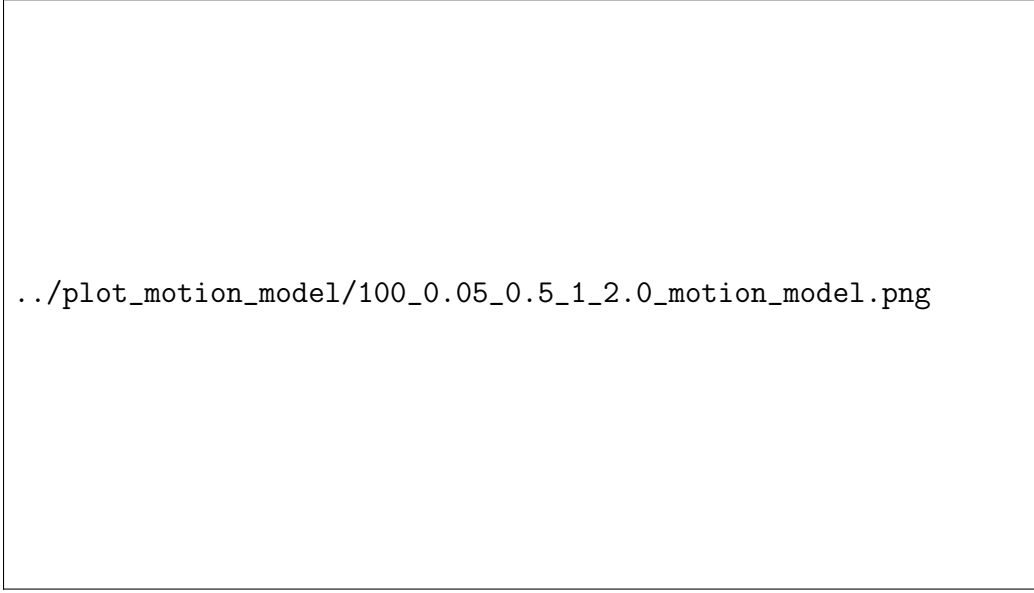
Figure 5.1: σ_{D_d}=0.05 $\sigma^*_{D_d} = 0.2$ Sensor Noise= 2.0

../plot_motion_model/100_0.05_0.2_1_2.0_motion_model_1.png

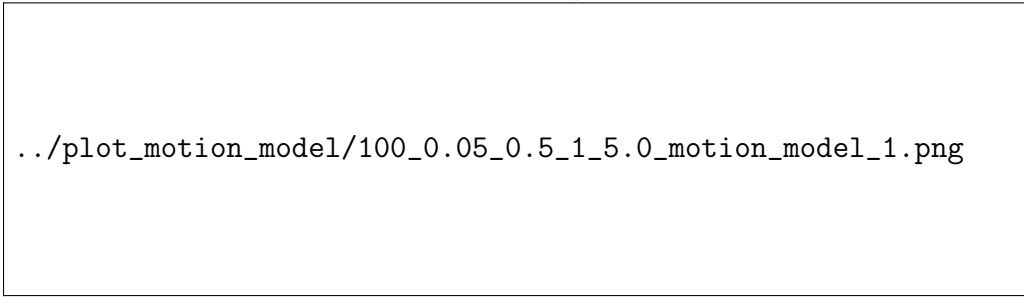Figure 5.2: σ_{D_d}=0.05 $\sigma^*_{D_d} = 0.2$ Sensor Noise= 5.0

../plot_motion_model/100_0.05_0.2_1_5.0_motion_model_1.png

Figure 5.3: $\sigma_{D_d}=0.05$ $\sigma_{D_d}^* = 0.5$ Sensor Noise= 2.0
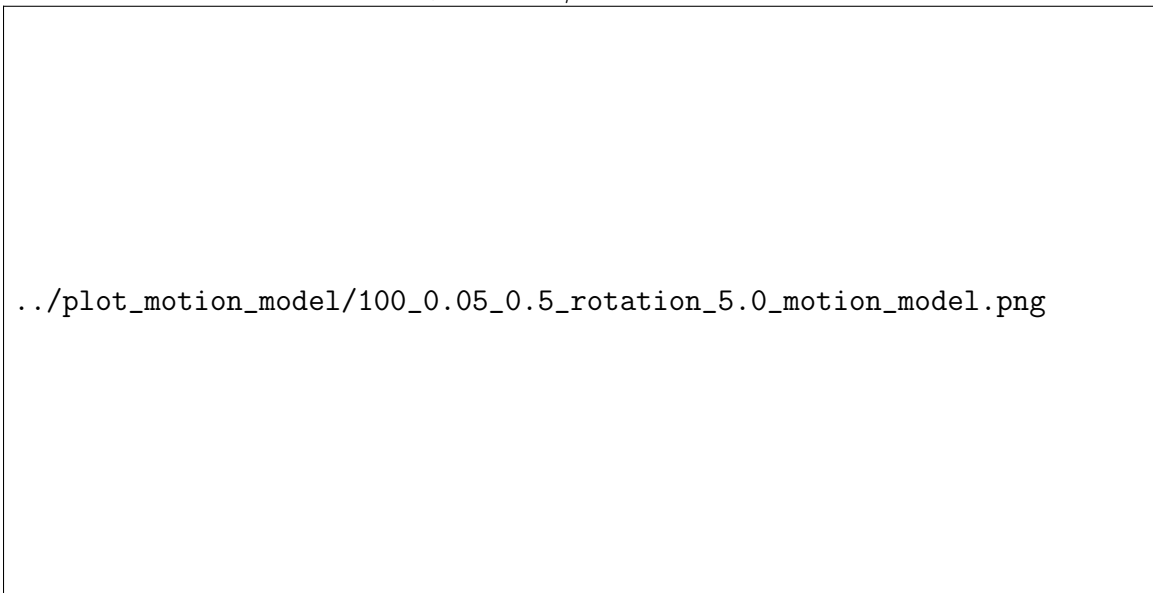
../plot_motion_model/100_0.05_0.5_1_2.0_motion_model.png

Figure 5.4: $\sigma_{D_d}=0.05$ $\sigma_{D_d}^* = 0.5$ Sensor Noise= 5.0

../plot_motion_model/100_0.05_0.5_1_5.0_motion_model_1.png

when the sensor noise is 2.0 as compared to the second case. This can be accounted for the fact that our localization algorithm is more confident on the sensor model as compared to the motion model.

As we can see in Figure 3 and Figure 4 at 5$^{\text{th}}$ time step the error shooting up but the learned motion model brings back the error whereas the static motion model takes time to recover back depending upon the sensor noise. In both the figures we can see that the error is pretty static in the learned motion model whereas in the static motion model there is a lot of fluctulation.

Figure 5 and Figure 6 describe the errors when the robot rotational motion is much more than the translational motion. We can clearly see that the learned motion model quickly adapts to the changes whereas the static motion model struggles to get the

Figure 5.5: σ_{T_r}=0.05 $\sigma_{T_r}^* = 0.2$ Sensor Noise= 5.0

../plot_motion_model/100_0.05_0.2_rotation_5.0_motion_model.png
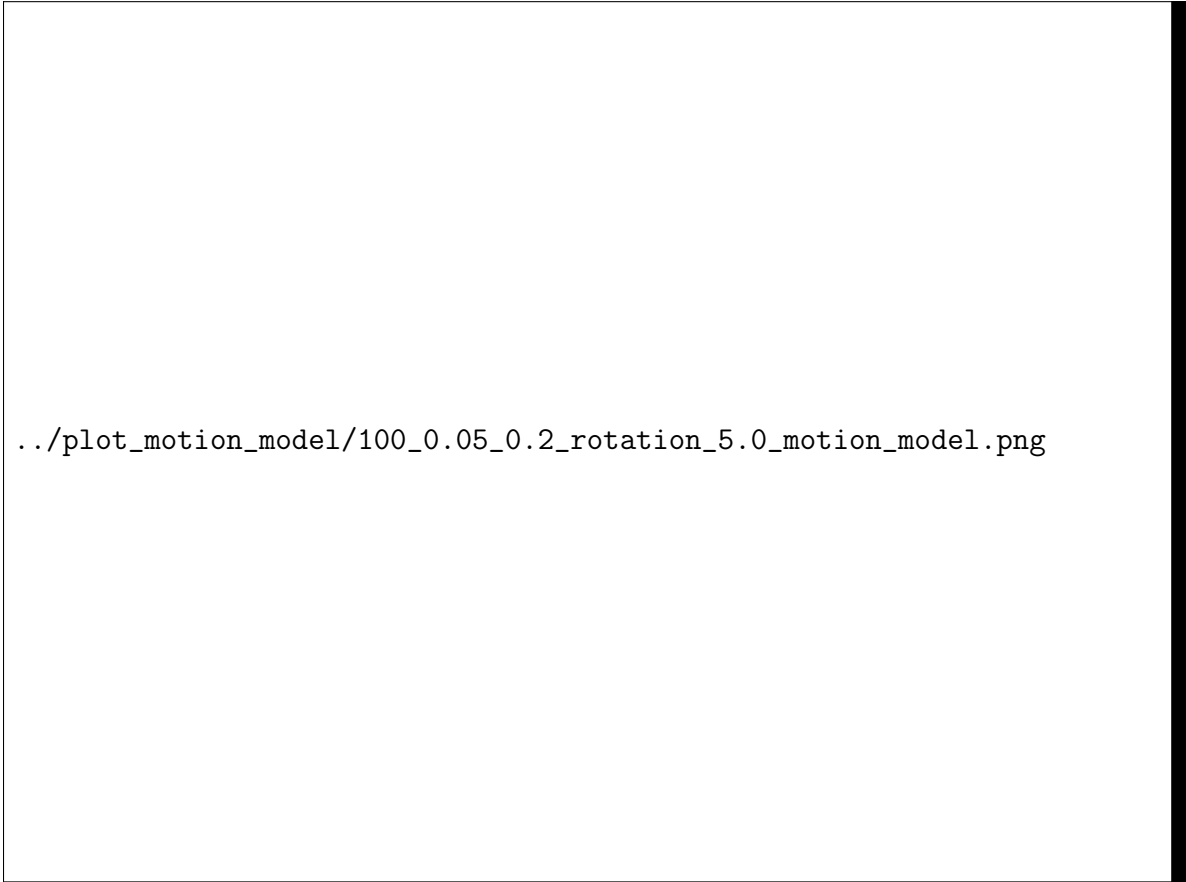
Figure 5.6: σ_{T_r}=0.05 $\sigma_{T_r}^* = 0.5$ Sensor Noise= 5.0

../plot_motion_model/100_0.05_0.5_rotation_5.0_motion_model.png

error down.

In all the cases it was very clear that we could see the adaptive motion model performing better than the static. The sensor noise had its impact on the overall error. Robot calibration is important to process in mobile robotics. The proposed algorithm is an automated process which can help us in better navigation of the robots and can be used for any motion model.

5.2 Motion estimation using side sonar images

Chapter 6

Conclusion

Did it!

Bibliography

[1] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[2] Zhe Chen. Bayesian filtering: From Kalman filters to particle filters, and beyond. *Statistics*, 182(1):1–69, 2003.

[3] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[4] Arnaud Doucet, Simon J Godsill, and Mike West. Monte Carlo filtering and smoothing with application to time-varying spectral estimation. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II701—-II704. IEEE, 2000.

[5] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

[6] Austin I Eliazar, Parr Cs, and Duke Edu. Learning Probabilistic Motion Models for Mobile Robots. 2004.

[7] Simon J Godsill, Arnaud Doucet, and Mike West. Monte Carlo Smoothing for Nonlinear Time Series. *Journal of the American Statistical Association*, 99(465):156–168, March 2004.

[8] Neil J Gordon, David J Salmond, and Adrian F M Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

[9] G Grisettiyz, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437. IEEE, 2005.

[10] Oddvar Hallingstad and Kongsberg Maritime. Comparison of Mathematical Models for the HUGIN 4500 AUV Based on Experimental Data Commonsfor dervabiov menfotionetepenotations hed so-led hdodmnamice depries . ntheionseis are oces andimomets . os the. (7491):17–20, 2007.

[11] Rudolph Emil Kalman and Others. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[12] Andrew Lammas, Karl Sammut, and Fangpo He. 6-DoF Navigation Systems for Autonomous Underwater Vehicles. 2004.

[13] S M LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, U.K., 2006.

[14] John J Leonard, Andrew A Bennett, Christopher M Smith, and H Feder. Autonomous underwater vehicle navigation. In *IEEE ICRA Workshop on Navigation of Outdoor Autonomous Vehicles*, 1998.

[15] Branko Ristic, Sanjeev Arulampalm, and Neil James Gordon. *Beyond the Kalman filter: Particle filters for tracking applications.* Artech House Publishers, 2004.

[16] Marshall N Rosenbluth and Arianna W Rosenbluth. Monte Carlo calculation of the average extension of molecular chains. *The Journal of Chemical Physics*, 23:356, 1955.

[17] N. Roy and S. Thrun. Online self-calibration for mobile robots. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 3:2292–2297.

[18] Stuart Russell. *Artificial intelligence: A modern approach, 2/E.* Pearson Education India, 2003.

[19] Fossen Thor. Guidance and Control Of Ocean Vechiles.

[20] Sebastian Thrun, Wolfram Burgard, Dieter Fox, and Others. *Probabilistic robotics*, volume 1. MIT press Cambridge, MA, 2005.

[21] Miomir Vukobratovic and Miomir Vukobratovic. *Introduction to robotics.* Springer-Verlag Berlin, Germany, 1989.

[22] Teddy N. Yap and Christian R. Shelton. Simultaneous learning of motion and sensor model parameters for mobile robots. *2008 IEEE International Conference on Robotics and Automation*, pages 2091–2097, May 2008.