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

Table of Contents

Abstract . . . . **v**

Acknowledgements . . . . **vi**

Chapter 1 Introduction **1**

1.1 Motivation 1

1.2 Calibration and Localization 2

1.3 Motion Estimation 3

1.4 Contributions 4

Chapter 2 Background **5**

2.1 Motion Model 5

2.2 Particle Filter 8

2.3 Particle smoothing 10

2.4 SURF 11

Chapter 3 Learning the motion model **12**

3.1 General Architecture 12

3.2 Dynamic Landmarks 13

3.3 Parameter Estimation using Expectation Maximization 14

3.4 Visual Motion Estimates 16

3.5 Results 18

Chapter 4 Conclusion **23**

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 the person to know its position in the surrounding environment. Similarly for robots the need to know its state in the world is an important task and it is termed as localization. To interact with the environment we need some sort of understanding of it.Primarily mapping the environment and localizing the robot in it i.e. Simultaneous Localization and Mapping (SLAM) has been a way to interact with the world. The integral part of SLAM is the way the robot moves and senses the world. These characteristics are captured in the motion and sensor models. However infrequently discussed but essential input are the parameters to these models. Most of the models are considered to be static which is a wrong assumption to make. For example the movement of the robot on carpet is very different compared to tiles. Another example would be the models can change from general wear and tear on the robot. Generally the motion and sensor models are hand tuned by experts based on their experience. The whole process can be tedious and we propose a way to automate it. As we are moving ahead in our goal of making robots a ubiquitous part our everyday life they need to adapt to our dynamic surroundings. The algorithm that we are proposing not only will help automate the process but also help us adapt the models to our environment.

Water environments like oceans,rivers are highly dynamic and change in a matter of seconds which affects the movement of the vehicles.For example in Autonomous Underwater Vehicle(AUV) the density and the temperature of the water can lead to changes in the motion model. For my thesis we will specifically deal with underwater environments and propose an online system for an AUV to adapt its motion model. We specifically dealt with AUV that are equipped with side sonar sensor.

Sound Navigation and Ranging (SONAR) is a technique based on sound propagation used for communication,detecting objects underwater. The SONAR sensor is the only imaging tool which can work at high depth. The side sonar images in my thesis is used for two applications. In the first application the side sonar images can be used to gather information for our sensor model. In second application by matching features on consecutive side sonar images we can get an estimate of the movement for the AUV.

To learn the right parameters of the model we use Expectation Maximization(EM). It is an unsupervised machine learning technique primarily used to estimate parameters. It alternates between the Expectation Step which 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 E step.

For visual motion estimate we use Speeded Up Robust Features(SURF) to extract interest points in a side sonar image. There are robust local feature detector and is used in computer vision tasks like object recognition or 3D reconstructions.We extract key points and match them in subsequent images to estimate motion of the AUV.

To summarize I am proposing a algorithm to learn the right parameters of the model specifically to deal with changes underwater. We are learning these parameters on the fly without having a prior map or revisiting places. We also put forward an idea to estimate the movement of AUV from side sonar images. Results of simulated experiments with environment changes are presented to show the effectiveness of the algorithm. Real mission datasets are used to validate our motion estimation algorithm.

1.2 Calibration and Localization

Localization can be termed as the process of estimating the robot position and orientation in the world. It is the answer to the question Where am I?. The correct estimate of its position is very crucial for the robot to achieve its goal. A erroneous estimate can lead to hazardous and fatal situations for the robot. Generally a prior map is provided and the robot is equipped with sensors to perceive the environment and localize itself in it. This process is an integral part in the autonomy of the robot. Various algorithms like Particle Filter, Kalman Filter help us in estimating the state of the robot in an environment.

In our framework we use particle filters to localize the robot. We define a motion and a sensor model for our robot. Instead of a map we use dynamic landmarks to gather sensor readings. The motion and sensor measurements are then processed by particle filters to estimate the pose of the robot.

Calibration is the process of determining the parameters to the kinematic or dynamical model. Better the calibration of the robot higher the accuracy of localization. Generally the robot is calibrated at the start of the experiment and the parameters values are not changed throughout the experiment. As our environment is dynamic our models whould also be adaptive.Thrun first proposed an algorithm for online calibration using the maximum likelihood estimate. Alziar further continued the work and learned the motion model for land robots using Expectation Maximization framework. The EM framework was further extended by Teddy Yapp to learn the right parameters for motion as well as sensor model.

As stated earlier underwater environment is highly dynamic.Therefore we use the same framework to estimate the motion models for underwater vehicles. We use side sonar images to extract landmarks and use it as our sensor data. Particle smoothing is performed to get us a set of trajectories. We perform maximum likelihood estimate of the parameters given the trajectories and actual data.

1.3 Motion Estimation

In mobile robots the position and orientation is determined through wheel encoders and velocity estimates. These techniques do not generalize well as they cannot be applied to every robot. On top of that the wheels tend to slip on the floor and also the estimates drift over time. As the errors accumulate, the motion estimation becomes unreliable. We propose a method to estimate motion from only side sonar images.Compared to other techniques they are not restricted to particular locomotion method as well as don't suffer from drift.In our approach we extract and map key points between consecutive image in underwater environment . We use Speeded Up Robust Features (SURF) algorithm first proposed by Herbert Bay in 2006 to extract interest points in the image.

Silvia Silva da Costa Botelho proposed a similar algorithm which used camera on AUVs to estimate visual motion and Scale Invariant feature transform(SIFT) was used to extract key points.

There are two major differences in our algorithm. We used side sonar images because at deeper levels in the water cameras don't produce any meaningful images. Secondly we used SURF which is several times faster than SIFT and is more robust against image transformations.

1.4 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 we use only a limited amount of interest points which lead to lesser computation and memory usage.

Chapter 2

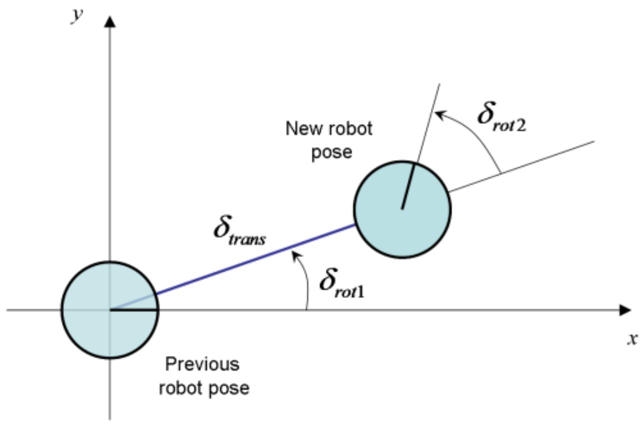
Background

2.1 Motion Model

A motion model is responsible for capturing the relationship between the control input and the change in robot's configuration. Our approach models the motion of the 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 an integral part of algorithms such as localization,mapping etc.

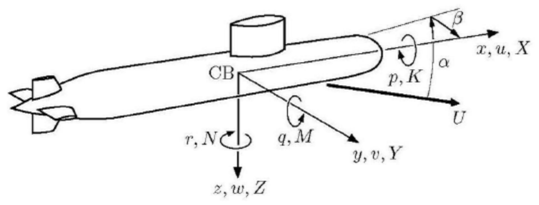
Let $X = (x, y, \theta)$ be the initial pose of the robot in x-y space. Mathematically the motion model can be described as $P(X^{'}|X, u)$, where $X^{'}$ is the pose after executing the motion command u. Based on the control input we can divide the motion model in two classes 1) Odometry based motion model 2) Velocity based motion model. The first class of motion models are used for robots equipped with wheel encoders. Velocity based models calculate their new position based on velocities and time elapsed.

Figure 2.1: Markov Chain



For Autonomous Underwater Vehicle(AUV) the pose is represented in 6 Degrees Of Freedom(DOF). The pose can be represented as $s = (x, y, z, \theta, \phi, \psi)$. The first

Figure 2.2: Body-fixed reference frames



three coordinates correspond to the position along the x,y,z axes while the last three coordinates describe the orientation. Generally for marine vehicles these motion components are defined as surge,sway,heave,roll,pitch and yaw.

The 6-DOF rigid body equations of motions are

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

$Y = m[v^{\cdot} - wp + ur - y_G(r^2 + p^2) + z_G(qr - p^{\cdot}) + x_G(qp + r^{\cdot})]$

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

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

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

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

The first three equations represent the translational motion and the last three represent the rotational motion.X, Y, Z, K, M, N are the external forces and moments of external forces. u, v, w and p, q, r represent the linear angular velocity of X, Y, Z respectively. x_G, y_G, z_G represents the center of gravity.

The above equations can be represented in a more compact and vectorial form.

$$\boxed{M_{RB}\mathcal{V}^{\cdot} + C_{RB}(\mathcal{V})\mathcal{V} = \tau_{RB}}$$

Here $\mathcal{V} = [u, v, w, p, q, r]^T$,$\tau_{RB} = [X, Y, Z, K, M, N]$,$M_{RB}$and C_{RB} are the inertia matrix and centripetal matrix.

The forces acting on AUV can be broken down into three classes-:

- Radiation-induces forces
  - added inertia
  - hydrodynamic damping

  - restoring forces

- Environmental Forces

  - Ocean currents
  - Waves
  - Wind

- Propulsion Forces

  - Thruster/Propeller Forces
  - Control Surfaces/Rudder Forces

The external forces and moments vector $\tau_{RB}$is the sum of the forces listed above

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

Here τ_H is the radiation induced forces and moments, τ_E and τ are the environmental and propulsion forces and moments.

The control is the input to a motion model and in our case it is the velocity of the AUV. The velocity is calculated from the propellers. In our algorithm we assume a static model $u_r = f(n_s)$ to get a velocity estimate.u_r, n_s are the water relative linear velocity in x direction and propeller rotation rate. This can be implemented as a table look-up based on experimental data. The same control input won't product the same output every time as it is dependant upon the environment. Therefore we assume a Gaussian distribution over the velocity and try to learn the right parameters over time. For my master's thesis we reduce the degrees of freedom and represent the pose of the AUV is two dimensional with orientation.

In the algorithm the robot can be represented by $X_t = (x_t, y_t, \theta_t)$ where x_t, y, θ_t are the robot's coordinates in x and y plane and orientation at time t. We base our motion model equations on Teddy N. Yap motion model which specifically dealt with odometry to update the pose of the robot. In the set of equations for our model we replace the odometry with velocity.

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

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

$\theta_t = \theta_{t-1} + W_{t-1}\delta t$

The terms V_t and $W_t$are the translational and rotational velocities respectively. They are represented by a Gaussian distribution to account for the change in forces. They can mathematically represented as

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

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

In the above equations v_t and $w_t$are reported translational and rotational velocity. σ_{A_b} describes the contribution of velocity term b to the variance of the distribution over A. $\sigma_{V_1}$and σ_{W_1} take into account the independent errors that are not proportional to translation and rotation of the robot. $\sigma_{V_v}^2, \sigma_{W_v}^2, \sigma_{V_w}^2, \sigma_{W_w}^2, \sigma_{V_1}^2, \sigma_{W_1}^2$ are the motion parameters that our algorithm intends to learn.

2.2 Particle Filter

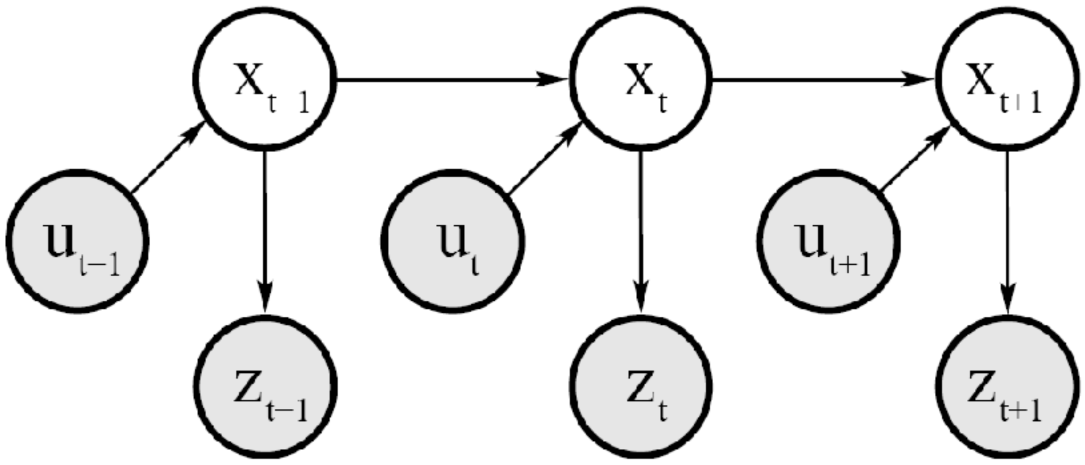
Particle Filter is a state estimation algorithm based on a sampling method for approximating a distribution. It is an alternative non-parametric implementation of the Bayes filter. It also can be called as a sequential Monte Carlo algorithm. It was first proposed by MK. Pitt and N Shephard[refer the paper].

Particle Filters have no restrictions in model i.e. it can be applied to non-Gaussian models,they are super set of other filtering methods i.e. Kalman filter is Rao-Blackwellized particle filter with one particle.Additionally they can model non linear transformations of random variables therefore can be applied to any systems and measurement models. All the advantages make them a good alternative to Extended Kalman Filter(EKF) and Unscented Kalman Filter(UKF).

Particle Filters are used to calculate the approximate of a state in Markov chain. The key idea behind the particle filter is to represent the posterior bel(x_t) by a set of random state samples drawn from this posterior.The state $x_t = p(x_t|x_{t-1}, u_t)$ is dependant upon x_{t-1} as well as the control input u_t. The observations are introduced in the system in the form of measurements.The $p(z_t/x_t)$ are used to assign weights to the particles and give survival of the fittest mentality to the algorithm.

The algorithm for particle filters is described below-:

Figure 2.3: Markov Chain



```
Input: $X_{t-1}$: particle set $u_t$: most recent control $zt$: 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

The importance factor for each particle x_t^m is calculated denoted w_t^m. Importance is the probability of the measurement z_t under the particle x_t^m. Thus importance factor are used to incorporate the measurement into the particle set. In practice, the number of particles used is a large number(e.g.-:1000).

The key part of the algorithm is the re-sampling step which are line 8 through 11 in the 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.

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. It is defined as the computing the posterior distribution of the past states given all the evidence upto the present. Mathematically it can be represented as $p(x_t|u_{1:T}, z_{1:T})$ for some t with $0 \leq t \leq T$.Russel and Norving 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 which was based on the technique presented by Docuet et al. and Godsill et al. The algorithm is used to generate samples from the entire joint smoothing density $p(x_{0:T}|u_{1:T}, z_{1:T})$.This algorithm assumes that particle filtering is already done on the entire data set and resulting in a distribution of the pose.

```
Input: $X_t, t = 0, 1, ..., T$: particle approximations to the posterior pdfs
        $p(x_t|c_{1:t}, s_{1:t}), t = 0, 1, .., T$
$c_{1:T} = (c_1, c_2, ..., c_T)$: 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
        density $p(x_{0:T}|c_{1:T}, s_{1:T})$
begin
    draw $i$ with probability $\propto w_T^{[i]}$ $x'_T \leftarrow x_T^{[i]}$
    for $t \leftarrow T - 1$ down to 0 do do
        for $i \leftarrow 1 to N_s$ do do
            $w_{t|t+1}^{[i]} \leftarrow w_t^{[i]} p(x'_{t+1}|u_{t+1})$
        end
        draw $i$ with probability $\propto w_{t|t+1}^{[i]}$
        $x' \leftarrow x_t^{[i]}$
    end
end
```

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

The particle filter process gives us a set of particles with their corresponding weights at the present time step. In the first step of smoothing we choose a particle

with the probability proportional to the weight of the particles. Then we move a time step back till time-0 and calculate the new smoothed weights for every particle. They are calculated by the product of the forward probability and the weight of the particle.
$p(x_t|x_{t+1:T}, u_{1:T}, z_{1:T}) = p(x_{t+1}|x_t, u_{t+1})p(x_t|u_{1:t}, z_{1:t})$
To calculate the trajectory we draw particles according to the new smoothed weights. At every time step we pick a particle and add it our trajectory set till time=T. We can repeat this process any number of time to get several trajectories. These trajectories are used in estimating the parameters of the motion model as they are treated as ground truth.

2.4 SURF

Still have to write about it

Chapter 3

Learning the motion model

3.1 General Architecture

Figure 3.1: Block Diagram of the framework

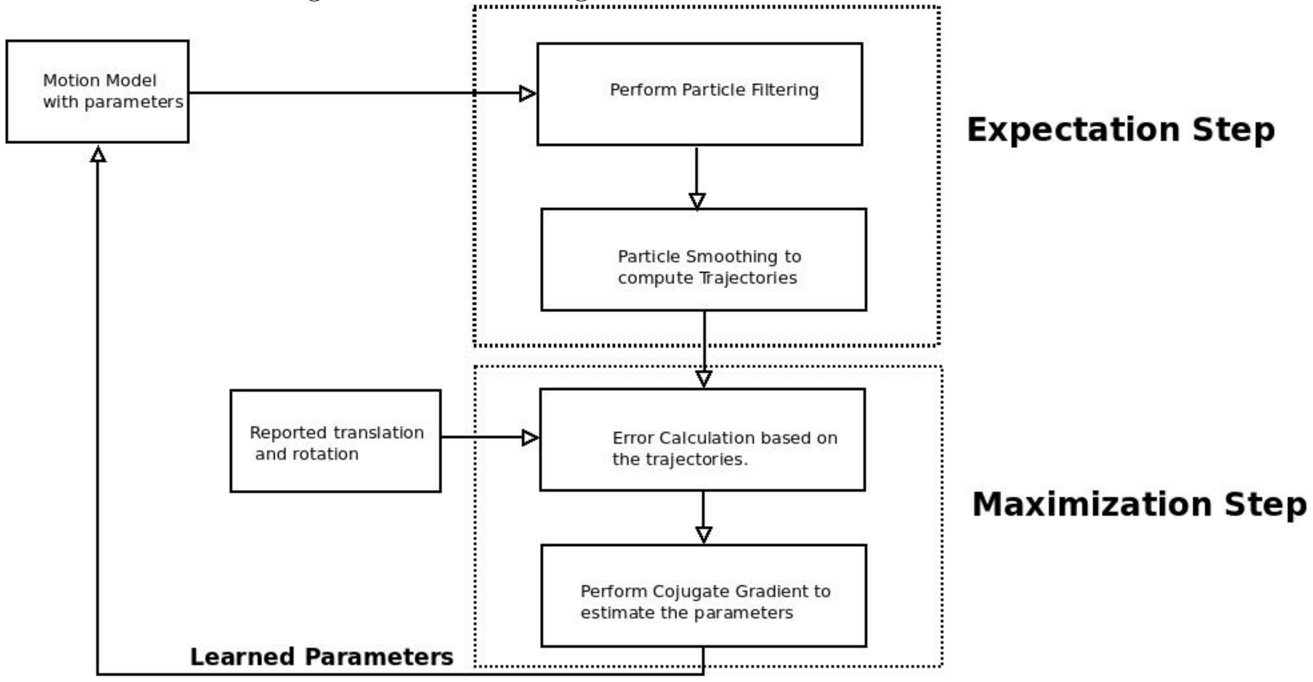

Motion Model with parameters
Perform Particle Filtering
Expectation Step
Particle Smoothing to compute Trajectories
Reported translation and rotation
Error Calculation based on the trajectories.
Maximization Step
Perform Cojugate Gradient to estimate the parameters
Learned Parameters

Figure 1.1 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 the 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.

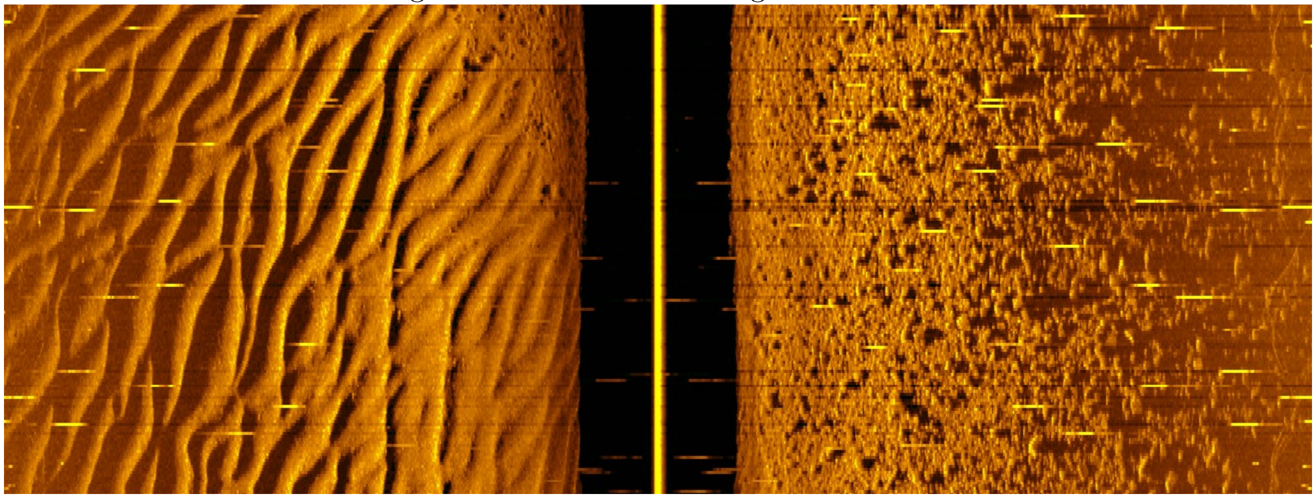
Based on the trajectories and the reported translational and rotational motion we calculate errors at each time step. We perform newton conjugate gradient on the errors to give us the best estimate of the parameters. This completes the Maximization Step.

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 SURF,Kalman Filter are dependant 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 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

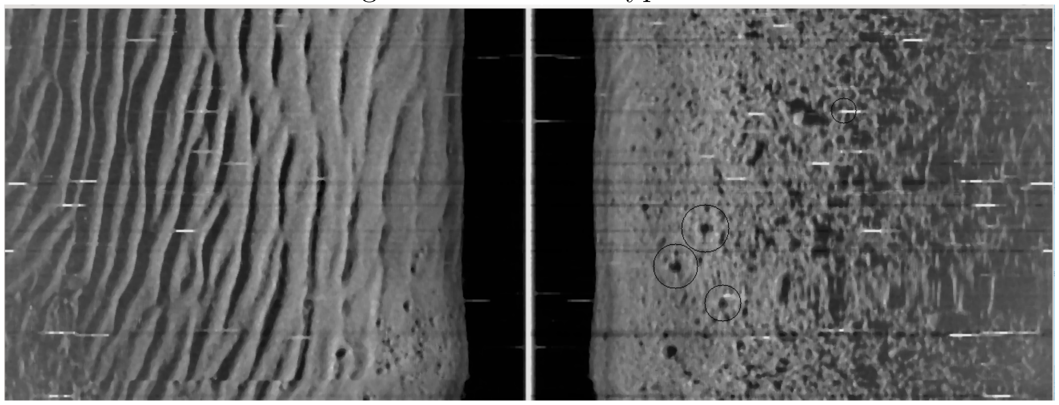
Figure 3.2: Side Sonar Image



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 3.3: SURF Keypoints



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.

3.3 Parameter Estimation using Expectation Maximization

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. 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. As stated earlier the goal is to maximize the likelihood function

$p(X|\theta)$ with respect to θ.

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 to step 4. In our algorithm we use Newton Conjugate Gradient to estimate the right parameters.

As we know the output of Expectation 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 reported by the encoder.

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

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

$\epsilon_{T_t}^{[j]}$ and $\epsilon_{D_t}^{[j]}$ are rotational and translational errors for t=0,1...T-1 for j^{th} sampled trajectory. $r_t^{''}$ and $d_t^{''}$ are the reported odometry values.

Based on the motion model described

$\epsilon_{D_t}^{[j]} \sim \mathcal{N}(0, d_t^{''2} \sigma_{D_{d''}}^2 + r_t^{''2} \sigma_{D_{r''}}^2 + \sigma_{D_1}^2)$

$\epsilon_{T_t}^{[j]} \sim \mathcal{N}(0, d_t^{''2} \sigma_{T_{d''}}^2 + r_t^{''2} \sigma_{T_{r''}}^2 + \sigma_{T_1}^2)$

The log likelihood functions are

$\mathcal{L}(\sigma_{D_d}^2, \sigma_{D_r}^2, \sigma_{D_1}^2) = -\frac{1}{2} \sum_j \sum_{t=0}^{T-1} [\log 2\pi + \log(d_t^{''2} \sigma_{D_d}^2 + r_t^{''2} \sigma_{D_r}^2 + \sigma_{D_1}^2) + \frac{(\epsilon_{D_t}^{[j]})^2}{d_t^{''2} \sigma_{D_d}^2 + r_t^{''2} \sigma_{D_r}^2 + \sigma_{D_1}^2}$

$\mathcal{L}(\sigma_{T_d}^2, \sigma_{T_r}^2, \sigma_{T_1}^2) = -\frac{1}{2} \sum_j \sum_{t=0}^{T-1} [\log 2\pi + \log(d_t^{''2} \sigma_{T_d}^2 + r_t^{''2} \sigma_{T_r}^2 + \sigma_{T_1}^2) + \frac{(\epsilon_{T_t}^{[j]})^2}{d_t^{''2} \sigma_{T_d}^2 + r_t^{''2} \sigma_{T_r}^2 + \sigma_{T_1}^2}$

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.

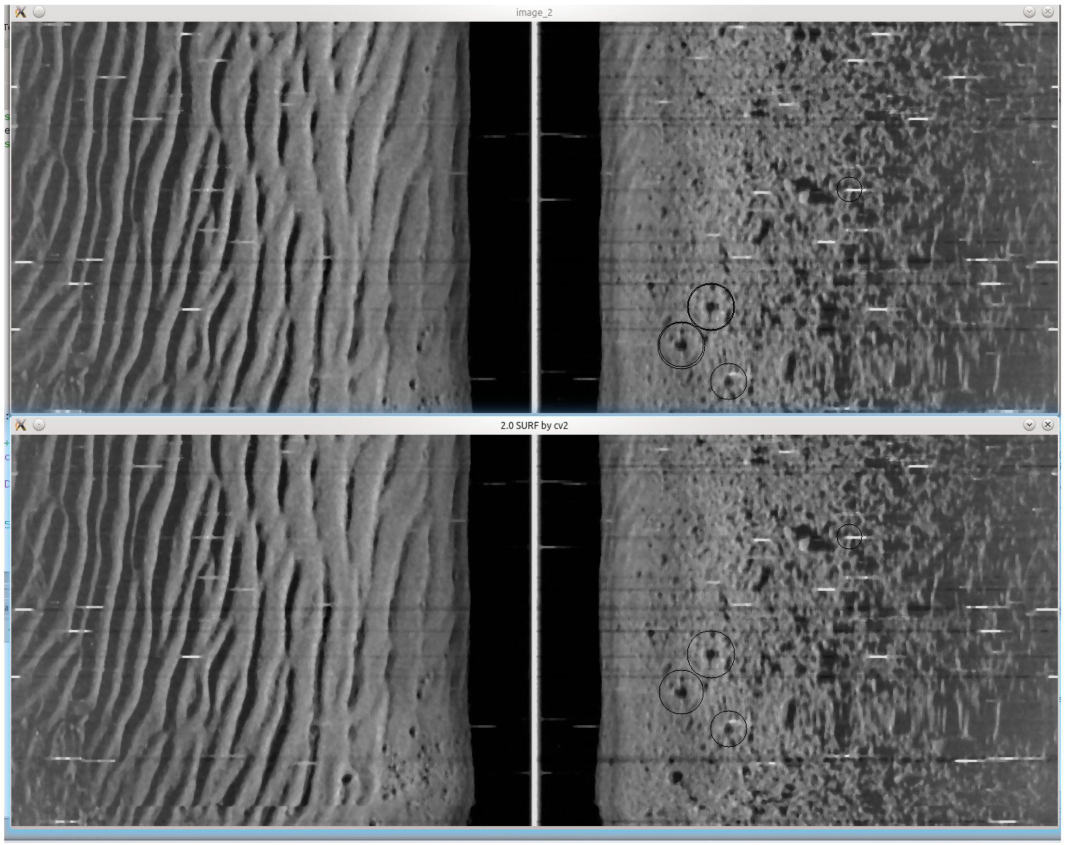
3.4 Visual Motion Estimates

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 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.

Figure 3.4: SURF Keypoints



3.5 Results

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

σ_{D_d},σ_{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 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 5th 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

Figure 3.5: σ_{D_d}=0.05 $\sigma^*_{D_d}$ = 0.2 Sensor Noise= 2.0

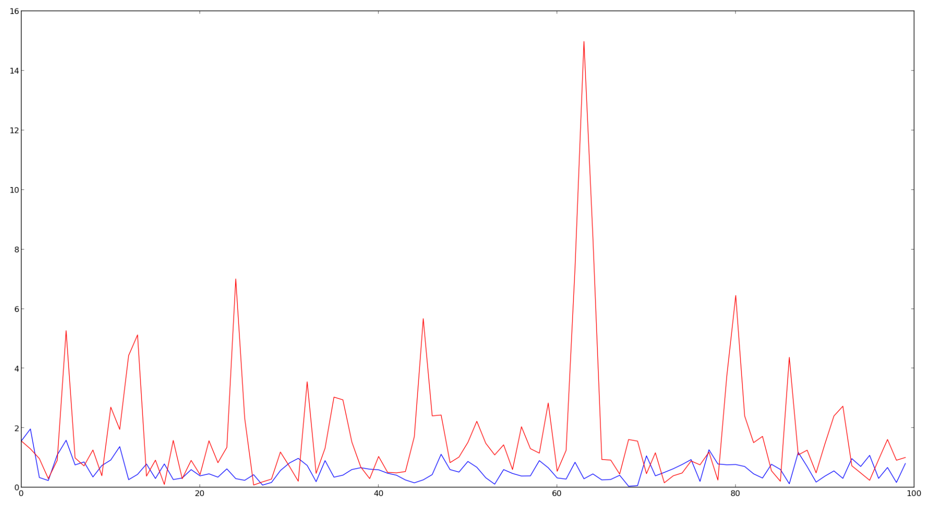


Figure 3.6: σ_{D_d}=0.05 $\sigma^*_{D_d}$ = 0.2 Sensor Noise= 5.0

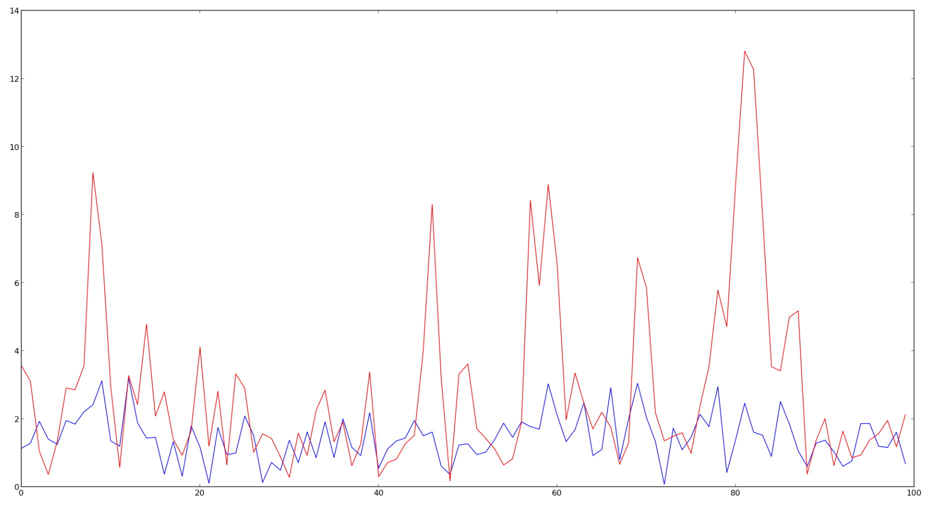


Figure 3.7: σ_{D_d}=0.05 $\sigma^*_{D_d}$ = 0.5 Sensor Noise= 2.0

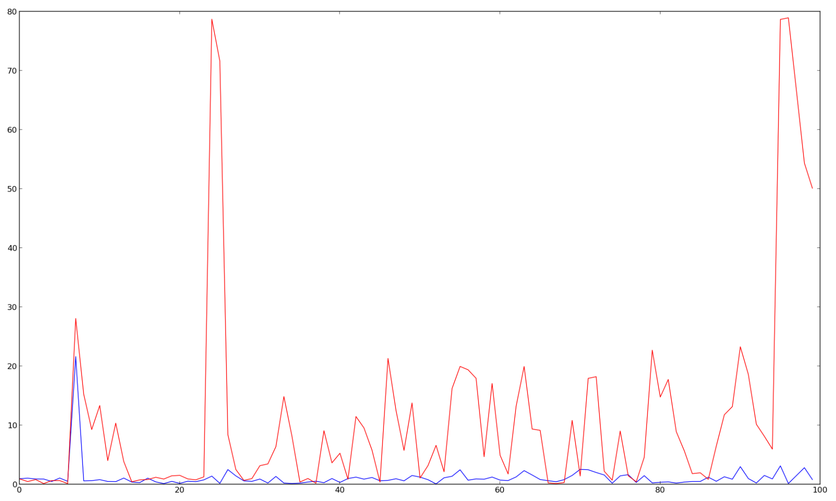


Figure 3.8: σ_{D_d}=0.05 $\sigma^*_{D_d}$ = 0.5 Sensor Noise= 5.0

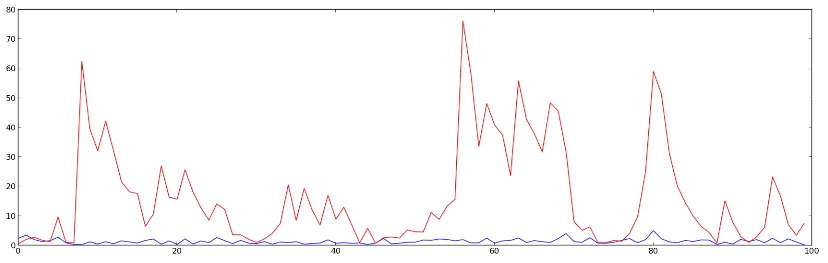
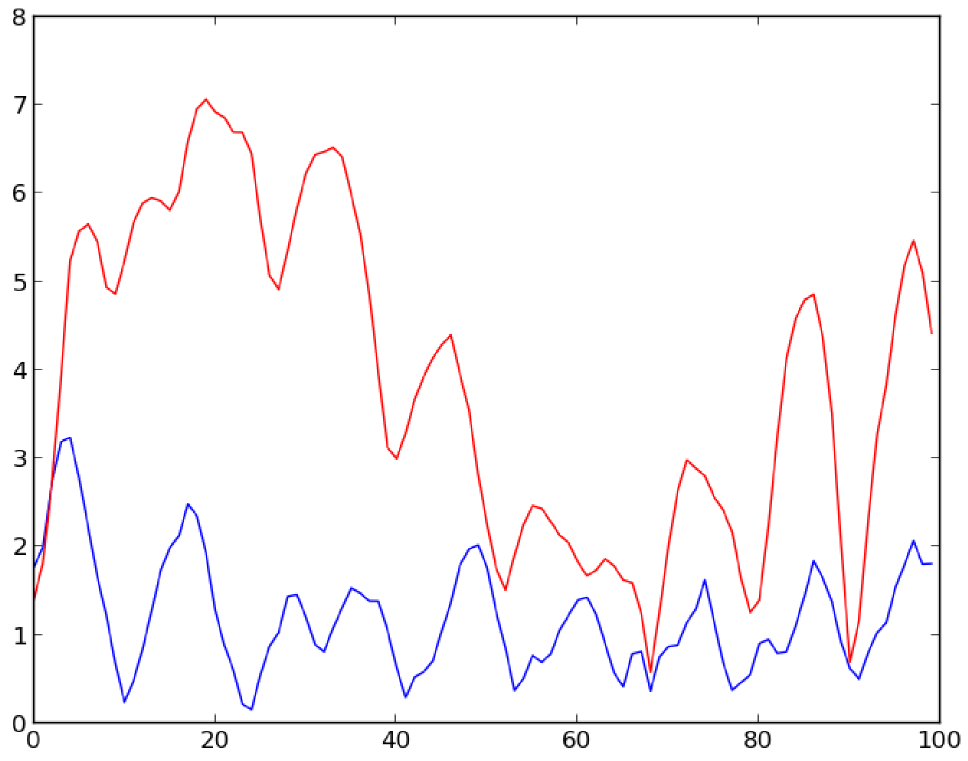


Figure 3.9: σ_{T_r}=0.05 $\sigma^*_{T_r}$ = 0.2 Sensor Noise= 5.0

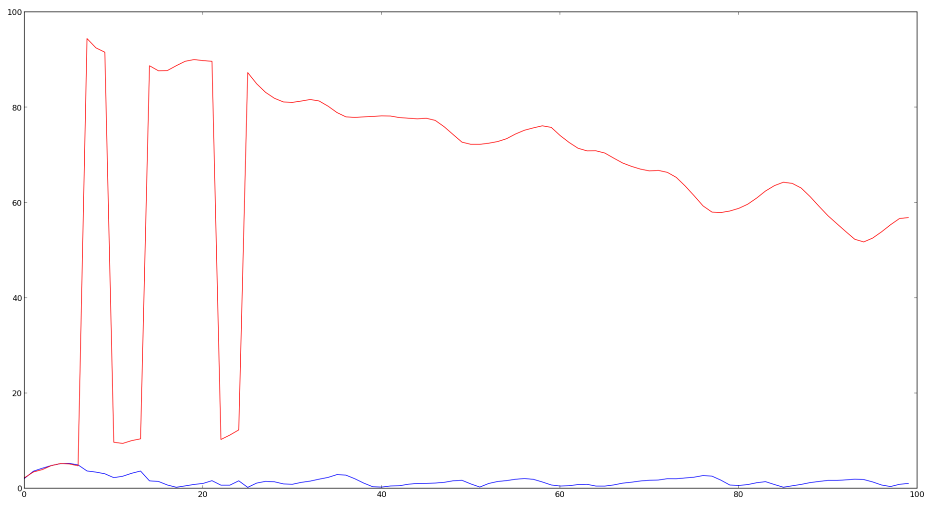


can see that the error is pretty static in the learned motion model whereas in the static motion model there is a lot of fluctulation.

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 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.

Figure 3.10: σ_{T_r}=0.05 $\sigma^*_{T_r}$ = 0.5 Sensor Noise= 5.0



Chapter 4

Conclusion

Did it!