MIDDLE EAST TECHNICAL UNIVERSITY
Department of Electrical and Electronics Engineering

EE 583 ARTIFICIAL INTELLIGENCE
Spring 2006

PROJECT FINAL REPORT

Simultaneous Localization and Mapping Implementation on Khepera Robot Simulator

Bengi Koç , Barış Tanrıverdi , Salim Sırtkaya
(Group 5)

TABLE OF CONTENTS

LIST OF FIGURES

1. Outline of the Report

This document is the project final report of EE586–Artificial Intelligence course given in Spring 2006 at METU-EE by Afşar Saranlı. The project is about Simultaneous Localization and Mapping Implementation on Khepera Robot Simulator. The outline of the report is as follows,

In Section 2, the problem is stated formally with the selected solution approach.

In Section 3, the theoretical basis for the solution is given regardless of the implementation constraints.

In Section 4, the implementation details are given with the algorithm.

In Section 5, results of the implementation are presented with details and with some comments.

In Section 6, work allocation between the team members is given

In Section 7, references are given

2. Problem Definition

A critical ability of an autonomous robot is navigation. The functions of navigation can be expressed in four questions: where am I going, which is the best way to get there, where have I been, and where am I? In order to solve these questions, the robot needs to know its environment. As the robot explores new environments, it may be part of its mission to map them. The acquisition of unknown environments is known as robotic mapping [1].

In particular, Simultaneous Localization and Mapping (SLAM) can be described as the ability of a robot to build a map of its environment while localizing itself in that map to reach true autonomy [2]. SLAM problem tries to answer the following question: "Is it possible for an autonomous vehicle starting at an uncertain location in an unknown environment to incrementally build a map of the environment while simultaneously using this map to compute the vehicle's location?"[3]. This scenario is shown in Figure 1 [4].

A possible realization for this problem may be an autonomous robot which relies on GPS for the navigation purpose. As the robot navigates in an outdoor environment the GPS will be

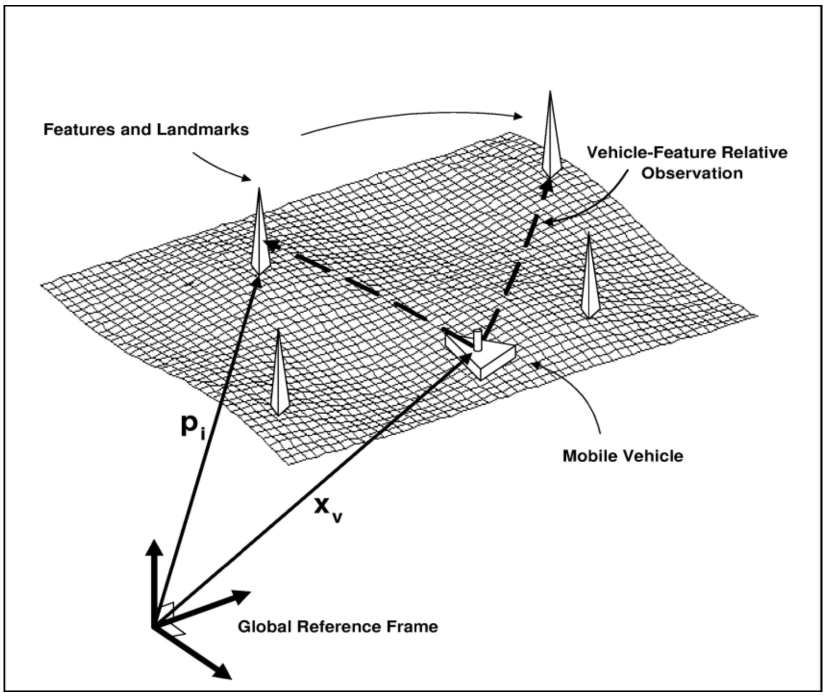

Figure 1 A vehicle taking relative measurements to environmental landmark for SLAM

enough for the navigation purpose since it's an external reference which has a bounded position error. However, if the robot enters to an indoor environment where the GPS satellites are not perceptible, it will loose this bounded error navigation mechanism. It may continue navigating using its odometer sensors. But, the error with odometer navigation will not be bounded anymore. This is because of the fact that, odometer is an internal sensor which gives velocity; therefore the error of its measurements will be integrated in the navigation process yielding an exponentially growing position error (dead-reckoning). This necessitates an external reference for robust navigation of such a robot. SLAM is a practical solution for this external reference necessity.

A number of approaches have been proposed to address both the SLAM problem and also more simplified navigation problems where additional map or vehicle location information is made available. After a deep literature survey, we chose the so called fastSLAM algorithm for the defined problem. fastSLAM is one of the most popular and recent solution to the SLAM problem. In the following section the theoretical basis for the fastSLAM algorithm is given.

The algorithm is tested on synthetic data (for proof of concept) and on Khepera Robot Simulator Kiks. Kiks seems to be a suitable test bed for current problem, since it's an indoor robot which has proximity sensors, shaft encoders etc. However, as we'll analyze in Chapter 4 it has some problems concerning the type of measurements it's giving. To accommodate Kiks to SLAM problem, it is required to post-process the sensor data.

3. Basic Theory of fastSLAM

The FastSLAM algorithm is a new solution to stochastic SLAM that is not based on the Kalman filter but, instead, uses a particle filter to approximate the ideal recursive Bayesian filter. More precisely, it involves a partitioned state-space whereby the robot pose states are represented by particles and the landmark states are estimated analytically by Kalman filters. fastSLAM decomposes the SLAM problem into a robot localization problem, and a collection of landmark estimation problems that are conditioned on the robot pose estimate.[7] fastSLAM makes use of the observation that the landmark estimates are conditionally independent given the robot pose. Basically, the algorithm is a particle filter, in which every particle represents a sampled robot trajectory plus a set of Kalman filters estimating the position for each landmark as a Gaussian distribution. These distributions are independent, so small covariance matrices are needed, instead of one large matrix. The computation time for integrating a measurement is for particles with storage space. This algorithm can cope with uncertain landmark identification, which is a unique advantage over the other algorithms. [5]

3.1. Bayesian Approach to SLAM

Consider the case of a mobile robot moving through an unknown environment. The environment consists of a set of stationary landmarks. The position of landmark $\boldsymbol{k}$ with respect to some global reference frame is denoted $\boldsymbol{\theta_k}$. The robot moves according to a known probabilistic motion model $\boldsymbol{p(s_t|u_t,s_{t-1})}$, where $\boldsymbol{s_t}$ and $\boldsymbol{u_t}$ denote the robot configuration and the control input at time $\boldsymbol{t}$, respectively. As the robot moves around, it takes measurements of the landmarks in its environment. A measurement $\boldsymbol{z_t}$, is related to the position of a landmark through a stochastic observation model $\boldsymbol{p(z_t|s_t,\theta,n_t)}$, where $\boldsymbol{\theta}$ is the set of all landmark positions and $\boldsymbol{n_t}$, is the index of the particular landmark observed at time $\boldsymbol{t}$.

The SLAM problem is that of simultaneously inferring the location of all landmarks in the environment and the path followed by the robot based on a set of measurements and inputs. Ideally, one would like to recover the posterior distribution $\boldsymbol{p(s^t, \theta_1, \ldots, \theta_K|z^t,u^t)}$, where $\boldsymbol{K}$ is the number of landmarks observed so far and the notation $\boldsymbol{s^t}$ denotes $\boldsymbol{s_1,\ldots,s_t}$ (and similarly for other variables). [6],[7]

3.2. General Motion and Measurement Models for SLAM

The SLAM problem was defined as the problem of recovering a map and a robot pose (location and orientation) from data acquired by a mobile robot. The robot gathers information about nearby landmarks, and it also measures its own motion. Both types of measurements are subject to noise. They are compiled into a probabilistic estimate of the map along with the robot's momentary pose (location and orientation).

To acquire a map, the robot can sense. Sensor measurements convey information about the range, distance, appearance etc. of nearby features. This is illustrated in Figure 2, in which a robot measures the range and bearing to a nearby landmark. For each measurement z_t, n_t specifies the identity of the observed feature.

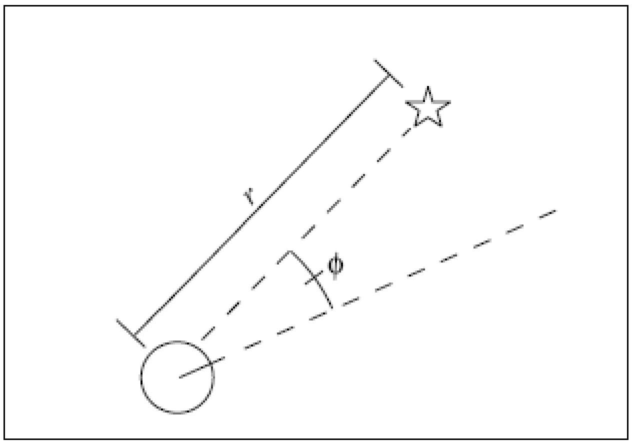

Figure 2 Vehicle observing the range and bearing to a nearby landmark.

The measurement model is given by the following probabilistic equation

$$p(z_t \mid s_t, \theta_{n_t}, n_t) \quad = \quad g(\theta_{n_t}, s_t) + \varepsilon_t$$

The measurement model is conditioned on the robot pose s_t, the landmark identity n_t, and the specific feature θ_{nt} that is being observed. It is governed by a (deterministic) function g distorted by noise. The noise at time t is modeled by the random variable ε_t, which will be assumed to be normally distributed with mean zero and covariance R_t.

A second source of information for solving SLAM problems are the controls of the vehicle. Controls are denoted u_t, and refer to the collective motor commands carried out in the time interval [t-1; t]. The probabilistic law governing the evolution of poses is commonly referred to as kinematic motion model, and will be assumed to be of the following form:

$$p(s_t \mid u_t, s_{t-1}) \quad = \quad h(u_t, s_{t-1}) + \delta_t$$

The goal of SLAM is the recovery of the map from sensor measurements z^t and controls u^t.

3.3. Contribution of Conditional Independence to Bayesian Approach, fastSLAM

The key observation underlying the fastSLAM approach, which makes it different from the traditional Bayesian approach, is that: given a known robot path, measurements corresponding to different landmarks are independent of each other (conditional independence). This is illustrated in Figure 3, where the SLAM problem is represented as a Bayesian network. Observe that measurements corresponding to the landmark locations θ_1 and θ_2 are only linked together through their joint dependence on the robot path s^t.

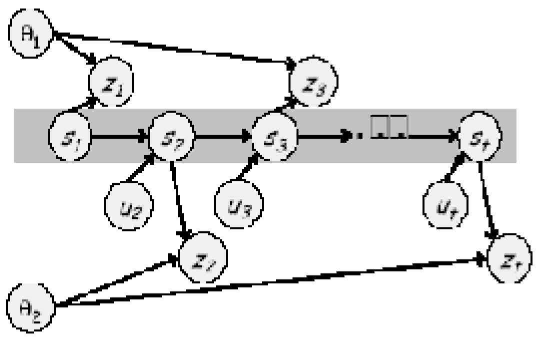

Figure 3 A Bayesian network depicting the structure of the SLAM problem

3.3.1. Factored Representation

For now, assume that we know the identity of each landmark, namely the correspondence, n^t. Hence we'll recognize a landmark if we observe it more than once. We'll propose an algorithm later, for the correspondence problem.

Using the conditional independence on the Bayesian Network given in Figure 3 to factor the desired posterior distribution yields [6]

$$p(s^t, \theta_1, \ldots, \theta_K | z^t, u^t, n^t) = p(s^t | z^t, u^t, n^t) \prod_{k=1}^{K} p(\theta_k | s^t, z^t, u^t, n^t).$$

The problem can be decomposed into ***K+1*** estimation problems one problem of estimating a posterior over robot paths $\mathbf{s}^t$, and K problems of estimating the locations of the K landmarks conditioned on the path estimate.

In fastSLAM, a particle filter is used to estimate the path of the robot by maintaining an estimate of its current configuration [9]. This filter can sample efficiently from the probabilistic state space of the motion model, providing a good approximation of the posterior even under non-linear motion kinematics. Each particle maintains its own estimate of the landmark locations. Because these are conditionally independent of each other given robot path, they may each be maintained by a low-dimensional EKF. Thus, for $\boldsymbol{M}$ particles and $\boldsymbol{K}$ landmarks, there will be a total of $\boldsymbol{M*K}$ Kalman filters, each of dimension 2 (for the two landmark coordinates) [7].

3.3.2. Particle Filter Path Estimation

fastSLAM employs a particle filter for estimating the path posterior $\boldsymbol{p(s^t|z^t,u^t,n^t)}$ [7]. At each point in time, the algorithm maintain a set of particles representing the posterior $\boldsymbol{p(s^t|z^t,u^t,n^t)}$ denoted by $\boldsymbol{S_t}$. Each particle $\boldsymbol{s^{t,[m]}}$ Є St represents a "guess" of the robot's path:

$$\boldsymbol{S_t} = \{\boldsymbol{s^{t,[m]}}\}_{\mathrm{m}} = \{\boldsymbol{s_1^{[m]}}, \boldsymbol{s_2^{[m]}},\ldots, \boldsymbol{s_t^{[m]}}\}_{\mathrm{m}}$$

We use the superscript notation $^{[m]}$ to refer to the m-th particle in the set. The particle set $\boldsymbol{S_t}$ is calculated incrementally, from the set $\boldsymbol{S_{t-1}}$ at time $\boldsymbol{t-1}$, a robot control $\boldsymbol{u_t}$ and a measurement $\boldsymbol{z_t}$. First, each particle $\boldsymbol{s^{t,[m]}}$ in $\boldsymbol{S_t}$ is used to generate a probabilistic guess of the robot's pose at time $\boldsymbol{t}$:

$$\boldsymbol{s^{t,[m]}} \sim \boldsymbol{p(s_t|u_t\, s_{t-1}^{[m]})}$$

obtained by sampling from the probabilistic motion model. This estimate is then added to a temporary set of particles, along with the path $\boldsymbol{s^{t-1,[m]}}$. Under the assumption that the set of particles in $\boldsymbol{S_{t-1}}$ is distributed according to $\boldsymbol{p(s^{t-1}|z^{t-1},u^{t-1},n^{t-1})}$ (which is an asymptotically correct approximation), the new particle is distributed according to ' $\boldsymbol{p(s^t|z^{t-1},u^t,n^{t-1})}$. This distribution is commonly referred as proposal distribution.

After generating M particles in this way, the new set St is obtained by sampling from the temporary particle set. Each particle is drawn (with replacement) with a probability proportional to a so-called importance factor which is calculated as follows:
$\boldsymbol{w_t^{[m]}}$ = ***target distribution / proposal distribution***

The resulting sample set $\boldsymbol{S_t}$ is distributed according to an approximation to the desired pose posterior

3.3.3. Landmark Location Estimation (Extended Kalman Filter)

fastSLAM represents the conditional landmark estimates $\boldsymbol{p(\theta_k| s^t,z^t,u^t,n^t)}$ by Extended Kalman filters. Since this estimate is conditioned on the robot pose, the Kalman filters are attached to individual pose particles in St.

The posterior over the k-th landmark pose is depending on whether the landmark is seen before or not. If it's a new landmark, its pose is initialized according to simple trigonometric equations. If it's a previously seen landmark, its pose is updated according to classical Kalman filter update equations. Extended Kalman filter is an extension where the nonlinear motion and measurement models are linearized by expansion rules.

To sum up, based on the control input $\boldsymbol{u_t}$, a measurement $\boldsymbol{z_t}$ and its data association $\boldsymbol{n_t}$, a fastSLAM update is performed as follows. First, each particle is updated individually. For a particle $\boldsymbol{m}$, this involves updating the local pose estimate $\boldsymbol{s_t^{[m]}}$ by sampling the motion model. The measurement $\boldsymbol{z_t}$ and data association $\boldsymbol{n_t}$ are then used to update the appropriate local landmark location estimate $\boldsymbol{\theta_{nt}^{[m]}}$ using the standard EKF equations. Each particle is assigned an importance weight proportional to the likelihood of the measurement given the estimated robot pose. This quantity is computed in the prediction stage of the EKF. Once all particles have been updated, a new set of particles is generated through importance sampling, which involves drawing samples from the old particle set with a probability proportional to their importance weights.

4. Algorithmic Details

4.1. Outline of the fastSLAM Algorithm (a pseudo-code)

At the heart of the fastSLAM implementation lays the particles. Figure 4 illustrates the structure of the particles in fastSLAM. Each particle consists of a pose estimate, mean, variance and importance factor for each landmark. We added weight for each particle indicating its importance among the particle set.

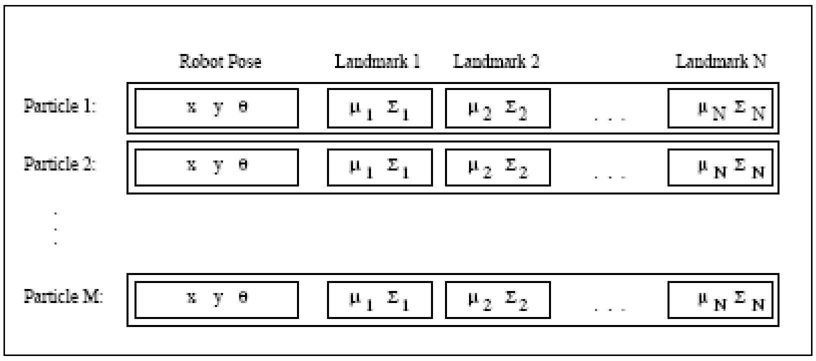

Figure 4 Particles in fastSLAM

There are totally M particles forming a set for a specific time. We update the particle set each time the robot gets a measurement and/or it moves. Actually this is a filtering process which generates a new particle set from the previous one. This filtering is conducted in three steps

- **Extending the path posterior by sampling new poses**: Update the pose of the robot using your motion model. The inputs to the model are the input given to the robot and the previous pose.
- **Updating the observed landmark estimate:** Use Extended Kalman Filtering for updating the observed landmark estimates. The theoretical issues are discussed in Chapter 3.
- **Resampling:** Resample the filtered particle set to achieve the posterior distribution. This step is necessary since we gave no attention on measurements while updating the pose of the robot. Resampling is a common technique in particle filtering to correct for such mismatches. In the implementation, resampling is achieved using the weights of the particles which are calculated based on the importance factor calculation. [6]

The algorithm is summarized in the following pseudo-code. Note that the correspondence estimation is conducted at the very beginning of the algorithm.

```
**Algorithm FastSLAM 1.0**$(z_t, u_t, S_{t-1})$:

  for $m = 1$ to $M$ do                                                   // loop over all particles
    retrieve $\left\langle s_{t-1}^{[m]}, N_{t-1}^{[m]}, \left\langle \mu_{1,t-1}^{[m]}, \Sigma_{1,t-1}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_{t-1}^{[m]},t-1}^{[m]}, \Sigma_{N_{t-1}^{[m]},t-1}^{[m]}, i_{N_{t-1}^{[m]}}^{[m]} \right\rangle \right\rangle$ from $S_{t-1}$
    $s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$                          // sample new pose
    for $n = 1$ to $N_{t-1}^{[m]}$ do                                         // calculate measurement likelihoods
      $\hat{z}_n = g(\mu_{n,t-1}^{[m]}, s_t^{[m]})$                             // measurement prediction
      $G_n = g'(s_t^{[m]}, \mu_{n,t-1}^{[m]})$                                  // calculate Jacobian
      $Q_n = G_n^T \Sigma_{n,t-1}^{[m]} G_n + R_t$                              // measurement covariance
      $w_n = |2\pi Q_n|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q_n^{-1}(z_t - \hat{z}_n)\right\}$   // likelihood of correspondence
    endfor
    $w_{N_{t-1}^{[m]}+1} = p_0$                                               // importance factor of new landmark
    $\hat{n} = \underset{n=1,\ldots,N_{t-1}^{[m]}+1}{\operatorname{argmax}} \; w_n$          // max likelihood correspondence
    $N_t^{[m]} = \max\{N_{t-1}^{[m]}, \hat{n}\}$                              // new number of features in map
    for $n = 0$ to $N_t^{[m]}$ do                                             // update Kalman fi lters
      if $n = N_{t-1}^{[m]} + 1$ then                                         // is new feature?
        $\mu_{n,t}^{[m]} = g^{-1}(z_t, s_t^{[m]})$                              // initialize mean
        $\Sigma_{n,t}^{[m]} = G_{\hat{n}}^{-1} R_t (G_{\hat{n}}^{-1})^T$        // initialize covariance
        $i_{n,t}^{[m]} = 1$                                                   // initialize counter
      else if $n = \hat{n}$ then                                              // is observed feature?
        $K = \Sigma_{n,t-1}^{[m]} G_{\hat{n}} Q_{\hat{n}}^{-1}$                 // calculate Kalman gain
        $\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]} + K(z_t - \hat{z}_{\hat{n}})^T$    // update mean
        $\Sigma_{n,t}^{[m]} = (I - K G_{\hat{n}}^T)\Sigma_{n,t-1}^{[m]}$        // update covariance
        $i_{n,t}^{[m]} = i_{n,t-1}^{[m]} + 1$                                 // increment counter
      else                                                                  // all other features
        $\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]}$                                 // copy old mean
        $\Sigma_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]}$                           // copy old covariance
        if $\mu_{n,t-1}^{[m]}$ outside perceptual range of $s_t^{[m]}$ then     // should feature have been observed?
          $i_{n,t}^{[m]} = i_{n,t-1}^{[m]}$                                   // no, do not change
        else
          $i_{n,t}^{[m]} = i_{n,t-1}^{[m]} - 1$                               // yes, decrement counter
          if $i_{n,t-1}^{[m]} < 0$ then discard feature $n$ endif              // discard dubious features
        endif
      endif
    endfor
    add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ to $S_{\text{aux}}$
  endfor
  $S_t = \emptyset$                                                         // construct new particle set
  for $m' = 1$ to $M$ do                                                    // resample M particles
    draw random index $m$ with probability $\propto w_t^{[m]}$               // resample
    add $\left\langle s_t^{[m]}, N_t^{[m]}, \left\langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \right\rangle, \ldots, \left\langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \right\rangle \right\rangle$ to $S_t$
  end for
  return $S_t$
end algorithm
```

4.2. Migrating the fastSLAM Approach from Scratch to Khepera Simulator

4.2.1. Basics of Khepera

Khepera is a robot designed as a research and teaching tool in the framework of a Swiss Research Priority Program. It allows confrontation to the real world of algorithms developed in simulation for trajectory execution, obstacle avoidance, preprocessing of sensory information, hypothesis on behaviors processing.

A Khepera robot is very small, about 55mm in diameter, and has two DC motors with incremental encoders and 8 infrared proximity/light sensors. Each sensor has a field-of-view of about 120°. The measurement of each sensor is related to the distance from the sensor to the object and the angle between the object and the sensor.

Figure 5 A general view of Khepera

4.2.2. KiKS (Khepera Simulator in MatLab)

We used a software package, called “KiKS”, to implement our fastSLAM algorithm on Khepera. KiKS is a Matlab application that simulates a Khepera robot connected to the computer in a very realistic way. The simulated Khepera is controlled from Matlab in the same way as real, physical Kheperas.

The simulated world is represented by a 2-dimensional obstacle matrix. Each matrix element corresponds to 1x1 mm^2. Each element in the obstacle matrix is either 0 or 1, where 1 indicates that there is an obstacle present and 0 indicates that there is no obstacle present.

4.2.3. Measurement and Motion Model for Khepera Simulator

We controlled Khepera robot by making the left and right motor to move 10mm each time we send “move” command to Khepera. We set one motor speed positive and other one to negative to make Khepera turn 90^o around itself without any translational motion.

During the motion of Khepera, sensors take measurements in every 20 ms. Sensor data taken during each 10mm movement of Khepera are averaged to minimize errors.

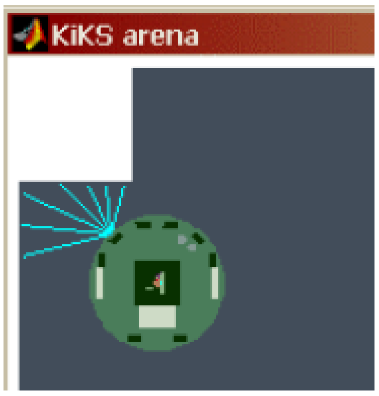

Figure 6 Sensor view of Khepera

Since we commanded only forward motion to the robot, only data taken from the 6 front sensors is considered to find the distance from and bearing of an obstacle, within the range.
To interpret the data provided by the 6 frontal sensors we used two different methods. For the first method we prepared a look up table by performing several experiments to map the data from the 6 front sensors to the location and bearing of the obstacle near Khepera. The look up table includes mean sensor data values for every possible situation that an obstacle exists within the range of Khepera which is -30 mm to 70 mm for y direction, and -70 mm to 80 mm for x direction, with respect to the origin of the Khepera. The second method uses the data taken from the two most activated sensors. We simply took the weighted average of these two sensors in each case to find the pose of the obstacle within a reasonable range.

5. Results

5.1. Results with Synthetic Data (proof of concept)

In order to verify the implementation, we tested our algorithm on synthetic data generated according to simple kinematic and measurement models. In Figure 7, the generated path and landmarks are shown with measurements. The heading (third element of the pose) is towards the direction of motion in all steps.

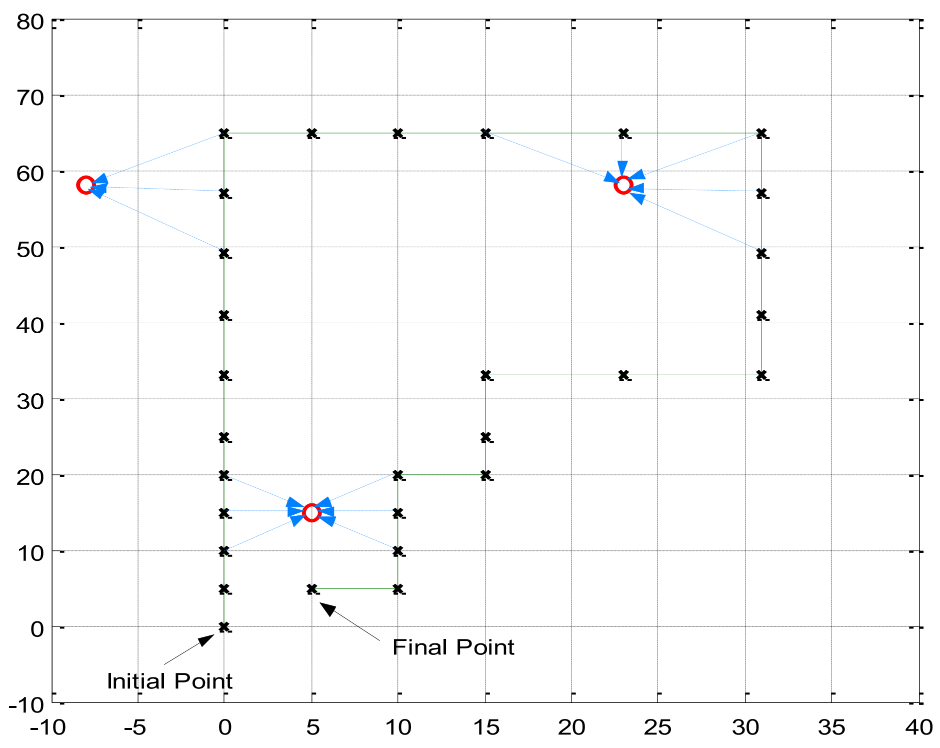

Figure 7 Original path of the synthetic data. Black crosses indicate the pose of the robot in each step. Note the landmarks and measurement points

In order to simulate the real case, we added white Gaussian noise to pose update and uniform noise to measurements. The reason for choosing uniform noise for the measurements was to build a non-linear measurement model. This way we expect to observe the power of particle filtering.

We selected 100 particles for the filter. In Figure 8 you can see the output of each step. Note that the variance of the pose (particle condensation) increases as the robot moves, due to noise. However, when it takes measurements from landmarks that are seen before, it updates both the landmark location and its pose resulting in a smaller variance. The steps where pose variance decreases are marked in the figure.

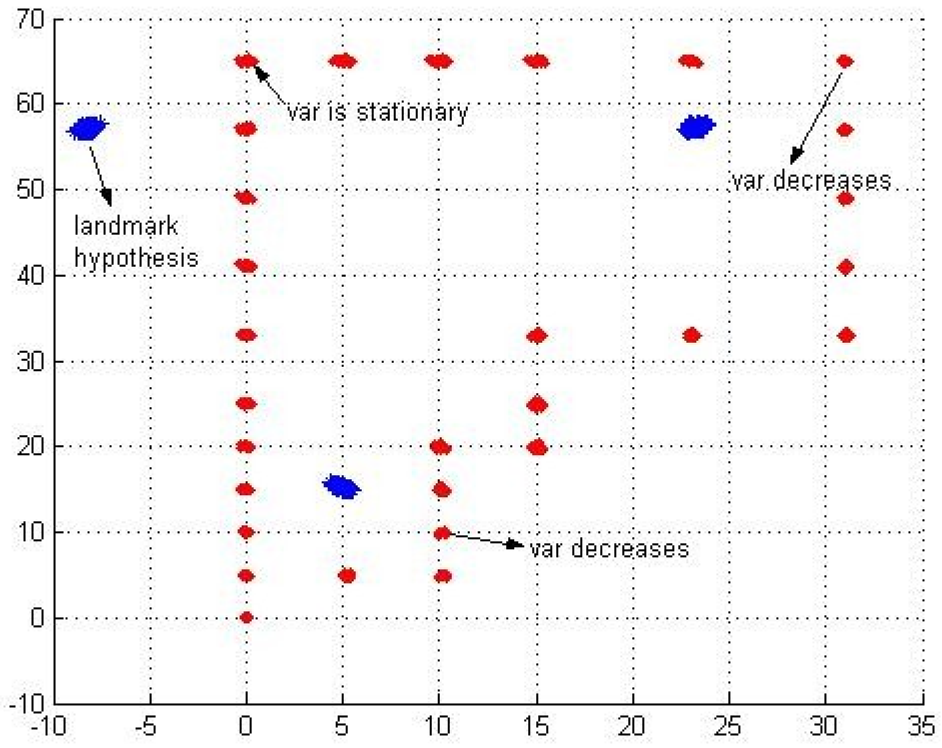

Figure 8 Result of the fastSLAM algorithm on synthetic data. All particles are shown in each step (100 particles)

The final result of the algorithm is a weighted sum of the particles both for landmarks and poses. The algorithm found three landmarks successfully. This proves that the data association (correspondence) problem is handled successfully by the filter. This way the EKF updates can be made robustly, meaning that the robot can distinguish a previously seen landmark from a new one. The final result is given in Figure 9.

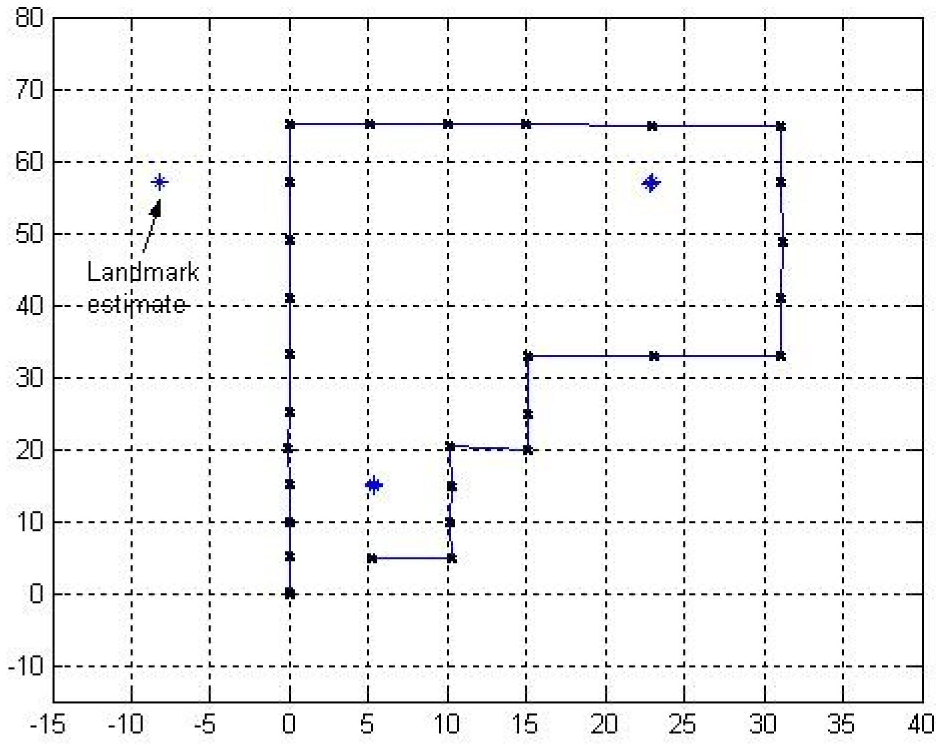

Figure 9 Final result of the fastSLAM algorithm on synthetic data. The poses and landmark locations are weighted average of the corresponding particles.

The effect of filtering on pose estimation can be analyzed by looking at the MSE error between the estimated pose and the real pose for filtered and unfiltered cases. In Figure 10 you can see the error in position and orientation of the robot separately for filtered and unfiltered cases. The MSE of the position error for filtered case is 0.3558. The MSE of the position error for unfiltered case is 0.4181. Similarly the MSE error for the orientation for filtered case is 1.9627. And it is 2.7855 for the unfiltered case. These results clearly show that filtering can handle noisy situations.

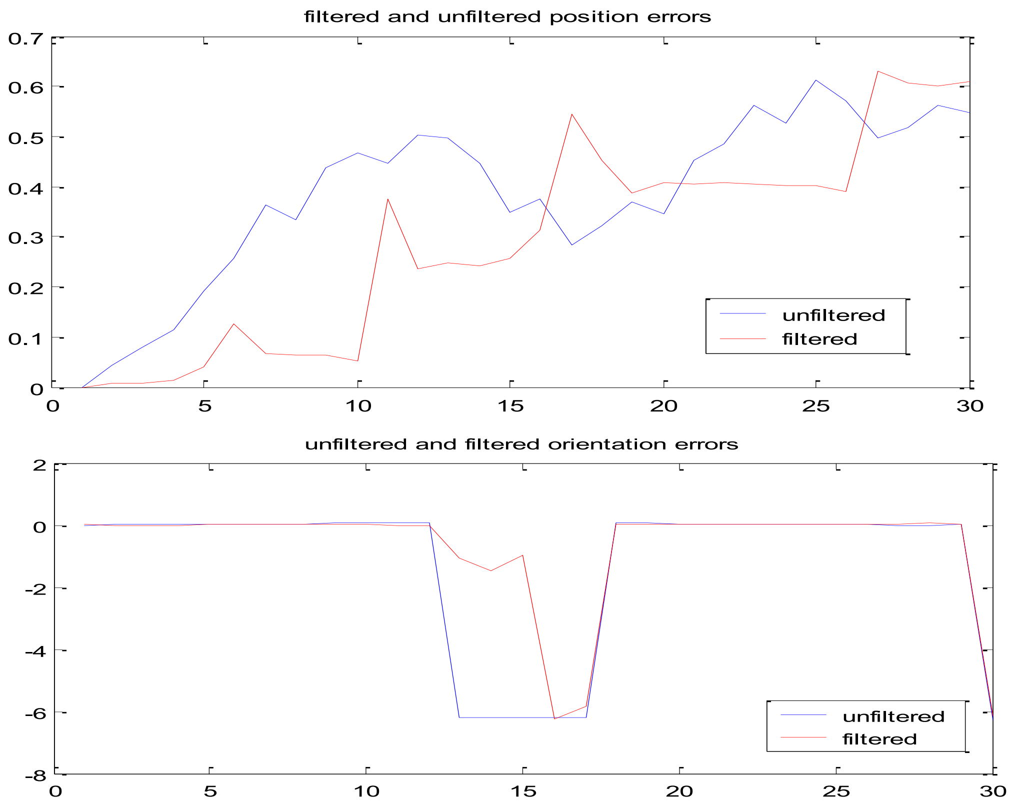

Figure 10 Position and orientation errors for filtered and unfiltered cases

5.2. Results for fastSLAM on Khepera Simulator

5.2.1. Measurement Model without LookUp Table

The fastSLAM algorithm is tested on Khepera robot simulator. The details were given in Chapter 4.2. In this section we'll show the results of fastSLAM for simple measurement model case. The path of the robot, therefore the inputs to the robot are predefined. Also the map is predefined and is static. However, initially the robot does not have any knowledge of the map. As the robot starts to move according to its predefined inputs it take measurements from the nearby landmarks. We added noise on the kinematic motion model of the robot since it does not make too much noise as will be in a real robot. The actual path and landmarks are given in Figure 11.

We used 50 particles for the fastSLAM filter in order to mimic the non-linear behavior and also have a pleasing processing time. The algorithm outputs the pose estimates and landmark estimates for all particles at each time increment. In figure these results are shown on top of each other. You can see the decrease in pose variance as the robot gets measurements as in the synthetic data case. The landmark estimations are not perfect due to very noisy sensor data.

Considering this noise, we can say that EKF estimation of the landmarks is very robust. In Figure 13, the final result of the fastSLAM algorithm is given. The pose and landmark estimates of particles are averaged according to their weights. This way the non-linearity is preserved. Note that the pose estimate is almost perfect and landmark estimates are pleasing.

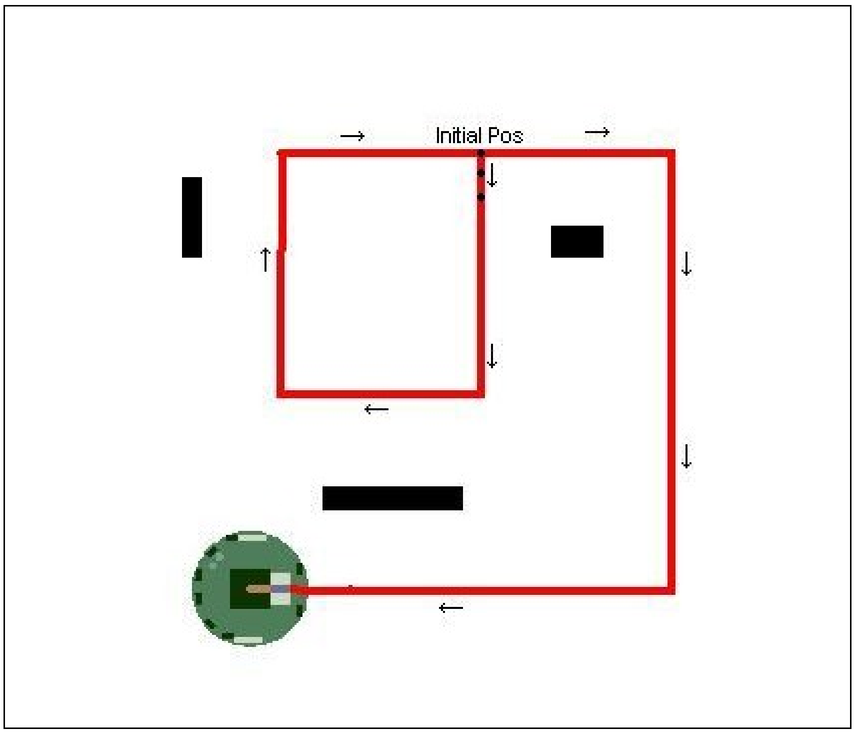

Figure 11 Actual path and landmarks on Khepera simulator

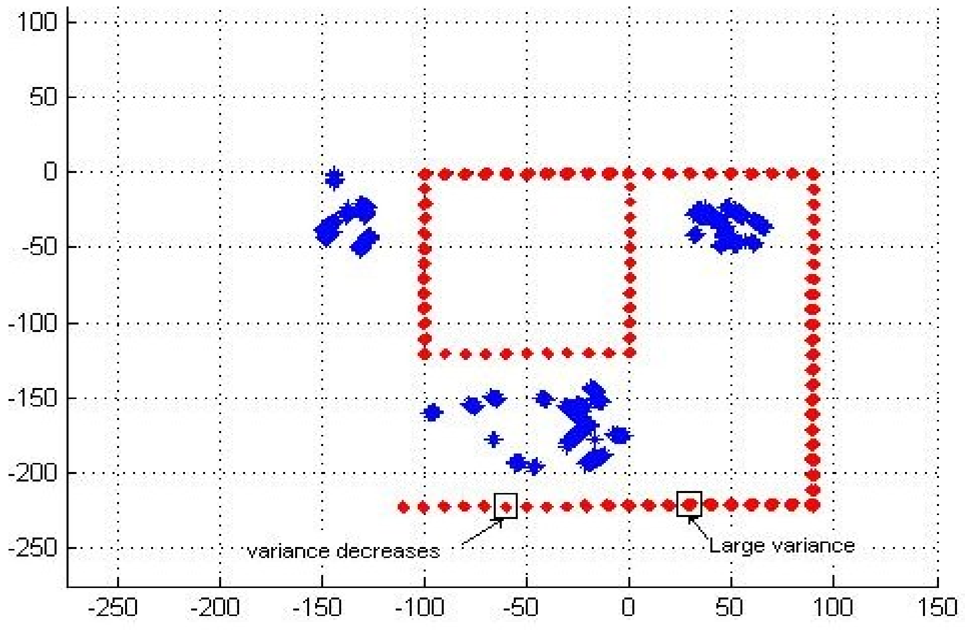

Figure 12 Result of fastSLAM (with simple sensor model)

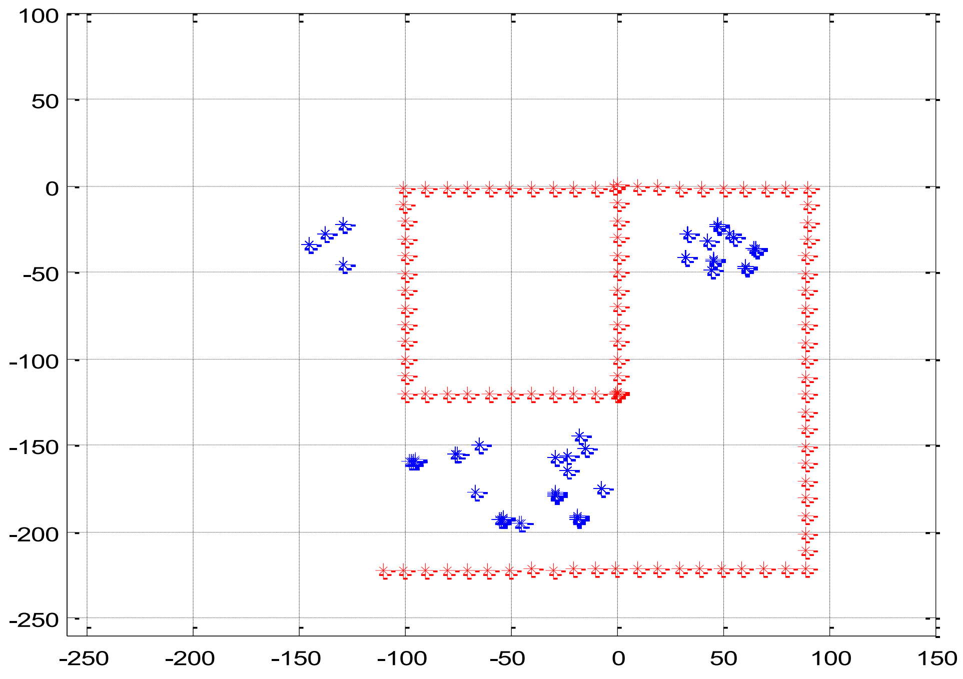

Figure 13 Weighted average result of fastSLAM (with simple sensor model)

In order to realize the effect of particle filtering and resampling we conducted the fastSLAM algorithm without the resampling step. The result is given in Figure 14. Note that the measurements are not taken into consideration therefore the variance gets larger and larger as the robot moves, like random-walk. As a result of this unreliable pose estimate, the landmark estimates also become unreliable.

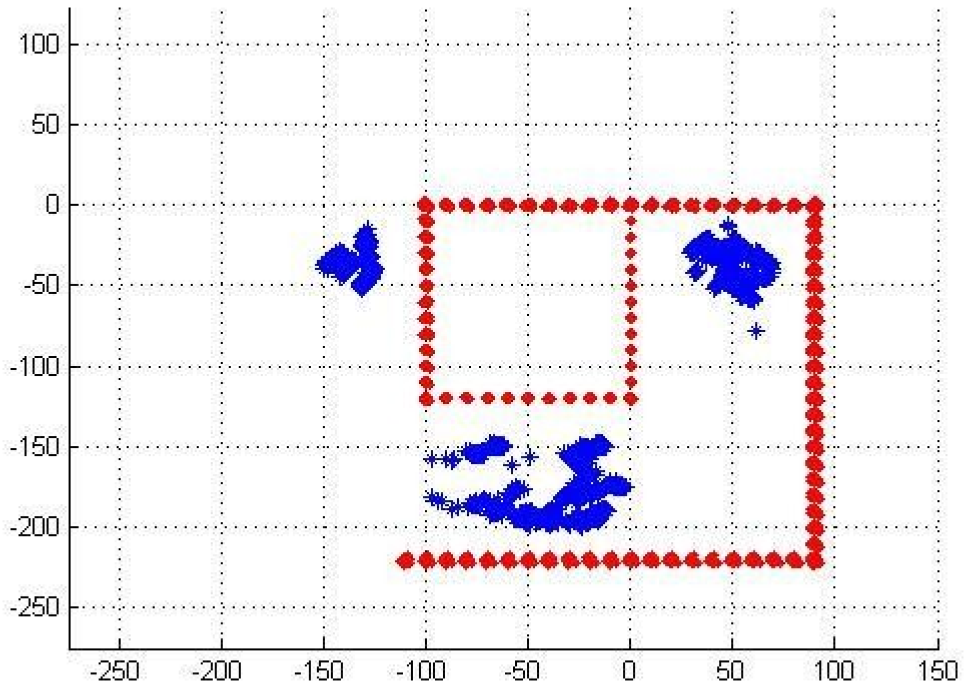

Figure 14 Result of fastSLAM without resampling (with simple sensor model)

5.2.2. Measurement Model with LookUp Table

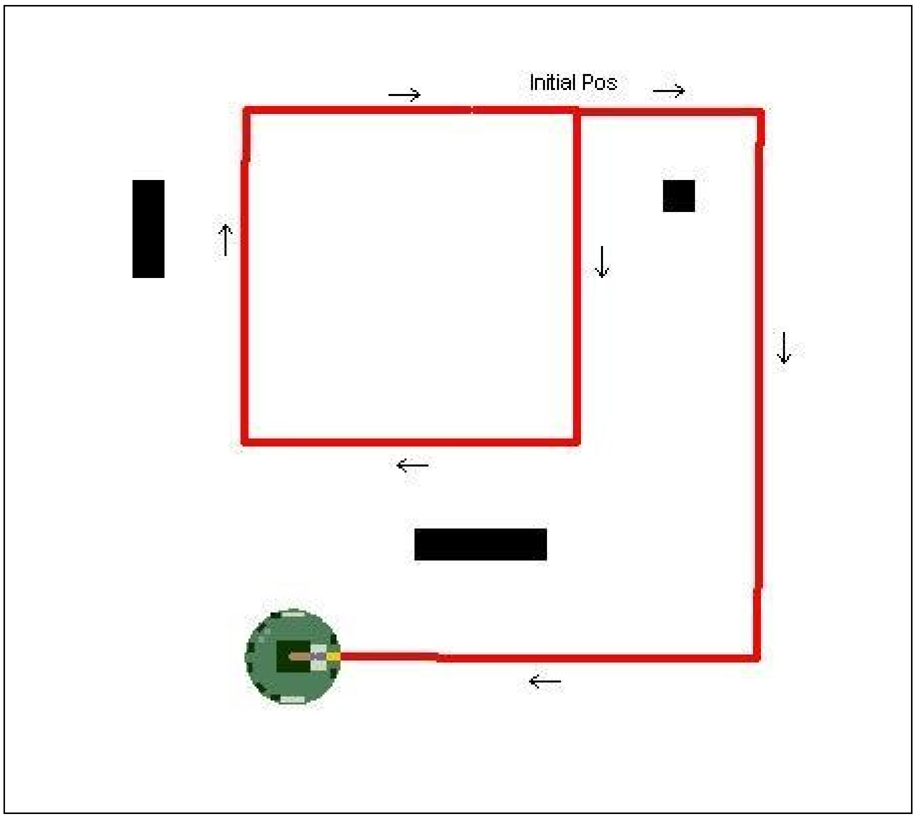

Figure 15 Actual path and landmarks on Khepera Simulator with look-up table method

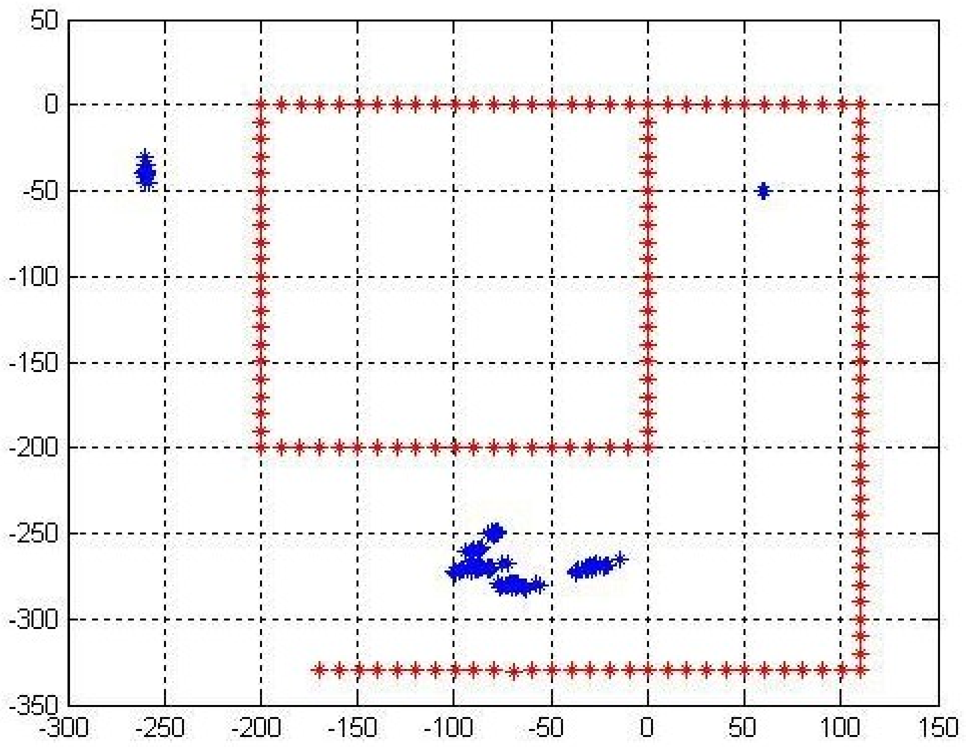

Figure 16 Weighted average result of fastSLAM on Khepera using lookup table

We observe that, although there are some alterations due to non-linearity of the sensors, the algorithm can successfully detect the landmarks. As the robot passes through the range of a landmark, the location of the landmark is updated. Hence, the resulting map becomes more and more accurate if the robot continues moving around and collecting data from the already observed landmarks. As we elaborate on the landmark positions in the figure, we see that the landmark on the upper-right of the map is located very precisely. This is due to the fact that the robot makes three passes around that landmark and it is able to identify it correctly on every pass. Furthermore, this landmark has a square shape, which is the same as the one we used in experiments to constitute the look-up table. Therefore, the algorithm can interpret the sensor data correctly, and locate the landmark accurately. On the other hand the landmark on the bottom is located somewhat less accurately, which is not surprising. Since the shape of that landmark is dissimilar to the one, which was used to form the look-up table, the sensor data cannot be converted into position information very precisely. However, the landmark is located around the correct position anyway.

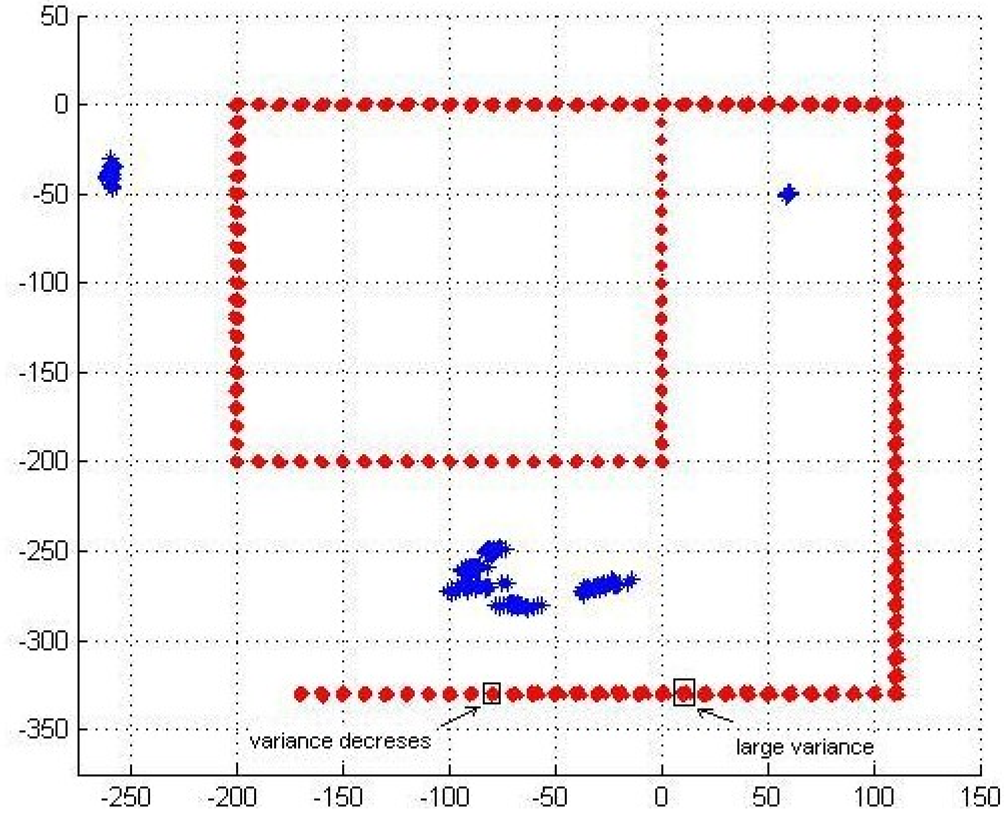

Figure 17 Result of fastSLAM with resampling using lookup table

From Figure 17, we can see that the variance of the robots position increases as it moves, due to the noise. However, when it passes around a landmark it observes the same landmark on several positions. This contributes better estimation to the pose and landmark location by means of Kalman filtering and resampling afterwards. From Figure 17, we can observe the points where the variance of the robot's position is decreased by this method.

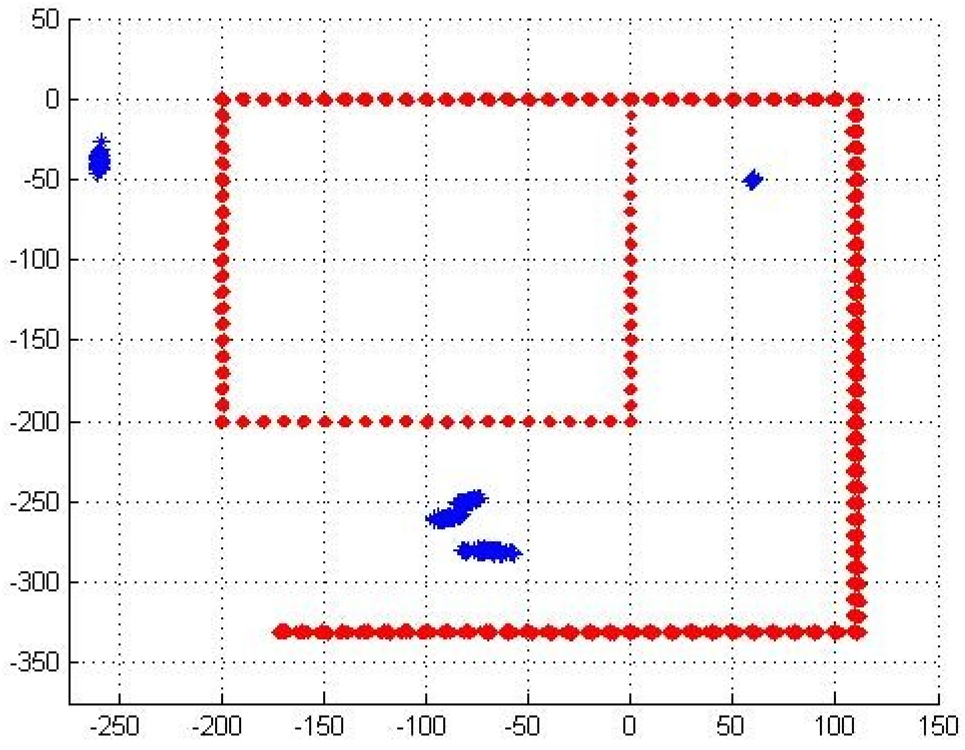

Figure 18 Result of fastSLAM without resampling using lookup table

The effect of re-sampling is drastically observed, when we disable the re-sampling process and compare the results. As it can be seen from the figure, the variance of the position increases continuously. Although, both motors of Khepera are of the same type, due to production issues their characteristics may differ. This will cause the position data to be less and less accurate as time passes. Hence, in the implementation on the real robot, the re-sampling becomes even more crucial.

5.3. Conclusions and Discussions

The results of the fastSLAM implementation on synthetic and Khepera data showed that, with a proper motion and measurement model, sufficient number of particles and proper initial covariance matrices, the algorithm is able to locate the robot and estimate the map at the same time reliably.

In the implementation stage, we realized that constructing a proper motion and measurement model is very crucial. Especially if the measurements are not suitable for modeling or cannot be mapped to what the filter wants correctly, the noise on the measurement gets very large. Therefore, a large initial covariance is chosen which makes the filter smoother. Actually this was the case with the Khepera simulator. The sensor structure was not suitable for SLAM implementation. Khepera has proximity sensors which are poor on localizing the nearby landmarks with bearing and distance data. But still, since it has multiple sensors, we adopted these sensors to our algorithm with large initial covariance and tuned our filter parameters accordingly. As a result, even for this noisy sensor data, the algorithm gave promising results. Nevertheless, it would be better to analyze the fastSLAM algorithm on a robot (or simulator) which is able to take measurements as distance and bearing. Laser range finders (scanning type) may be a suitable sensor type for this purpose.

6. Work Allocation for Team Members

Bengi Koç
Details of Khepera simulator, fastSLAM algorithm design, integration with Kiks, Kiks coding

Barış Tanrıverdi
fastSLAM detailed research, fastSLAM algorithm design, fastSLAM algorithm implementation (coding), integration with Kiks

Salim SIRTKAYA
Previous work about SLAM, fastSLAM algorithm design, implementation of Kalman filter (coding), motion modelling

7. References

[1] Biologically-inspired robotic mapping as an alternative to metric and topological approaches , *Alejandra Barrera Ramirez, Alfredo Weitzenfeld Ridel*

[2] Local Maps Fusion for Real Time Multirobot Indoor Simultaneous Localization and Mapping* , *Diego Rodriguez-Losada, Fernando Matia and Agustin Jimenez*

[3] ARIA and Matlab IntegrationWith Applications , *Jonas Borgström*

[4] Simultaneous Localization and Mapping, 2002 Summer School, *Eduardo Nebot Australian Centre for Field Robotics The University of Sydney NSW 2006 Australia*

[5] A Multilevel Relaxation Algorithm for Simultaneous Localization and Mapping , Udo Frese, Per Larsson, and Tom Duckett

[6] fastSLAM: An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data Association , *Sebastian Thrun1, Michael Montemerlo1,* Daphne Koller, BenWegbreit Juan Nieto, and Eduardo Nebot

[7] fastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit

[8] Rao Blackwellised Particle Filtering for Dynamic Bayesian Networks , Kevin Murphy and Stuart Russell

[9] Particle Filters in Robotics , Sebastian Thrun

[10] http://www.cim.mcgill.ca/~cathy/project765/node3.html

[11] Khepera II User Manual version 1.1 March 2002

[12] Khepera II Programming Manual

[13] KIKS is a Khepera Simulator User Guide

[14] Khepera II IR Sensor Report version 1.1 January 2002