

SLAM - Implementation and Benchmarking



Mangal Deep .B.M

7206937

Submitted in partial satisfaction of the requirements for the
degree of

M.Eng Embedded Systems for Mechatronics

School of Informatics

Dortmund University of Applied Sciences and Arts

15-3-2021

A blank page

This is an optional page environment you could use for things like:

- Your own custom preamble chapters (use `\chapterTitle` for titles!)
- *“This work is dedicated to Dad and Mom”*
- A copyright notice
- Additional notes
- Quotes
- List of publications
- An actual blank page
- Nomenclature / glossaries, etc.

It is not recommended to use this in your undergraduate thesis for submission. It has been left in here to make you aware of its existence. This is largely because it does not follow the guidelines set out for undergraduate theses, but you could include this environment for your own personal printed copy. Consult with your supervisor to see if you can use it.

In addition, you can pass an optional parameter to this environment with the value `c` to centre this text vertically on the page (use the `center` environment to align horizontally).

Abstract

This is where your abstract will go. Usually this is written last, after writing the entirety of your thesis.

Acknowledgements

Firstly, I want to thank Dad and Mom.¹ Here is another note.

¹Here is a footnote

Table of Contents

Chapter 1

Introduction

Simultaneous Localization and Mapping, commonly abbreviated as *SLAM* is one of the fundamental and essential task of any mobile robots which helps in building the Map of the environment without the knowledge of actual location and orientation of the robot. It is one of the key initial step for the development of navigation systems for any mobile robot. It is also referred as the chicken or the egg problem, as it requires precise location of the robot in the environment, to get an accurate map of the environment, but in order to determine the precise location, the map of the environment is required. This complexity has made it harder to find a precise solution, accurate localization and mapping.

The complex task of localizing and mapping the surrounding has triggered a huge interest widely in the research field and it continues to improve as the technology evolves in both software and hardware. Over the past decade numerous strategies and methods were developed to solve the SLAM problem.

A typical SLAM system uses a combination of odometry data, GPS, IMU for localization and Camera, LiDAR or Radar for perception. The information obtained from these sensors are fused to obtain a full estimate on the state of the system. Though technologies evolved in acquiring precise measurement uncertainties and errors are still prevalent due to various factors. Hence designing a solution considering the uncertainties in the measurement is a key to achieve a good estimate of the location

and map of the surrounding. Therefore a probabilistic approach to the solution is the key to globally consistent Map.

The SLAM problem can be classified on various basis. SLAM problem can be broadly classified into *Online SLAM* and *Full SLAM*. In the *Online SLAM* problem, only the current state of the robot is estimated along with Map. In the *Full SLAM* problem, the entire state of the path traversed by the robot is estimated. Based on the setup of environment to the mapped the SLAM system can be broadly classified into *Landmark-based SLAM* and *Grid-based SLAM*. In *Landmark-based SLAM*, the robot knows the location of landmarks *a priori*. However in the real world scenario, the robot should establish a correspondence between the expected landmark measurement and the actual landmark observed. The data association plays a key role in assigning the measurement to its corresponding landmark. With the known correspondence and known pose of the robot relative to a landmark, the problem reduces to Mapping with known poses. The *Landmark-based SLAM* is widely applied in many robotics application for instance, underwater exploration, mapping the underground mines. These SLAM system commonly use Ultrasonic sensors, camera for perception and GPS, odometry are used to find the motion of the robot.

In absence of landmarks, *Grid-based SLAM* provides a complete SLAM solution with the help of 2-D or 3-D laser scanners. The observations made by the LiDAR are converted to a obstacle map also referred to as occupancy map. These observations are also used to correct the pose estimate of the robot. This SLAM approach provides a more useful way to map the surroundings as landmarks cannot be installed and observed in many challenging environment. More details on this approach will be provided in the later chapters.

Numerous research has been done in the SLAM community. Some of the implementations are made available to the public in ROS. The most common are GMapping and Cartographer

This paper presents insights on Particle Filter based and Graph based solutions to SLAM problem. These approaches are implemented, tested and results are com-

pared thoroughly to provide detailed report on throughput of the algorithms. The algorithm is designed and the testing is performed on the data set obtained from CARLA. An autonomous driving car mounted with sensors such as LiDAR, GNSS and IMU. The Lidar is the perception system in the simulated car. It is a 360 degrees rotating 3D LiDar with 32 beams with a range of 50meters rotating at a frequency of 20Hz. All the sensors are placed at the same geometrical position on the vehicle and are oriented in the same direction. This avoids transforming measurements to different coordinate systems.

The system is developed for an autonomous driving car trying to create a 2D Map of the simulated environment. Hence from this point Map generally refers to Grid Map, Pose refers to Special Euclidean (SE2) and orientation, unless specified. The paper is organized as follows, The detailed literature review is provided on existing SLAM algorithms and applications, variants of SLAM in chapter 2. Probabilistic formulation of the SLAM problem is derived in chapter 3. Particle Filter based solution to SLAM is discussed and different implementations such as Rao-Blackwellized Particle Filter and GMapping algorithm are discussed in detail and analyzed in chapter 4. Graph based solution to SLAM is discussed and its various flavors are discussed in detail in chapter 5. Comparison and benchmarking of the above methods are performed based on test results and computation is provided in chapter 6. Concluding remark on all the observations made so far in the paper is discussed in chapter 7.

Chapter 2

Literature review

The state of a robot such as position, orientation, velocity, acceleration, parameters or even a map, is critical to get the accurate results. Needless to mention, the more information the robot knows about itself, the solution to find the unknowns becomes easier. In the context of SLAM, the pose information and the Map of the environment are the states of absolute interest. Given the high precision pose information of the robot, then the problem simplifies to Mapping of the environment with known poses. Here the state of interest is only Map which does not even require state estimation rather use the available Mapping algorithm to draw the Map of the environment. On the other hand, given the Map of the environment, then the problem reduces to localization problem with data association in certain cases. Here the state of interest is Pose and the correspondence term. A survey on various *SLAM* methods were presented by **C.Cadena** and **T.Takleh**. According to **C.Cadena**, the history of SLAM is broadly classified into *classical age* consisting of *Extended Kalman Filter (EKF)* methods, *Rao-Blackwellized Particle Filter (RBPF)* methods and *Maximum Likelihood Estimation*. Followed by *algorithmic-analysis age* which predominantly studied the Observability, convergence and consistency of SLAM, leading to efficient SLAM Solvers and many open source libraries. Followed by *robust-perception age* which focuses on Robust performance, Resource awareness, high level understanding and Task-Driven perception. Finally In my perspective, With the development and uprise in the application of Deep Learning techniques in SLAM, It would be suitable to term the current trend as *Machine Learning era of SLAM*.

In the beginning, hardware capabilities were very less and compute power embedded in mobile robots were limited to fewer processes at an instance. Hence the choice of sensors to be mounted on the robots were very critical. Historically the very first feasible solution to the SLAM problem proposed the use of Extended Kalman Filters(EKFs) by estimating the joint posterior distribution over pose of the robot and the landmark positions. Kalman Filters are widely used state estimators but are limited to linear system and works on the assumption of Gaussian distribution of data. Therefore they cannot be directly applied to solve the non-linear SLAM problems. By linearizing the data at the mean of the distribution it is possible to approximate the solution, the Extended Kalman Filter works on this principle. The state estimate and the associated uncertainty are provided by the mean and covariance matrix of the posterior of the joint probability. However the odometric drift observed by the robot seems to hold down the accuracy of the posterior estimate. **J.Kang** proposed modified Neural Network aided EKF (mNNEKF) to improve the accuracy of the estimate by compensating for odometric error of the robot using Neural Network. This method claims to work well even under colored noise and systematic bias error.

Chapter 3

Formulation

Introduction

SLAM is analogous to Dynamic Bayes Network as shown in figure.. The Pose in-

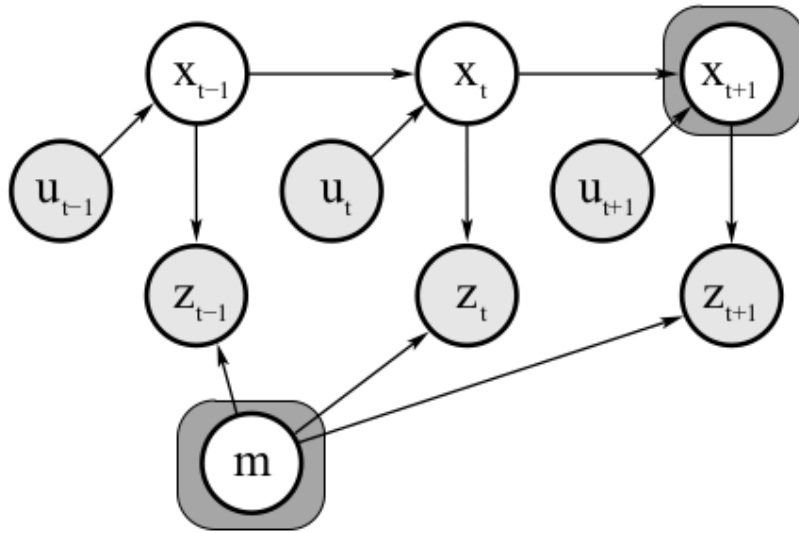


Figure 3.1: Dynamic Bayes Network for Online SLAM

formation of the robot at time instant t is represented as x_t which constitutes (x, y, θ) . where (x, y) correspond to the location and θ represent orientation in the 2D Plane. Let u_t be the Odometry reading of the motion executed between time $t-1$ and t . Let Z_t be the LiDAR measurement taken at time t . Let m be the Map created from the set of measurements. In the figure ??, the empty circles denote the states to be estimated and the shaded circles denote the variables that can be measured. Considering

the circles to be the nodes and the arrows as edges, It is seen that the previous information on the state and the present command determines the current state of the system, which then influences the measurements obtained from the map. The nodes m and $x_t + 1$ are the required output parameters. There are three basic fundamental approaches to the SLAM problem, namely, *Kalman Filter*, *Particle Filter*, *Graph Network*. Each approach to the problem have their specific advantages and disadvantages, which will be discussed briefly in this chapter and extensively in upcoming chapters. However all the methods address the basic problem - Given the relative distance moved as recorded by the odometer and the observation at the current location, the system must be able to correctly Localize and Map its environment. In this chapter we will also look at how the probabilistic framework helps in achieving the task in hand and also the components commonly used across all the SLAM methods.

3.1 Probabilistic approach to state estimation

In solving the Online SLAM Problem, the joint posterior distribution over the current pose and the map is the actual state of interest.

$$p(x_t, m | z_{1:t}, u_{0:t-1}) \quad (3.1)$$

whereas to solve the full slam problem, the joint posterior distribution over all the poses the robot had traversed and the map is the actual state of interest.

$$p(x_{1:t}, m | z_{1:t}, u_{0:t-1}) \quad (3.2)$$

The distributions ?? and ?? are applicable only to the grid based mapping. For Feature-based mapping the correspondence between the landmark measured and the measurement needs to be established. This is a data association problem and it could be estimated as a part of the posterior distribution.

$$p(x_{1:t}, m, c_t | z_{1:t}, u_{0:t-1}) \quad (3.3)$$

As this work is applicable only to Grid-based mapping, Henceforth the derivations and equations are with respect to it without being mentioned explicitly. Applying the Bayes rule to determine the joint posterior distribution

$$p(x_{1:t}, m | z_{1:t}, u_{0:t-1}) = \alpha.p(z_t | x_t, m). \int p(x_t | x_{t-1}, u_{t-1}).p(x_{1:t-1}, m | z_{1:t-1}, u_{0:t-2})dx_{1:t-1} \quad (3.4)$$

However estimating the states using ?? becomes computationally expensive when integrating over all the previous poses and observation. This could be overcome by use of Extended KalmanFilter under three assumptions *Feature-Based Maps* such that the number of landmarks are less , *Gaussian Noise Assumption* such that the noise levels in the robot motion and perception is in limits that does not disturb the linearization of EKF and *Positive Information* such that unseen landmarks do not influence the estimation. In many practical applications the correspondence of a landmark to the measurement is not known. In such cases data association has to be derived during runtime. In such cases, the correspondence is also estimated in the posterior distribution as shown in ??.

In the online SLAM process , as the robot moves through the environment, the system state vector and the covariance matrix are updated upon new measurements. On observing new landmarks, new state variables are added to the system state vector and the covariance matrix. The EKF involves State Prediction and Correction step for every time a new information is received. The State Prediction and Correction steps are elaborated in subsequent sections. However the EKF has their own limitations. First, The covariance matrix maintained by the KalmanFilter has $O(K^2)$, where K is the number of landmarks. Thus an increase in the number of the landmarks results in longer sensor updates. Second, The correspondence of the measurement and the landmarks is assumed to be known. Any wrong data association leads to filter divergence.

Overcoming the problem, Rao-Blackwellized Particle Filters were used as a effective means to estimate the full posterior distribution. Instead of working on the entire distribution, particles were sampled at random and with sufficient number of particles

the entire distribution could be covered. Here it is possible to relax the Gaussian assumption made in the EKF SLAM, as any arbitrary distribution could be modelled as the Target distribution. It basically involves steps such as Sampling, Importance Weighting, Resampling and Map Estimation. Sampling from proposal distribution corresponds to State prediction step and Importance Weighting corresponds to State Correction steps in EKF. More detailed discussion on RBPF is provided in the next chapter. This method is applicable to both Grid based and Feature Based mapping with fewer modifications.

The third category of SLAM ...

3.2 State Prediction


State prediction is the key component in predicting the motion of the robot in case of EKF and Filter based SLAM methods, also it helps in creating the graph in front-end of graph based SLAM. The mobile robot is assumed to be equipped with odometer which provides relative motion of the wheels between time instances and a LiDAR sensor which can perceive the environment. Consider the Positional state of the robot x_{t-1} at time $t - 1$, when subject to motion command u_{t-1} the robot takes a new positional state x_t . The change in the positional state of the robot can be determined by use of motion models derived from translational and rotational kinematics or by using odometers. In equation ??, the conditional density $p(x_t|x_{t-1}, u_{t-1})$ denote the state prediction part of the equation which could be replaced with appropriate motion models to predict the state of the robot in motion. The motion models derived will also need to factor the noise and uncertainty in its prediction, this is captured in the process noise covariance matrix.

3.2.1 Odometry Motion Models

The odometers are basically wheel encoders which reads how much the wheels have moved through the environment on receiving the command. Hence it can only provide motion information post-the-fact, i.e. on receiving the motion command,

the robot executes the command and then the odometry information is obtained. In many practical scenarios the robot might not follow all the motion commands as instructed due to slippage, wear and tear or due to any external forces. Under any such circumstances the odometer provides much reliable information on the state of the robot. However in case of motion planning this behaviour is not beneficial as the pose information at the next instance is critical to create a motion plan. In such cases the kinematic model of the robot is used to predict the motion of the robot. Algorithm for state prediction using odometry based motion model is very widely used in many robotics applications. Algorithms presented in the book **Thrun98** **probabilistic** can be used off-the-shelf for motion estimate implementation with odometry. On the downside, it is very common to observe drift and slippage in the odometer information and yet, it is most widely used in many robotics and industrial applications.

3.2.2 Kinematic Motion Models

In applications such as motion planning the positional state of the robot is required to be known in advance for better planning and control. Depending on the number of tractions, dimensions of the robot the kinematic equations of the motion can be used to estimate the state of the system. **R.Schubert** provides a detailed study on many different motion models which falls largely under the linear and curvilinear models. Linear motion models assume only straight motions and do not take rotations into calculation, whereas the curvilinear models assume the robot takes a circular path at a constant radius only exception being Constant Turn Radius and Acceleration (CTRA) which assumes that the robot follows a clothoid. The Figure  provides the relationship and assumptions made by each of the motion models. The experiment results prove that curvilinear models provide better performance than linear models. Further CTRA shows better tracking results than the other curvilinear models. In **Polack**, similar studies were made comparing Kinematic Bicycle Model and Dynamic models with 9-DOF, It was found that the earlier models were comparable to the dynamic model at low speeds and low angular acceleration. However at higher speeds or lateral acceleration, the dynamic

models with higher degrees of freedom were able to accurately plan the trajectory. Simplifying the solution to the problem and considering the theoretical evidence in the literature, CTRA Motion Model is chosen as the motion model for state prediction in our task. The motion models derived are in continuous domain and these need to be discretized before it could be implemented on a computer. Methods such as Euler discretization or analytical methods are some of the common methods used in deriving discrete time representation of the process model and process covariance matrix. **D.Svensson** provides the derivation of the discrete time equations for the CTRA model along with the process noise covariance modeled. The states involved in the prediction are longitudinal position($x(t)$), lateral position($y(t)$), heading angle($\phi(t)$), speed($s(t)$), acceleration ($a(t)$) and yaw rate ($\omega(t)$). In continuous domain the prediction function is derived and the discrete time model is derived using linearized discretization approach. The prediction equations are given by

$$x_{k+1}^{CTRA} = \begin{bmatrix} x_k \\ y_k \\ s_k \\ \phi_k \\ a_k \\ \omega_k \end{bmatrix} + \begin{bmatrix} \Delta x_k \\ \Delta y_k \\ a_k T \\ \omega_k T \\ 0 \\ 0 \end{bmatrix} + v_k \quad (3.5)$$

where T is prediction time and $v_k \sim \mathcal{N}(0, Q_k^{CTRA})$ is the discrete time process noise that is a zero mean Gaussian noise with covariance Q_k^{CTRA} . For the complete derivation of the prediction function and the process covariance matrix refer to **D.Svensson**.

3.3 State Correction

The state of the system predicted using the motion models may not accurately define the state and its variance modeled using the process covariance matrix. In order to further determine the exact posterior distribution, a confidence or a correction factor is required upon the predicted state. State correction techniques help in achieving a

confidence or correction factor which helps in extracting precise information about the state predicted using the motion model. Various observation or measurement models can be chosen based on the sensor configuration present in the robot. In equation ??, the conditional density $p(z_t|x_t, m)$ denotes the state correction part of the equation which can be replaced with appropriate measurement models or even scan matching algorithms to estimate a state with confidence. Given the map m and pose of the robot x_t , the model must be able to summarise how the surrounding environment would seem to be. Numerous of research have been conducted on robot perception systems. Sensor systems such as LiDAR, Cameras, ultrasonic, radar are commonly used in various robotic applications depending on the environmental conditions where the robot is deployed.

3.3.1 Observation Models

In case of LiDAR, it is safe to assume that every beam is independent of each other, hence the noise parameters and the measurement errors can be individually modelled. The measurement model density can thus be written as

$$p(z_t|x_t, m) = \prod_{k=1}^k p(z_t^k|x_t, m) \quad (3.6)$$

where z_t^k corresponds to a single laser beam. The density $p(z_t^k|x_t, m)$ can be modelled using different algorithms. Generally it is a mixture of four distribution- Gaussian distribution at the incidence of an obstacle, Exponentially decaying distribution to model unexpected objects in the view of actual obstacle, uniform distribution at the maximum range of laser beam and a uniform distribution throughout the range to include all unexpected measurements. The estimated value of the true measurement can be achieved through ray casting operations in case of beam based models or nearest neighbour function in case of likelihood fields model. Algorithms suggested in the book **Thrun98aprobabilistic** can be used off-the-shelf for estimating the ditribution.

3.3.2 Scan Matching

With advancement in computing power and use of LiDAR for perception it is possible to obtain more reliable results by registering two scans taken at two different relatively close poses and extract the relative poses between the two scans. This process of matching two scans is termed Scan Matching and it has been widely used for various applications. In other words, it is the method of finding the correct pose of the robot within a 3 dimensional search space which could provide the maximum value for the density $p(z_t|x_t, m)$.

$$\hat{x}_t = \operatorname{argmax}_x p(x|m_{t-1}, z_t, x_t) \quad (3.7)$$

Scan Matching is the core of this thesis. It will be discussed elaborately in subsequent chapters.

3.4 Summary

Chapter 4

Particle Filter

Introduction

Parametric filters such as EKF work on the assumption that posterior density is Gaussian parameterized by mean and covariance. In cases where the true density is non Gaussian, these filters fail to describe the actual posterior. In many practical applications, linear and gaussian assumptions would not hold true, hence better approaches are required to describe the posterior density. Particle Filters are non parametric filters which can model any posterior distribution as it does not assume gaussian requiring mean and covariance. This chapter provides in-depth view of Particle Filters and its application to SLAM.

4.1 Particle Filter

Particle Filters manage to model the required distribution by use of set of random samples sampled from the distribution. However no a-prior information about the distribution is known. Hence a proposal distribution is used as a initial guess to sample particles from it and weigh it against the target distribution. This process is repeated until the proposal distribution matches the target distribution and it requires resampling from the proposal distribution when necessary. The samples which weigh more are more likely to describe the target distribution. This leads to further discussion on what could be the guess on proposal distribution, how is it sampled, weighed, resampled and sequentially continued to obtain the full state estimate.

4.1.1 Sample

Sampling in particle filters is based on Sequential Importance Sampling principle and Sequential Monte-Carlo methods. Samples in general are generated in random from a proposal distribution. The more samples we are able to sample, the better is the approximation of the target distribution. With larger number of samples, Monte-Carlo methods provide equivalent representation of the target distribution and Sequential Importance Sampling approaches optimal Bayes estimate **S.Arulampalam**. In case of the Monte-Carlo approximation methods, the samples obtained are independent samples of the target distribution, whereas in the Importance Sampling method, the samples obtained are not only independent of the target distribution but independent on the proposal distribution as well. Thus the weights of each sample are not equal as in the case of Monte-Carlo approximation methods.

Let x^1, x^2, \dots, x^N be list of N particles generated from the proposal distribution $q(x)$.

$$p(x) \approx \sum_{i=1}^N w^i \delta(x - x^i) \quad (4.1)$$

where δ is the impulse function. The above equation provides the weighted approximation of the true density given that the proposal distribution is similar to the target distribution and it is easy to sample from the proposal distribution. A typical example of a proposal distribution is gaussian. However modeling the proposal distribution effectively for the task is discussed in later sections.

4.1.2 Importance weighting

Importance weighting is a useful outcome of Importance sampling. Each particle is weighed based on its similarity to the target distribution. More the similarity higher is the value of the weights.

$$w \propto \frac{p(x)}{q(x)} \quad (4.2)$$

The weights are always normalized for reasons discussed further below.

4.1.3 Sequential Importance Sampling

Sampling and weighting discussed above are useful to estimate the state for a given instance. However in most applications it is required to estimate the state recursively for every time instance using the previous state and the current reading. This is the core of the Sequential Importance Sampling. It involves sampling, weighting recursively. In the sampling step as the the sample from the motion model estimate is sufficient as the prior information untill the previous instance is already processed. For the i^{th} particle:

$$x_{0:t}^i \approx q(x_{0:t}|y_{1:t}) \quad (4.3)$$

where $q(x_{0:t}|y_{1:t})$ is the posterior density of the proposal distribtuion which could be factorized as

$$q(x_{0:t}|y_{1:t}) = q(x_t|x_{t-1}, y_t)q(x_{0:t-1}|y_{1:t-1}) \quad (4.4)$$

In the weighting step the proposal is compared to the target density also considering its prior weight update.

$$w_t^i = w_{t-1}^i \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t|x_{t-1}, y_t)} \quad (4.5)$$

With the right choice of the proposal distribtuion the weights can be derived with measurement model.

$$q(x_t|x_{t-1}, y_t) = p(x_t|x_{t-1}) \quad (4.6)$$

$$w_t^i = w_{t-1}^i * p(y_t|x_t^i) \quad (4.7)$$

In most cases the weights of the particles in a sample set can end up with high variance, due to which the particle with the highest weight gets sampled repeatedly in the resampling step. This is referred to as Particle Degeneration. Hence the system's

belief is stuck to one particle leading to ignorance of uncertainty and consequitively to divergence of the filter from the true value.

4.1.4 Resample

As the variance of the weights increase the sample set is dominated only by very few particles with high weight which results in poor state estimate. Hence to avoid it is better to perform a resampling step where N particles are sampled from the current particle set using various strategies. Depending on the strategies used to resample various resampling techniques exists such as Systematic resampling, Random resampling, Selective resampling ... This replaces the old sample set with new set of samples with equal weights. However in most cases the clarity of tolerable variance of the weights for a particle set is not clear. It is found that Adaptive resampling provides the best criteria for resampling. This requires calculating the effective number of the particles N_{eff}

$$N_{eff} = \frac{1}{\sum_{i=1}^N w_i^2} \quad (4.8)$$

where w_i is normalized weight of the particles in the sample set and N is the number of particles. Resampling is performed only when the $N_{eff} < N/2$

4.2 Rao-Blackwellized Particle Filter

4.3 gMapping-Improved RBPF

4.4 Results and Analysis

4.5 Summary

Chapter 5

Scan Matching

Introduction

Unveiling the full potential of LiDAR and its robustness in creating a scan map of the environment, LiDAR Odometry and Mapping(LOAM) was presented in **ZhangS14**. In this approach only a subset of the point clouds is used to extract useful features such as planes, lines and edges which is later used to derive various metrics for the scan registration. In **D.Hahnel**, consistent Maps were generated with the help of 2D scan matching and also were able to detect and track people even with the help of sample based Joint Probabilistic Data association filter. Depending on how the point cloud data is modelled existing approaches can be classified into global and local scan matching. In global scan matching the point cloud is considered as a whole, but in local scan matching only segments of the point cloud are matched.

Some of the local scan matching approaches include Iterative Closest Point(ICP), Normal Distribution Transform(NDT) and Global scan matching approaches include Correlative Scan Matching(CSM)

5.1 Iterative Closest Point

It is the most well-known widely used algorithm known for efficient registration of given two 2D or 3D scan or point cloud data under euclidean transformation. It is the most simple and intuitive algorithm. Provided the correspondence between the points in the point cloud and with a approximate initial guess, ICP tries to find the

optimal motion parameters - rotation \mathcal{R} and translation \mathcal{T} of given two point cloud data. In practice the correspondence is usually unknown, hence iterative procedure is followed with good initial guess. If the initial guesses are close enough, a converging solution is obtained. The iteration is generally stopped after certain number of iterations or if the error value reaches below a threshold. Different variants of ICP are developed such as Point-to-Point, Point-to-Plane, Plane-to-Plane each with its pros and cons.

ICP using Spectral Value Decomposition First proposed in **KS.Arun**, the centroid of the two point cloud data are aligned and relative rotation of the two point cloud data is obtained using SVD such that the least squared error is minimized. In simple words the squared error in the observation from two different relatively close view points need to be minimized. This approach results in closed form solution only when the point correspondence of two point clouds is known. Hence it is iterated over until convergence.

Given the two point cloud data p_1 and p_2 , where $p_2 = \mathcal{R}p_1 + \mathcal{T} + N$, find \mathcal{R} and \mathcal{T} such that error function

$$E(\mathcal{R}, \mathcal{T}) = \frac{1}{N_{p_1}} \sum_{i=1}^{N_{p_1}} \|p_2 - \mathcal{R}p_1 - \mathcal{T}\|^2 \quad (5.1)$$

is minimized. where N_{p_1} is the number of points in the point cloud data.

It can be accomplished by taking the centroids of the point clouds p_1 and p_2 such that

$$\mu_{p_1} = \frac{1}{N_{p_1}} \sum_{i=1}^{N_{p_1}} p_{1_i}$$

$$\mu_{p_2} = \frac{1}{N_{p_2}} \sum_{i=1}^{N_{p_2}} p_{2_i}$$

$$P_1 = p_1 - \mu_{p_1}$$

$$P_2 = p_2 - \mu_{p_2}$$

$$H \triangleq \sum_{i=1}^N P_1 P_2^T$$

(5.2)

Now to find the rotational component of the motion parameters \mathcal{R} , Spectral Value Decomposition can be applied on the cross covariance matrix H .

When the $\text{rank}(H) = 3$, then a unique and optimal solution for $E(\mathcal{R}, \mathcal{T})$ is obtained. The rotational component \mathcal{R} can be obtained by $\mathcal{R} = \mathcal{U}\mathcal{V}^T$

The translation component \mathcal{T} can be obtained by $\mathcal{T} = \mu_{p_1} - \mathcal{R}\mu_{p_2}$

The error encountered in the process is given by $E(\mathcal{R}, \mathcal{T}) = \sum_{i=1}^{N_{p_2}} (\|x'_i\|^2 + \|y'_i\|^2) - 2^*$ Singular values of H .

The approach presented above is applicable to point to point alignment and it could be modified and used for other variants of ICP. Being an iterative procedure and non-convex problem, ICP is susceptible to settle at local minima. This might result in non optimal solution. Methods such as GO-ICP **Yang_2016** were developed for global optimal solution. It is based on Branch-and-Bound theory for global optimization and the local-ICP procedure. It constitutes an outer loop of BnB search in the rotation space of $SO(3)$ and then solves for optimal search in inner loop for translational space. The search for the optimal parameters then stops on reaching a so-far-the best error threshold or the search is terminated when the search cubes are sufficiently small.

ICP using Gauss-Newton (Least squares approach) However this put restrictions on the choice of error function to be used, hence a more generic approach to find solution to the problem is to take least squares using Gauss-Newton method. This method also tries to minimize the error but the error function can be chosen appropriately. The error function is then linearized using Taylor series expansion and by forming the hessian matrix and jacobian vector the desired parameters can be obtained iteratively. Given the two point cloud data p_1 and p_2 , where

$p_2 = \mathcal{R}p_1 + \mathcal{T} + N$, the error function defines how varied are the points. Let the error function be $E(\mathcal{R}, \mathcal{T}) = \sum_{i=1}^{N_{p_1}} \|p_2 - p_1\|^2$ assuming the correspondences are known.

Linearizing the above equation $E(\mathcal{R} + \Delta\mathcal{R}, \mathcal{T} + \Delta\mathcal{T}) = E(\mathcal{R}, \mathcal{T}) + \mathcal{J}_E(\mathcal{R}, \mathcal{T})(\Delta\mathcal{R}, \Delta\mathcal{T})$ where, $\mathcal{J}_E(\mathcal{R}, \mathcal{T}) = \partial(E)/\partial(\mathcal{R}, \mathcal{T})$

Solving the above error equation by Gauss-Newton approach, the change in the parameters is derived as a small step towards reaching the global minimum of the error function. $\Delta(X) = -\mathcal{H}^{-1}b$ where, $\mathcal{H} = \mathcal{J}_E^T \mathcal{J}_E$ $b = \mathcal{J}_E^T E$

Since the Gauss-Newton approach takes only in steps to reach the global minima, the procedure is repeated in iterations.

ICP with Point-to-Plane correspondence The point to point correspondence assumed in the above methods can be achieved using Nearest Neighbour algorithms. However this strategy might lead to wrong correspondences as it only considers the points as discrete whereas in reality these points actually represent the hidden structure in the surrounding. Thus considering the surfaces of the obstacle would provide better correspondences, this results in Point-to-Plane matching.

In the Point-to-Plane correspondence still the closest points are considered as initial guess to define the surface. Then the point to point projection of the error vector is projected on to the surface of the normal of the target scan. This process is repeated until the projection of error vector on the normal is minimal. The Point-to-Plane correspondence can be used along with Gauss-Newton approach to get better results. Further the robustness of the ICP algorithms can be enhanced by rejecting the outliers in the point cloud data such as dynamic obstacles and reflections.

5.2 Real-Time Correlative Scan Matching

Perhaps the most efficient, intuitive and robust method commonly used in almost all modern day robot for scan matching is the Real-Time Correlative Scan Matching proposed by E.Olson in **E.B.Olson** inspired from **Konolige** correlation based

localization. It provides a novel multi-resolution approaches to compute the motion parameters in conventional CPUs and modern day GPUs. It is based upon cross correlation for two lidar scans through probabilistic approach. It finds the rigid body transformation that maximizes the probability of having observed a scan instead of using a local search algorithm to find global maximum. The efficient computation of the density $p(z_t|x_t, m)$ is made possible by implementing a multi-level resolution of the rasterized cost map. First a low-resolution cost map is used to find the area of the global maximum with the position of the robot established predicted by the motion model. This ensures that search volume in high resolution is reduced. With the region of global maximum observed, the search is repeated with the high-resolution cost map. This method proves to be exceptionally robust and it is been widely used in the industry and many open source SLAM implementations like cartographer. This method has an advantage that it not only computes the motion parameters, but it tries to compute the entire density $p(z_t|x_t, m)$, hence uncertainty of the estimated motion parameters is also calculated. Based on **E.Olson**, a similar multi-resolution approach is proposed with increased accuracy and better quality in **P.Vath**. In this the author uses not only the occupied cells but also the unoccupied cells for scoring the scan match based on the choice of pose in the 3D search space. The score function adds up positively in case of matching occupied cells and negatively for mismatching unoccupied cells.

5.3 Normal Distribution Transform

P.Biber proposed a alternative method for matching lidar scans using normal distributions to cells in a local map that can be used to match another normal distributed cells in a map using Newton's optimization algorithm. The Occupancy grid created around the robot is discretized into cells of appropriate size and a normal distribution is formed for the cells which contain more than 3 detections in it. Now a 2D plane in the form of probability distribution is achieved. In order to minimize discretization effects, maps which are shifted by all 4 directions are overlapped to get a continuous

distribution. In order to match two given scans, the NDT of the first scan is created and with the predicted motion parameters, the second scan points are mapped into the coordinate frame of first scan and the normal distribution is constructed for each matched point. The score for the parameters is determined by evaluating the distribution and summing the result. This is then optimized using Newton's equation and repeated until the convergence is met. Since only the distribution of the detections is considered, It is proven to be faster than ICP. The algorithm is proven to work efficient for SLAM and position tracking. Similar approach was taken by **K.Ryu** in which new scans were matched to previous scan by ICP but are later corrects the error by matching to globally defined map. The global map is constructed as a Normal Distribution and the new scan is matched to the NDs for correcting the errors. Thus it performs a ND-to-ND matching using Kullback-Leibler (KL) divergence. The author claims that this methods has the benefits of both ICP and NDT

HaoFU evaluated popular scan matching approaches using a LiDAR dataset recorded in off-road environment and proposed an alternative approach combining local and global scan matching to obtain the best of both.

5.4 Loop Closure

In several cases the robot might traverse through a previously visited location and it must be able to recognize it and it is termed as *loop closing*. This enables to achieve better accuracy of the Map. Apart from Map building the ability of robots to recognize the previously visited place or learned location enables the bot to fall-back to the specific location in use cases such as trouble shooting, charging dock and determining destination. The problem of Loop closure is challenging as in the real world the algorithm must be able to achieve good results even in dynamic environment. Also the robot must be capable of identifying similar looking environment. This method of relating the current measurement with a previous measurement is a process of *data association*. Errors are common in *data association* and it leads to divergence of the map when the errors in *data association* are not handled care-

fully. With the presence of landmarks the covariance information of the landmarks is required. **E.Olson/LocalSM** proposed methods to determine whether a local scan match is globally correct. It also incorporates ambiguity and outlier testing using *Single Cluster Graph Partitioning (SCGP)*. It also argues that the amount of evidence to determine the similarity between two places scales with robots positional uncertainty. Find Local matches within a search area provided by a prior, then combine multiple scans to get a larger local matches. In order to estimate the relative positional uncertainty between nodes a and b, the determinant of the covariance matrix provides the search space to find the pose b from pose a. Using Dijkstra projection algorithm, the uncertainty is estimated and the relative uncertainty between two paths is dominated by the shortest one.

Grouping pairwise consistency local uniqueness and outlier rejection global sufficiency

Apart from SCGP methods such as Combined Constraint Data Association, Joint Compatibility Branch and Bound and Cyclic verification of cumulative transformations are also available to remove false positive loop closures. However the performance of these methods depends on good initialization **P.Agarwal**.

Chapter 6

Comaprison and Benchmarking

Chapter 7

Conclusions

Chapter 8

Reflective Analysis

Appendix A

Some random python code

This template includes the `minted` package, which allows you to import code and syntax highlight it. For example, the text below is imported directly from the `code/test.py` file using the `\inputminted` command:

```
python3code/test.py
```

And here is a snippet of Python with the `minted` environment:

```
[breaklines]python Why don't you try running this? See what it does? hm?
m = [ 2, 3, 0, 1, 4 ]; x = [ 'rmdq', 'd', 'n'slk', 'odftp'v', 'hdk' ];
c = ".join(list(map(lambda y: chr(ord(y) ^ 5).upper() + ' ify! ' =
'else'', '.join([x[m[i]]for i, vin enumerate(x)]))));
print('
```

Minted supports many, many languages – so you’re not just limited to Python. For example, here’s some random C++ code.

```
[breaklines]cpp void CTimesTable::Print(const int number, const int upTo) const  
for(int i = 1; i <= upTo; i++) printf(")
```

Appendix B

Some other stuff

Here is just an example of some other stuff.