

SVM Based SLAM Algorithm for Autonomous Mobile Robots

Jiali Shen and Huosheng Hu

*Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, United Kingdom*

Email: jshenh@essex.ac.uk, hhu@essex.ac.uk

Abstract - Support Vector Machine (SVM) is a classification algorithm with some advantages over other machine learning methods, which provides an efficient tool to select new features from sensor observations. This paper presents a SVM based Simultaneous Localization and Mapping (SLAM) algorithm that enables autonomous mobile robots to operate in a dynamic or unstructured environment. The observation models and the SVM based visual feature processing algorithm are designed. SVM is adopted in several steps of observation in this paper in order to achieve fast processing and accurate localization. The simulation results are given to show its feasibility and good performance.

Index Terms - Support Vector Machine, Autonomous Navigation, Simultaneous Localization and Mapping.

I. INTRODUCTION

Support Vector Machine (SVM) is an advanced classification and pattern recognition algorithm. The basic idea of SVM is to find a hyperplane that can divide the manually classified sets of vectors (training data) and provide the maximum margin between different sets of training data. The vectors on the edges of margins decide the hyperplane, namely "Support Vector". In some linearly non-separable case, kernel functions have been introduced to transfer the data space to a new one in which training vectors are linearly separable [3][4]. Recently, SVM has been adopted in the area of robot navigation, such as view-based localisation in outdoor environments [7], flexible view recognition for indoor navigation [5], and a two-step topological localization algorithm [1]. However, a map is normally required in advance in these approaches.

In many real-world applications, a prior map may not be available or unable to represent a dynamic environment. Therefore, it is required that a robot can explore an unknown or dynamic environment to build maps and localise itself at same time, which is called Simultaneously Localization and Mapping (SLAM) [11]. Extended Kalman Filter (EKF) is one of powerful tools in the most of SLAM implementations, which are also adopted in this paper for data fusion. The main contribution of this paper is the construction of observation models and the SVM based visual feature processing algorithm, based on our previous work [1]. SVM is adopted in several steps of observation in order to achieve fast processing and accurate localization.

The rest of this paper is organized as follows. EKF and its application in robotic navigation are briefly reviewed in

Section II. Section III presents the SVM based observation algorithm, which include the feature recognition and relative localization of the observed beacons. The EKF loop for this system and the SVM based classifier are then described in Section IV. Section V presents some experimental results of our SLAM algorithm using a Matlab-based simulator. Finally, a brief conclusion and future work are given in Section VI.

II. EKF IN ROBOT LOCALIZATION

Extended Kalman Filter (EKF) has been adopted in robotic localization for many years [12], which have the following system model:

$$X_k = F(X_{k-1}, u_k) + w_k \quad (1)$$

where X_k is the robot state at the k^{th} moment, i.e. $X_k = [x_k, y_k, \theta_k]^T$, and $u_k = [v, R]^T$ is the control input (translation and rotation speed). F is the system transforming functions, and w_k is the noises with covariance matrix Q_k . For a differential driving vehicle, F is embodied as:

$$\begin{aligned} x_k &= x_{k-1} + v_k \cos(\theta) \times \Delta t \\ y_k &= y_{k-1} + v_k \sin(\theta) \times \Delta t \\ \theta_k &= \theta_{k-1} + R \times \Delta t \end{aligned} \quad (2)$$

The observation model is:

$$Z_k = H(X_k) + v_k \quad (3)$$

where Z_k is the observation vector at k^{th} moment, H is the observation function, and v_k is the observation noise with the covariance matrix R_k .

Since the noise in the system model accumulates and is unbounded, the robot must fuse some external feature information to reduce the noise and make a correction from time to time. EKF is a recursive procedure that uses the observation to revise the noisy prediction in the procedure shown in Fig. 1.

The first step is to predict the robot state transition and observation with noises:

$$\hat{X}_{k|k-1} = F(x_{k-1}, u_k) \quad (4)$$

$$P_{k|k-1} = F_k P_{k-1} F_k^T + Q_k \quad (5)$$

$$\hat{Z}_{k|k-1} = H(\hat{X}_{k|k-1}) \quad (6)$$

where F_k is the Jacobean of transformation functions, and P is the covariance matrix of the estimation error.

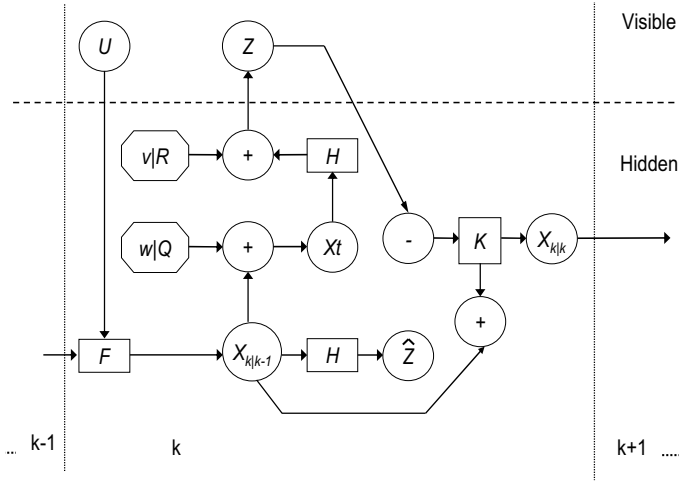


Fig. 1 Normal EKF structure

The next step is to update the state prediction with the new observation:

$$Y_k = Z_k - \hat{Z}_{k|k-1} \quad (7)$$

$$S_k = H_k P_{k|k-1} H_k^T + R \quad (8)$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \quad (9)$$

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k Y_k \quad (10)$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (11)$$

where H_k is the Jacobian of Observing Function H , $\hat{X}_{k|k}$ is the updated state estimation with the updated covariance matrix $P_{k|k}$, and K_k is a Kalman gain matrix, which is the best revision operator in Gaussian noise environment.

III. SVM BASED OBSERVATION

In our previous development [1], once a beacon is observed, the relative position of the robot can be deduced by a two-stage localization algorithm in the beacon's coordinate frame as $X_B = [r, \theta]^T$. Based on the information of the beacons global position (the map), the robot can be located in a global frame by a coordinate transformation.

$$X_G = C_B^G X_B \quad (12)$$

where X_G is a Global pose vector, X_B is a pose vector in the beacon frame, and C_B^G is a conversation matrix.

In an unknown environment, we do not have the position of beacons, and therefore a beacon to Global conversation C_B^G is not available. However, the relative pose of the robot, X_B , can be used to inverse the beacon's position in the robot frame. This relative position can be used as "observation" in SLAM. Fig. 2 presents a flow chart of our algorithm [1], in which the left blocks express the functional modules, and the right blocks indicate the information required. Below the dash line are the novel usage of these modules.

Beacon Object Extraction (BOE) only requires approximated and general knowledge about an indoor environment. Any natural indoor environment contains some

kinds of objects, such as extinguishers, radiators, doors, windows, etc. Based on the trained SVMs and the developed modules, the SLAM algorithm could extract the unexpected beacons to calculate the angles to these objects, by using the image, focus length and pinhole model.

The beacon recognition module (BOR) was designed to identify the extracted objects with known sizes, shapes and other parameters; therefore SVM in this module requires detailed training. This module could be applied in SLAM under the condition that objects are known. When an object is recognized, the program can calculate the observing distance by comparing the image size and the real size in map data. This is one of the functions in the developed module, i.e. Grid based localization.

The Coordinate Conversion module outputs the global position of the robot, but it requires the map in advance, which is not available for SLAM utilities. The SVM based observation can be expressed as:

$$Z = H(X, b) + v \quad (13)$$

where Z is the observation vector, X is the robot pose, b is the beacon objects and v is the observation noise with the covariance matrix R . H is the observation functions, which is the SVM based modules developed in this paper.

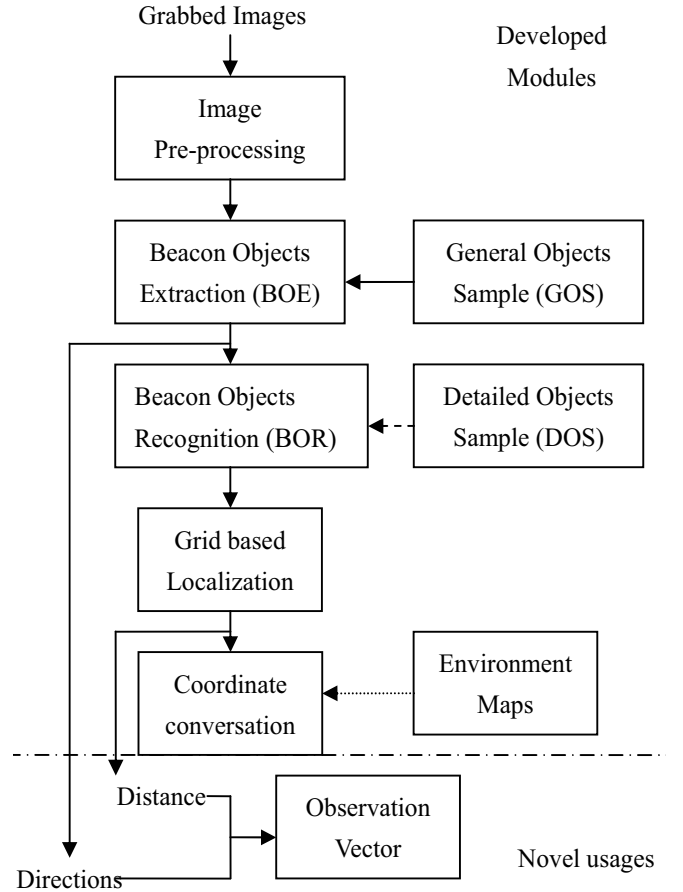


Fig. 2 Flow chart of generating observation vectors

According to the prior knowledge, there are two possible models: when the objects details are known

$$Z = \begin{bmatrix} dis_1, \dots, dis_n \\ dir_1, \dots, dir_n \end{bmatrix} \quad (14)$$

where $dis[1..n]$ denote the distance to n beacons, and $dir[1..n]$ denote the direction to n beacons.

When the beacon is unexpected and the distance information is not available, we have

$$Z = [dir_1, \dots, dir_n] \quad (15)$$

IV. EKF BASED SLAM

In SLAM algorithms, the map is updated with new observations, e.g. Bailey's EKF-SLAM [13] and Castellanos's algorithm [14]. The features in the map are also expressed as the elements in the state vector X , with the format $X_k = [x, y, \theta, r_1, \varphi_1, r_2, \varphi_2, \dots]^T$, where r_i and φ_i are the distance and direction of the i^{th} found features. Beside the state estimation in Equations (4)-(11), there is an element augment process in Bailey's flowchart to add new features to the state vector. The features in new observations are firstly classified as "found" and "novel". Only the "found" features are used to update the state; the "novel" features are listed after the state vector. In Bailey's algorithm, features are simply classified by a threshold. If a feature is far from any found features, it is a new one. Otherwise, it is judged as the new measurement of an old feature.

For some noisy observations, this simple strategy could generate wrong mapping results. As shown in Fig. 3, the small star on the wall is the true beacon, and the ellipses with crosses at the centre are the features in the built map. The one on the left is a false mapping due to an observation with the noise that is larger than the threshold. On the contrary, objects that are too close to beacons are easily regarded as one in the map building under this simple rule.

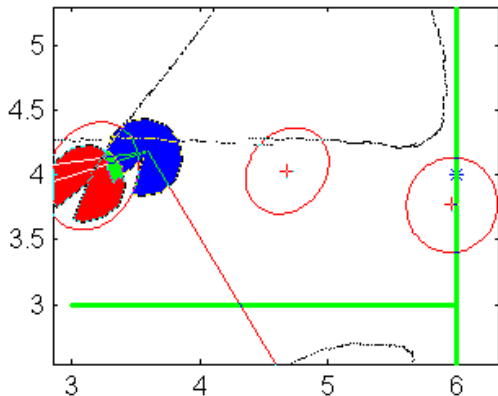


Fig. 3 A wrong mapping result with a classifier of threshold features

Fortunately, visual beacons contain plenty of information to classify the observations. This paper introduces a SVM for each observed beacon. The system model has the format as follows:

$$X_k = [x, y, \theta, r_1, \varphi_1, s_1, r_2, \varphi_2, s_2, \dots]^T$$

where r_i and φ_i are the distance and direction of the i^{th} found feature. s_i is a pointer to SVM for the feature identification.

The each beacon's SVM records visual information such as colour histogram, textures, outline shape, etc. It is set when it is added in the map, and updated by further training when it is observed again. When a beacon is observed, it is classified in two steps. A distance based classifier with a relatively bigger threshold is firstly applied to select the possible candidate beacons that are not too far away from the observation. Secondly, SVMs for all candidates are applied to the new observed beacon to make a final decision on whether the beacon is a new one or a known one. The EKF with a SVM based classifier and a feature augment module forms a data fusion loop for SLAM in this paper, which has the structure shown in Fig. 4.

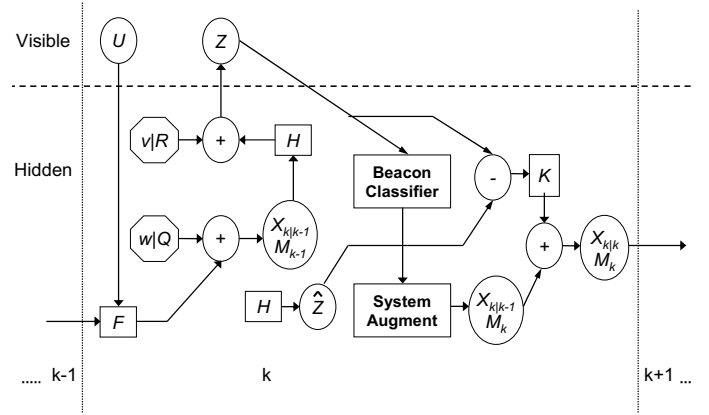


Fig. 4 The EKF with SVM classifier and System Augment

The vector for the SVM based classifier is composed of several parts: the normalized three-dimensional colour histogram, the height/width ratio of the outline rectangle and the contrast of each colour channel and edges. Once a beacon is extracted from observations, it is transferred into a vector as explained above. If the beacon is classified as a new one, three new elements are added into the system vector, and a new SVM is trained with the new beacon vector and other existed beacon vectors as the sample of the other class. If the beacon is regarded as a new observation of an existed object, the vector is used to train the corresponding SVM. In some particular condition, two different beacons have similar visual features, and can only be separated by the distance, they may share the observation data and use one SVM.

At the beginning, if the mapped beacons and training data are limited and SVMs are unreliable, the classifier relies more on distance rules. As SLAM continues, more samples for beacons are acquired and SVMs will be well trained. The SVM based classifier is then take the place of new observations process.

V. SIMULATION RESULTS AND ANALYSIS

A. Experiment Environment

The algorithm is tested in a Matlab based simulator that has been developed to show all the system states in a dynamic window. More specifically, the simulator has five functions: (i) Environment building; (ii) Robot and observation model setting, including the driving and control system and noise level. The simulator also calculates the image frame that the robot can see, according to the precise robot pose, the parameters of camera, and the environment. (iii) Bumping reaction simulation. If the robot touches the “walls”, the simulator can stop the robot even when it is commanded to move forward. This function prevents the robot passing through the “wall” like a ghost, and makes the simulation running like the experiment on real robots. (iv) Real-time display of the running processing and observations. (v) Statistical results of the whole running procedure, including the transient and average localization error.

The simulator has the architecture shown in Fig. 5. The environment, the robot and the observations are explained in “Customer Configure” file by the user. The algorithms to be tested are separated Matlab functions with standard parameters defined in advance in the “Customer Subroutines” file. The “program configure” module reads the user’s files and builds the 2-D experimental court and the robot. This module integrates customer subroutines in the whole program. The grey blocks indicate the module taken from the user’s file, while the white blocks mean the inner functions of the simulator. In this paper, the “data fusing & localization” module is the SVM based SLAM explained above. When the simulation starts, the “obstacle avoidance” modules take the role of preventing the robot passing the wall.

B. Experiment Procedure

We build a room with a size of 11×8 square metres, with two walls roughly separating in the middle. 6 Beacons are placed along two side walls. The robot is a differential driven robot, whose round body has a diameter of 0.6 metres. The robot is equipped with two DC motors wheels at the bottom; its wheelbase is 0.4 metres.

The robot starts from one room, and randomly navigates around the environment, as shown in Fig. 6. Without any priori knowledge of the environment, the robot locates itself and builds the map based on its initial pose, $X_0 = [0, 0, 0]^T$.

The observation noise is embedded in the visual processing algorithm, but the internal sensor performance and control accuracy also play important roles to test the robustness of the SLAM, and limited by the hardware technique of the robot. Several simulations have been tested with the different control and odometer noises. During the robot navigation, all the data have been recorded. Both observation models are tested: in the distance observation model, the detailed beacon objects are provided for the training and in direction only model, the robot has no information about the real size of objects.

The statistic results are listed in Table 1 – Table 3, where the symbol σ indicates the standard divisions of the speedometers for each driven wheel. This value reflects the uncertainties of two wheels of the differential driven robot in the simulator, based on the hypothesis that both wheels have same characters. What to be noticed is: the σ is expressed in the style of velocity; since the processing speed is 25 frames per second, the uncertainty in each EKF prediction is $1/25$ of σ value. For each single noise observation condition, a 200-second experiment was simulated, therefore 5000 sets of data were recorded in a single run.

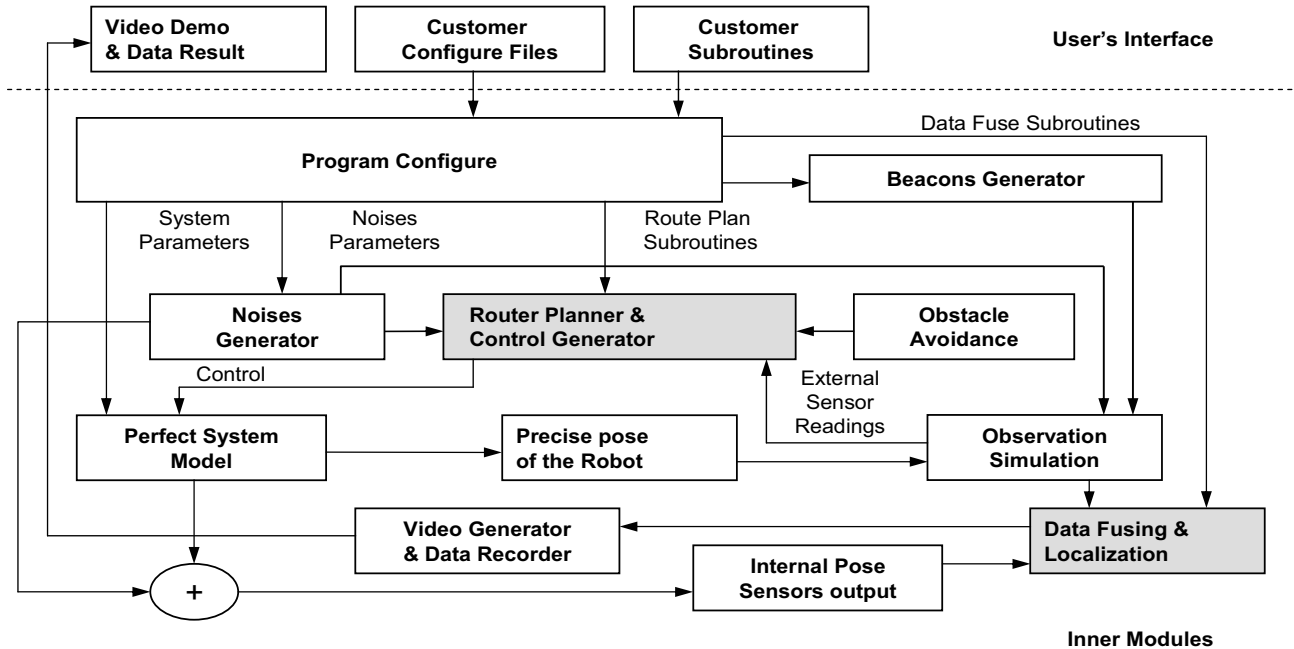


Fig. 5 The software architecture of the robotic navigation simulator

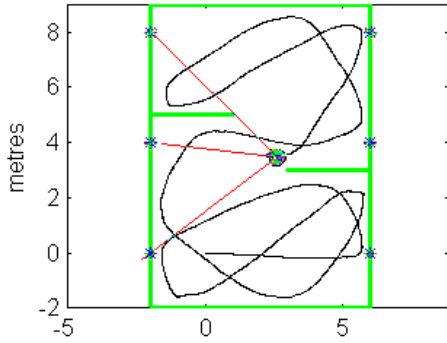


Fig. 6 The experimental environment and procedure

C. Experiment Results

The first set of experiments are under the condition that $\sigma=0.2$, which means relatively reliable internal sensor outputs. During the whole navigation process, we observed that the located position was very close to the true position of the robot; the encoder error is not accumulated to a large value before the robot found some beacons and begins to update the position with the external observations. The results are listed in Table 1 and the transient localization error is shown in Fig. 7.

TABLE I

RESULT ($\sigma=0.3$ m/s) Unit: cm

Statistic Value	Distance Observation	Direction Only
Maximum Error	20.36	32.24
Minimum Error	0.091	0.14
Mean Error	6.94	12.99
Standard Divisions	3.38	6.63

TABLE II

RESULT ($\sigma=0.6$ m/s) Unit: cm

Statistic Value	Distance Observation	Direction Only
Maximum Error	28.86	41.55
Minimum Error	0.11	0.021
Mean Error	10.11	15.97
Standard Divisions	4.40	7.99

TABLE III

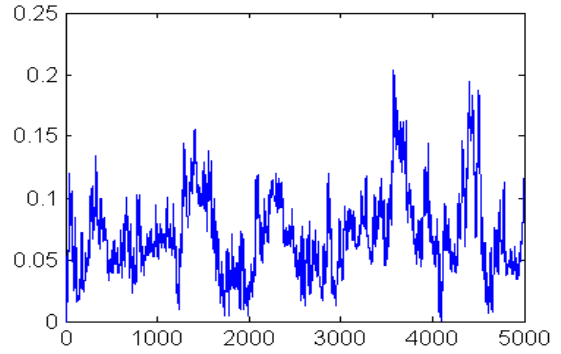
RESULT ($\sigma=1.0$ m/s) Unit: cm

Statistic Value	Distance Observation	Direction Only
Maximum Error	64.09	57.78
Minimum Error	0.24	1.03
Mean Error	15.10	22.83
Standard Divisions	8.94	10.01

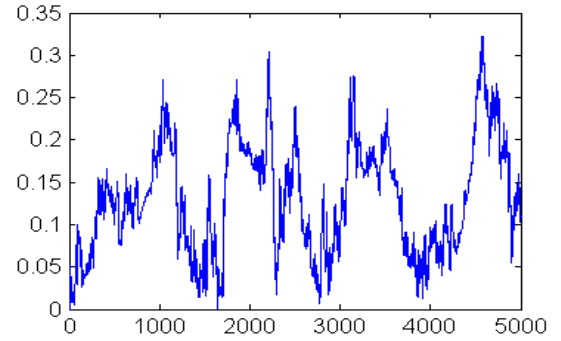
In the condition that $\sigma=0.6$, the algorithm performed better in a noisy environment, which is as expected. Table 2 gives the statistical results of the experiments and Fig. 8 shows the transient error during the navigation process.

In the condition that $\sigma=1.0$: The different noises on two driving wheels made the robot steering become uncertain,

which is indicated by θ in the system state vector. The inaccurate estimation of θ would cause the divergence between the estimated and real robot positions if it cannot be corrected in time. Therefore, at the beginning of the robot navigation process, the localization result deviated from the true position rapidly. The localization error was then limited and reduced down to an acceptable range after the beacons were found and the map was partly built. But in the direction only case, compared to the distance available case, the beacon contains less information and it was harder to build a reliable map quickly during the SLAM, the localization error decreased much slower. The experimental data are listed in Table 3, and the transient localization error is shown in Fig. 9.

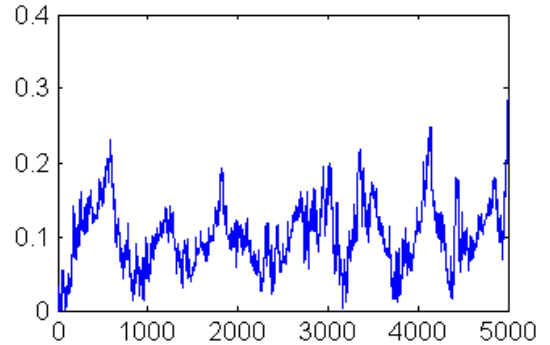


(a) The result of the distance available case

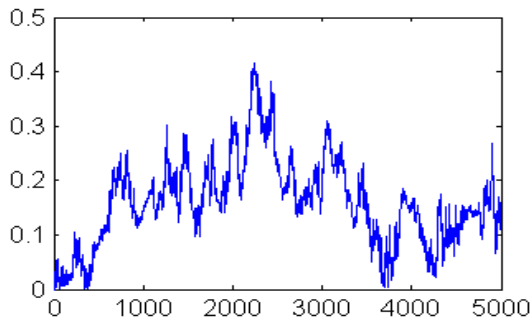


(b) The result of the direction only case

Fig. 7 The localization error in the experiments ($\sigma=0.2$)

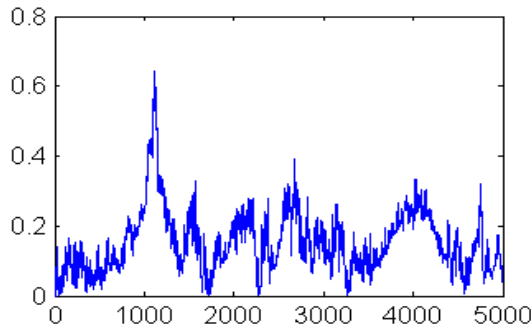


(a) The result of the distance available case

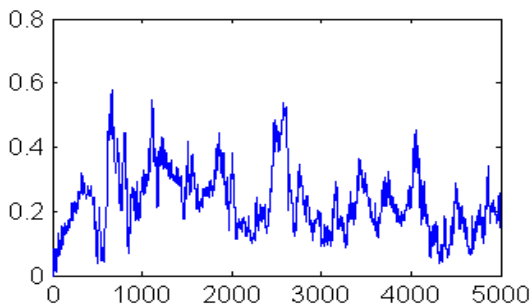


(b) The result of the direction only case

Fig. 8 The localization error in the experiments ($\sigma=0.6$)



(a) The result of the distance available case



(b) The result of the direction only case

Fig. 9 The localization error in the experiments ($\sigma=1.0$)

According to the results, we can see that errors are generally acceptable. The maximum errors usually exist in the early moment of the experiment, when the map has not been built accurately. Fig. 7, Fig. 8 and Fig. 9 are the real time error records during the whole experiment, which also shows the tendency that the errors become smaller and stable with the progress.

VI. CONCLUSION AND FUTURE WORK

This paper presents a SLAM algorithm based on support vector machine and Extended Kalman Filter. The SVM based observations provide fast visual signal processing capability for transferring images into position vectors. The EKF algorithm fuses sensor observations and system predictions to implement concurrent localization and mapping. SVMs are introduced to select new features to be mapped from the observations. They play an important role in the procedure, and make it possible to obtain distance information from a

single camera. SVMs also improve beacon classification so that the map is built more accurately. The SVM based SLAM algorithm has been tested in a Matlab-based simulator and achieved satisfying results.

Our future work will focus on the implementation of this SVM-based SLAM algorithm on a real mobile robot in real time in order to test its robustness and improve its accuracy under uncertainty in the real world.

REFERENCES

- [1] J. Shen and H. Hu, Visual based localization for mobile robots with Support Vector Machine, Proceedings of the 32nd IEEE Industrial Electronics Society Annual Conference, Paris, Nov. 2006, pages 4176-4181
- [2] http://www.acfr.usyd.edu.au/homepages/academic/tbailey/software/slam_sims/ekfslam_v1.0.zip
- [3] S. Gunn, "Support vector Machines for Classification and Regression", <http://homepages.cae.wisc.edu/~ece539/software/svmtoolbox/svm.pdf>
- [4] C. Hsu, C. Chang, and C. Lin, "A Practical Guide to Support Vector Classification" <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>
- [5] I. Autio and T. Elomaa, "Flexible view recognition for indoor navigation based on Babor filters and support vector machines", Journal of Pattern Recognition, Vol. 36, 2003, pages.2769-2779
- [6] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", Journal of Data Mining and Knowledge Discovery, Vol. 2, 1998, pages 121-167
- [7] H. Morita, M. Hild. J. Miura, and Y. Shirai, "View-Based Localization in Outdoor Environments based on Support Vector Learning", Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pages 3083-3088
- [8] H. Durrant-Whyte and T. Bailey, Simultaneous Localization and Mapping: Part I, IEEE Robotics and Automation Magazine, June 2006, pages 99-106
- [9] T. Bailey and H. Durrant-Whyte, Simultaneous Localization and Mapping(SLAM): Part II, IEEE Robotics and Automation Magazine, September 2006, pages 108-117
- [10] S. Se, D.G. Lowe and J.J. Little, Vision-Based Global Localization and Mapping for Mobile Robots, IEEE Transactions on Robotics, Vol. 21, No. 3, June 2005, pages 364-375
- [11] A. J. Davison and N. Kita, Sequential Localisation and map-building for real-time computer vision and robotics, Journal of Robotics and Autonomous Systems Vol. 36, 2001, pages 171-183
- [12] H. Hu and D Gu, "Landmark-based Navigation of Autonomous Robots in Industry", Journal of Industrial Robot, Vol. 27, No. 6, 2000, Pages 458-467
- [13] T. Bailey, J. Nieto, J. Guivant, M. Stevens, E. Nebot, Consistency of the EKF-SLAM Algorithm, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2006, Beijing, China, pages 3562-3568
- [14] J.A. Castellanos, R. Marinex-Cantin, J.D. Tard'os, J. Neira, Robocentric map joining: Improving the consistency of EKF-SLAM, Journal of Robotics and Autonomous Systems, Volume 55, Issue 1, January 2007, pages 21-29