MONTE CARLO LOCALIZATION

SIDDHANT SEKHAR, 18395

SARTHAK MISHRA, 18388

1 Abstract

Reliable position estimation is a key problem in mobile robotics. In This Project we tried to implement Monte Carlo Localization method, which is an algorithm for robots to localize using a particle filter where we represent the probability density involved by maintaining a set of samples that are randomly drawn from it .By using a sampling-based representation we obtain a localization method that can represent arbitrary distributions.Given a map of the environment, the algorithm estimates the position and orientation of a robot as it moves and senses the environment.

2 | Introduction

Two key problems in mobile robotics are global position estimation and local position tracking. We define global position estimation as the ability to determine the robot's position in an a priori or previously learned map, given no other information than that the robot is somewhere on the map. In this project, we present the Monte Carlo Localization method (MCL method) where we take a different approach to representing uncertainty: instead of describing the probability density function itself, we represent it by maintaining a set of samples that are randomly drawn from it. To update this density representation over time, we make use of Monte Carlo methods that were invented in the seventies [6], and recently rediscovered independently in the target-tracking [7], statistical [8] and computer vision literature.

3 | Monte Carlo Method

Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle. They are often used in physical and mathematical problems and are most useful when it is difficult or impossible to use other approaches. Monte Carlo methods are mainly used in three problem classes:[1] optimization, numerical integration, and generating draws from a probability distribution. In principle, Monte Carlo methods can be used to solve any problem having a probabilistic interpretation. By the law of large numbers, integrals described by the expected value of some random variable can be approximated by taking the empirical mean (a.k.a. the sample mean) of independent samples of the variable. When the probability distribution of the variable is parametrized, mathematicians often use a Markov chain Monte Carlo (MCMC) sampler.[3][4][5] The central idea is to design a judicious Markov chain model with a prescribed stationary probability distribution. That is, in the limit, the samples being generated by the MCMC method will be samples from the desired (target) distribution.[6][7] By the ergodic theorem, the stationary distribution is approximated by the empirical measures of the random states of the MCMC sample.

4 | Robot Localization

In robot localization, we are interested in estimating the state of the robot at the current time-step k, given knowledge about the initial state and all measurements

$$Z^k = (z_k, i = 1, ..., k)$$

up to the current time. Typically, we will work with a three-dimensional state vector $x = [x; y; Q]^T$,i.e. the position and orientation of the robot. This estimation problem is an instance of the Bayesian filtering problem, where we are interested in constructing the posterior density $p(x_k|Z^k)$ of the current state conditioned on all measurements.In the Bayesian approach, this probability density function (PDF) is taken to represent all the knowledge we possess about the state x_k, and from it we can estimate the current position. Summarizing, to localize the robot we need to recursively compute the density $p(x_k|Z_k)$ at each time-step.

5 | Method

In sampling-based methods one represents the density $p(x_k \mid Z^k)$ by a set of N random samples or particles $S_k = (s^i_k ; i = 1,...,N)$ drawn from it. We are able to do this because of the essential duality between the samples and the density from which they are generated [17]. From the samples we can always approximately reconstruct the density,e.g. using a histogram or a kernel based density estimation technique.The goal is then to recursively compute at each timestep k the set of samples Sk that is drawn from $p(x_k \mid Z^k)$.A particularly elegant algorithm to accomplish this has recently been suggested independently by various authors. Itis known alternatively as the bootstrap filter [7], the MonteCarlo filter [8] or the Condensation algorithm [9, 10].These methods are generically known as particle filters.The algorithm proceeds in two phases:(i) Prediction Phase (ii)Update Phase

5.1 Particle Filter

Particle filters, or sequential Monte Carlo methods, are a set of Monte Carlo algorithms used to solve filtering problems arising in signal processing and Bayesian statistical inference. The filtering problem consists of estimating the internal states in dynamical systems when partial observations are made, and random perturbations are present in the sensors as well as in the dynamical system. The objective is to compute the posterior distributions of the states of some Markov process, given some noisy and partial observations. Particle filtering uses a set of particles (also called samples) to represent the posterior distribution of some stochastic process given noisy and/or partial

observations. The state-space model can be nonlinear and the initial state and noise distributions can take any form required. Particle filter techniques provide a well-established methodology[1][3][4] for generating samples from the required distribution without requiring assumptions about the state-space model or the state distributions. Particle filters update their prediction in an approximate(statistical) manner. The samples from the distribution are represented by a set of particles; each particle has a likelihood weight assigned to it that represents the probability of that particle being sampled from the probability density function. Weight disparity leading to weight collapse is a common issue encountered in these filtering algorithms; however it can be mitigated by including a resampling step before the weights become too uneven. Several adaptive resampling criteria can be used, including the variance of the weights and the relative entropy with respect to the uniform distribution.[5] In the resampling step, the particles with negligible weights are replaced by new particles in the proximity of the particles with higher weights.

5.2 Prediction Phase

e In the first phase we start from the set of particles S_{k-1} computed in the previous iteration, and apply the motion model to each particle s^i_{k-1} by sampling from the density $p(x_k \mid s^i_{k-1}, u_{k-1})$

(i) For each particle s^i_{k-1}:draw one sample s'^i_{k-1} from $p(x_k \mid s^i_{k-1}, u_{k-1})$

In doing so a new set S'_k is obtained that approximates a random sample from the predictive density $p(_k \mid Z^k)$

5.3 Update Phase

In the second phase we take into account the measurement z_k, and weight each of the samples in S'^k by the weight $m^i_k = p(z_k \mid s'^i_k)$, i.e. the likelihood of s'^i_k given z_k. We then obtain S_k by resampling from this weighted set

(ii) For j =1,...,N draw one S_k sample s^i_k from (s'^i_k, m^i_k)

The resampling selects with higher probability samples s'^{i}_{k} that have a high likelihood associated with them, and in doing so a new set $S_(k)$ is obtained that approximates a random sample from $p(x_k \mid Z^k)$

After the update phase, the steps (i) and (ii) are repeated recursively. To initialize the filter, we start at time k = 0 with a random sample $S_0 = (s_{i0})$ from the prior $p(x_0)$

6 Results and Discussions

The Monte Carlo Localization algorithm was run for a simple local environment grid of 100 x 100 px. A total of 50 steps were run to estimate the final state of the robot. The results of the localized position of the robot was obtained for each step. We have compiled it into a single video for better understanding.

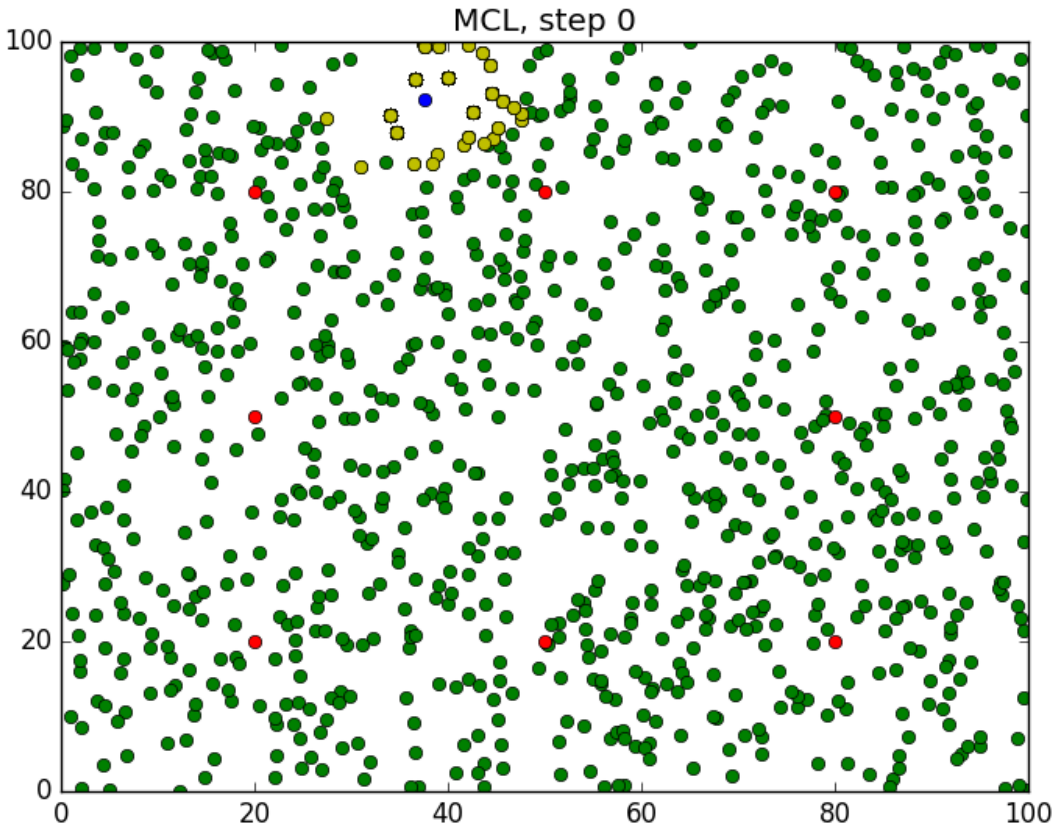

Figure 6.1: Initial State

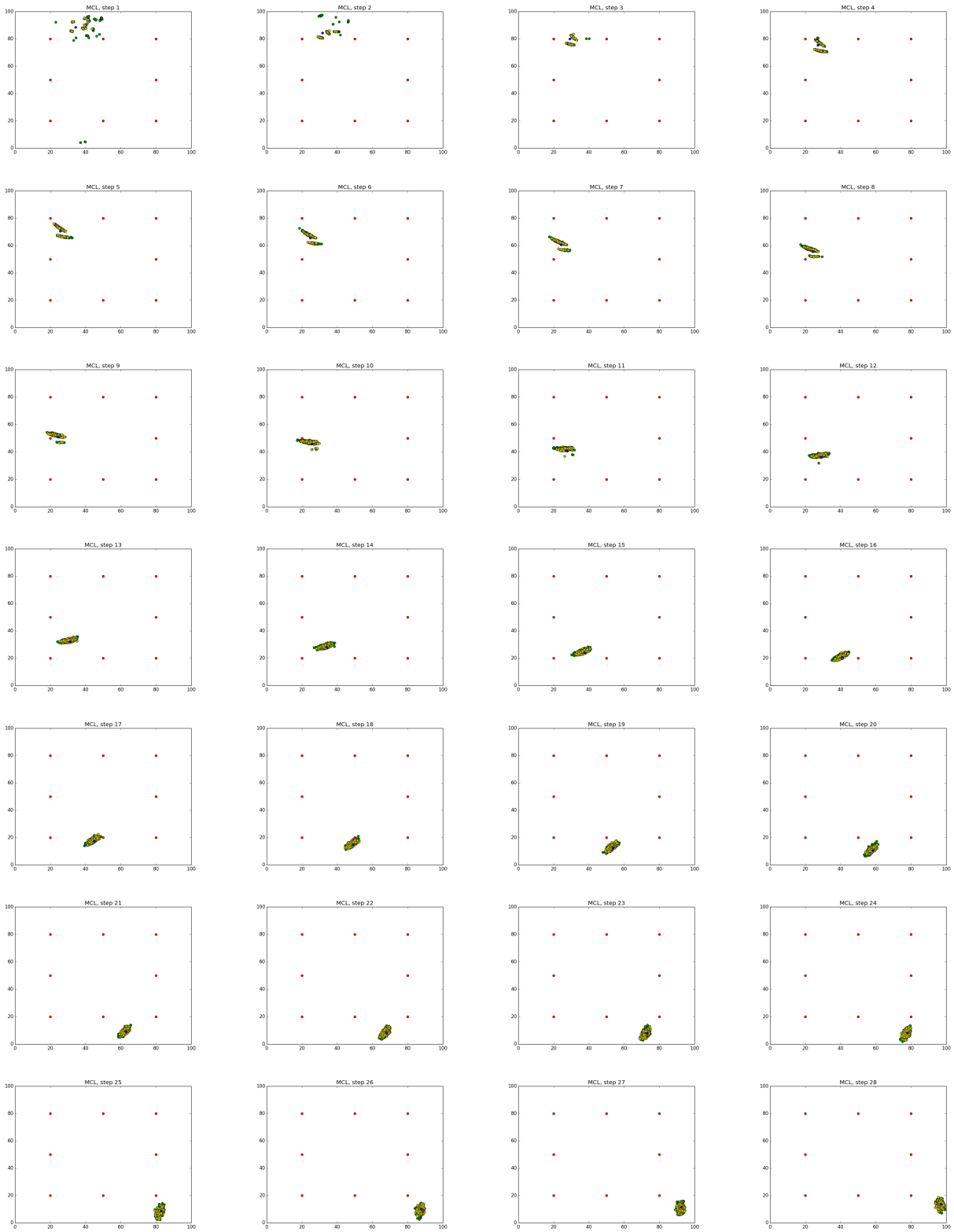

Figure 6.2: Results Output

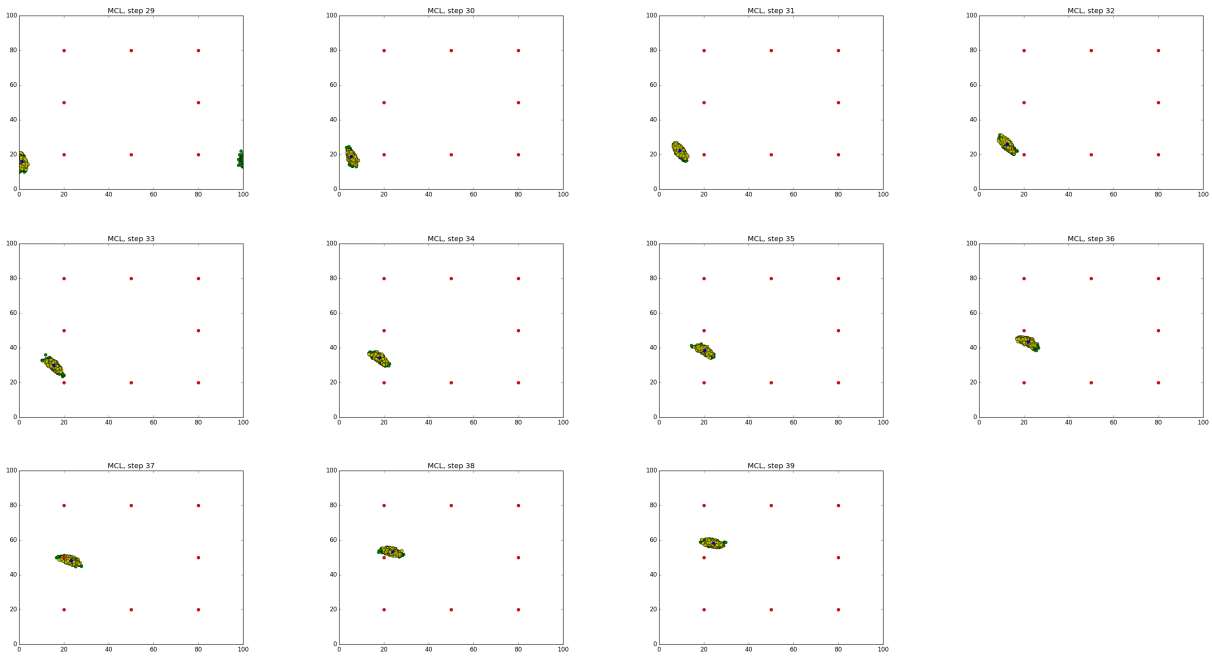

Figure 6.3: Results Output

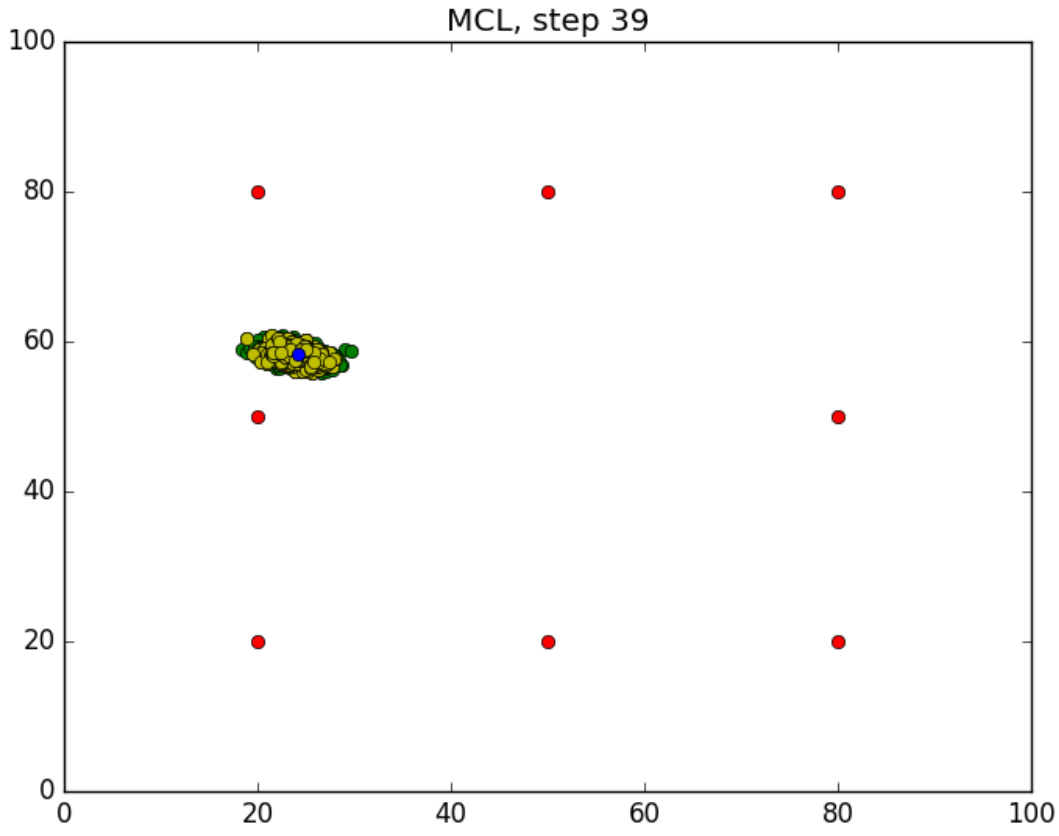

Figure 6.4: Final State

6.1 Limitations of MCL

The basic particle filter performs poorly if the proposal distribution, which is used to generate samples, places too little samples in regions where the desired posterior is large. MCL works best for 10% to 20% perpetual noise.MCL also performs poorly when the noise level is too small. In other words, MCL with accurate sensors may perform worse than MCL with inaccurate sensors.

7 | Conclusion

There were initially 100 particles randomly distributed in the grid. After each step of resampling, new positions of each particle was calculated as the Euclidean distance from the robot. This distance is minimized and thus after 50 steps it was found that the particles were localized around the robot and hence, it can be used for further calculation of the correct pose estimation of the robot.

Bibliography

1. Monte Carlo Localization for Mobile Robots

2. Monte Carlo Localization

3. Particle Filter

4. Particle Filter and Monte Carlo Localization Youtube Link

5. Robust Monte Carlo Localization for Mobile Robots