

Time Optimal Robot Motion Control in Simultaneous Localization and Map Building (SLAM) Problem

Shoudong Huang, Zhan Wang, Gamini Dissanayake
Centre of Excellence in Autonomous Systems
Faculty of Engineering,
University of Technology, Sydney
Sydney, Australia
Email: {sdhuang,zwang,gdissa}@eng.uts.edu.au

Abstract—This paper provides a technique for minimal time robot motion control in the estimation-theoretic based simultaneous localizations and map building (SLAM) problem. We consider the scenario that the robot needs to go to a destination which is a prescribed location in the coordinate system referenced by its starting position. The task of the robot is to reach the destination within minimal time while localizing itself and building a map of the environment with a prescribed accuracy. This task may be a real navigation task or may be a subtask in a SLAM problem of a large unknown environment. A global sub-optimal control law is derived using dynamic programming techniques.

I. INTRODUCTION

The simultaneous localization and map building (SLAM) problem is to place a robot at an unknown location in an unknown environment and then have it build a map of this environment while simultaneously using this map to compute the robot location. The SLAM problem is important in a large range of applications where absolute position or precise map information is not available. The estimation-theoretic based approach (in particular, the Kalman filter based approach) for SLAM problem was well developed (e.g. [3], [8]). The Kalman filter is used to provide estimates of robot and landmark locations. Especially, it was proved in [3] that a solution to the general SLAM problem exists and it is indeed possible to construct an accurate map and simultaneously compute robot position estimates without any prior accurate knowledge of robot or landmark locations.

Many SLAM algorithms did not take the motion control into account. That is, the robot trajectory is predetermined or randomly chosen. In fact, choosing the right trajectory during robot exploration is much more beneficial than using a fixed trajectory or moving the robot randomly. For example, in the localization problem, an active localization approach based on Markov localization [5] makes the robot localize itself more efficiently.

The motion control (trajectory control) in SLAM is not trivial. As far as we know, the first paper addressing the motion control in SLAM problem is [4] where Feder et al. provided an adaptive motion control techniques in SLAM. The robot creates a map and localizes itself simultaneously

while making local decisions on where to move next in order to maximize the information obtained from the observations. The inverse of the estimation error covariance is used as an approximation of the Fisher information of the system. The choice of the decision is based on a single-step look-ahead and it is assumed that no new beacons will be observed in the next step, hence, the results are locally optimal [4].

Though this kind of local optimization methods may work well in some situations, it can result in very poor performance in some particular environment. For example, consider a SLAM problem in a one dimensional environment. The motion that minimizes the error covariance in the next estimates of robot and observed landmark locations is, in many cases, to make the robot stationary or move backwards (revisiting old landmarks can reduce the estimation error). If the local optimization technique in [4] is used to decide the motion of the robot, then the robot may only choose to remain stationary or go backward in order to reduce the estimation error, this makes it impossible to complete the SLAM task. From this point of view, it is very important for the robot to keep the overall objective in mind at all the time. Recently, Alex Makarenko et al. [1] considered the integrated exploration problem where the tasks of localization, mapping and motion control are combined together. A frontier method [10] is used to generate potential destinations, then multiple utility functions are used to evaluate the utility of the potential destinations. The destination with the highest total utility is selected as the next destination. After the next destination is decided, a path to the destination is planned using the navigation function method and the robot will move to the destination by following the planned path. In this method, the destinations are generally located on the edge of the explored and the unexplored regions, and how to choose the speeds such that the robot can reach the next destination in minimal time is not considered.

It is the purpose of this paper to develop a minimal time motion control technique in the SLAM problem when the destination is a prescribed location related to the starting point (this point may be far away from the explored region).

We want the robot to reach the destination within minimal time while keeping the error in the estimations of robot and landmark locations below a prescribed level.

The paper is organized as follows. In Section II-A, the estimation-theoretic SLAM algorithm is briefly reviewed. The motivation and formulation of the problem are given in Sections II-B and II-C, respectively. Section III describes the development of a solution to the problem using dynamic programming. An example is given in Section IV with simulation results. Section V concludes the paper.

II. PROBLEM STATEMENT

A. Estimation-theoretic SLAM algorithm

We briefly review the estimation-theoretic SLAM algorithm provided in [3], [4].

Let the robot's state be denoted by $\mathbf{x}_r = [x_r, y_r, \phi]^T$ and the dynamic model for the robot was given by

$$\mathbf{x}_{r_{k+1}} = \mathbf{f}(\mathbf{x}_{r_k}, \mathbf{u}_k, \mathbf{d}_x) \quad (1)$$

where \mathbf{u}_k is the control input at time k , \mathbf{d}_x is the Gaussian process noise with covariance Σ . The exact formula of function \mathbf{f} depends on the type of the robot and the channels where the process noise \mathbf{d}_x comes in, etc.

The landmarks are assumed to be stationary and we use \mathbf{x}_m to denote the state of all the landmarks. Let the state vector $\mathbf{x} = [\mathbf{x}_r^T, \mathbf{x}_m^T]^T$ contain both the robot states \mathbf{x}_r and the landmark states \mathbf{x}_m , then the process model can be written as

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_x) = \begin{bmatrix} \mathbf{f}(\mathbf{x}_{r_k}, \mathbf{u}_k, \mathbf{d}_x) \\ \mathbf{x}_m \end{bmatrix}. \quad (2)$$

The observation model is

$$\mathbf{z}_k = \mathbf{H}(\mathbf{x}_k) + \mathbf{d}_z \quad (3)$$

where \mathbf{H} defines the nonlinear coordinate transformation from state to observation coordinates. \mathbf{d}_z is the Gaussian observation noise with covariance \mathbf{R} .

The estimation-theoretic SLAM algorithm is based on the work of Smith et al. [8]. It uses the Extended Kalman Filter algorithm to optimally estimate the state vector \mathbf{x} as well as the associate covariance matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{rr} & \mathbf{P}_{rm} \\ \mathbf{P}_{rm}^T & \mathbf{P}_{mm} \end{bmatrix}. \quad (4)$$

Suppose at step k , the estimation of the states of the robot and landmarks is $\hat{\mathbf{x}}_{k|k}$ and the relevant covariance matrix is $\mathbf{P}_{k|k}$. The Kalman Filter algorithm proceeds recursively in two stages:

(1) Predict the current states of the robot and the landmarks $\hat{\mathbf{x}}_{k+1|k}$ using the process model, also compute the state estimate covariance matrix $\mathbf{P}_{k+1|k}$:

$$\begin{aligned} \hat{\mathbf{x}}_{k+1|k} &= \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{0}) \\ \mathbf{P}_{k+1|k} &= \mathbf{F}_x \mathbf{P}_{k|k} \mathbf{F}_x^T + \mathbf{F}_{d_x} \Sigma \mathbf{F}_{d_x}^T, \end{aligned} \quad (5)$$

where $\mathbf{F}_x, \mathbf{F}_{d_x}$ are the Jacobians of \mathbf{F} with respect to \mathbf{x}, \mathbf{d}_x evaluated at $\hat{\mathbf{x}}_{k|k}$, respectively.

(2) Update the estimation using the observation \mathbf{z}_{k+1} :

$$\begin{aligned} \hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}(\mathbf{z}_{k+1} - \mathbf{H}(\hat{\mathbf{x}}_{k+1|k})), \\ \mathbf{P}_{k+1|k+1} &= \mathbf{P}_{k+1|k} - \mathbf{K}_{k+1} \mathbf{S}_{k+1} \mathbf{K}_{k+1}^T, \end{aligned} \quad (6)$$

where

$$\begin{aligned} \mathbf{K}_{k+1} &= \mathbf{P}_{k+1|k} \mathbf{H}_x^T \mathbf{S}_{k+1}^{-1}, \\ \mathbf{S}_{k+1} &= \mathbf{H}_x \mathbf{P}_{k+1|k} \mathbf{H}_x^T + \mathbf{R}, \end{aligned} \quad (7)$$

and \mathbf{H}_x is the Jacobian of \mathbf{H} with respect to \mathbf{x} evaluated at $\hat{\mathbf{x}}_{k+1|k}$.

B. The minimal time motion control problem in SLAM - Motivation

In SLAM, the more observations are made, the more accurate the map and the localization will be. Also, the faster the robot moves, the larger the error in the prediction becomes. In practice, we will need the robot to complete the SLAM task in a certain period of time. So there is always a compromise between the estimation accuracy and the time used.

In many situations, we may want the robot to localize itself and complete the map of an environment (both with a certain level of accuracy) using minimal time. This can be regarded as the **Minimal Time SLAM Problem**.

The general *Minimal Time SLAM Problem* is probably intractable due to the high uncertainty of the environment and the high complexity of the problem. This motivates us to consider a more concrete problem where the destination of the robot is given. This problem may be a real navigation problem in which the destination is a particular point of interest. It may also be possible to divide a general SLAM task into several subtasks where the destination is a prescribed location in each subtasks.

There is still a big difficulty involved – obstacle avoidance. Since there are good path planning methods available (e.g. A^* Algorithm in [7], D^* Algorithm in [9]), we decouple the obstacle avoidance from the motion control as follows.

At the very beginning, a path from the starting point to the destination is computed based on all the available knowledge about the environment, this path may be renewed when some new information about the environment is available. So at any time, a path from the robot to the destination is available and our motion control problem is simplified to how to choose the direction (forward or backward) and the speed of the robot (along the path). We would like to provide a control law which only depends on the current estimation error and the current distance to the destination. So the path replanning does not matter in our motion control method.

Because we decouple the motion control into path planning and motion control along the planned path, the computation complexity is reduced, however, the result we obtained will be only sub-optimal.

At any time step, when the control action is to be decided, there is always a trade off between reducing the estimation error and reducing the distance to the destination. We would like to focus on the long term strategy instead of the local strategy.

Based on the above motivation, we formulate the problem considered in this paper as follows.

Problem: Given the starting point and the destination point and a path between the two points, find the right control policy (the direction and the velocity at each time step) such that the total time taken to reach the destination is minimized, while the error of the estimation of the robot and landmark locations is kept below a prescribed level.

C. The minimal time motion control problem in SLAM – Problem Restatement

In the following, we use some mathematical notations to state the problem more rigorously.

We suppose the control actions can be chosen from a given **set of possible control actions** – \mathbf{U} . Without loss of generality, we assume \mathbf{U} is a finite set, i.e.

$$\mathbf{U} = \{\bar{u}^1, \bar{u}^2, \dots, \bar{u}^m\}, \quad (8)$$

where $\bar{u}^1, \dots, \bar{u}^m$ are all the possible control actions. Here a control action contains not only the velocity and the direction, but also the time period the control action lasts. For example, one of the control action may be “go forward at 5 cm/s for 15 seconds”, another control action may be “rotate 180° at 5°/s” (takes 36 seconds). Since each control action lasts for a relatively long period, the effects of the speeding up or slowing down period are ignored here.

For each possible control action \bar{u}^i , $1 \leq i \leq m$, we use $T(\bar{u}^i)$ to denote the **time needed to finish the control action** \bar{u}^i . In the following, we use time steps to denote the time when the previous control action is finished and a (possibly) new control action is needed to be chosen. Note that this time step is different from the time step used in Section II-A. The actual time difference between two adjacent time steps depends on the time needed to finish the particular control action. For example, if the control action chosen at time step k is “rotate 180° at 5°/s”, then the time period between time step k and time step $k+1$ is 36 seconds. The time at which the first control action is chosen is time step 0.

In order to obtain an acceptable map in the SLAM problem, we need to keep the **error bound** of the estimation of the robot and landmarks locations less than a given positive number δ at all the time steps. We use p_k to denote this error bound at time step k . At the starting point, the error bound is denoted as p_0 . So the constraints in our control problem are

$$p_k \leq \delta, \quad \forall k \geq 0.$$

Another important factor is the **distance from the robot location to the destination**. We use d_k to denote the distance at time step k . Then d_0 is the distance from the starting point to the destination.

Now our minimal time robot control problem can be stated as follows.

Problem Restatement: Given the set of the possible control actions \mathbf{U} , the required bound of the estimation error δ , the initial error bound p_0 , and the distance from

the starting point to the ending point d_0 , find the optimal control action at each step

$$u_k \in \mathbf{U}, \quad k \geq 0,$$

such that the total time taken to reach the destination is minimized under the error bound constraints $p_k \leq \delta$, $k \geq 0$.

Remark 2.1: There are many choices for the error bound p_k . For example, the upper bound of the variance of the states of the robot (the upper bound of the diagonal elements in the submatrix \mathbf{P}_{rr} in (4)), the upper bound of the diagonal elements in the whole covariance matrix \mathbf{P} , the determinant of the covariance matrix \mathbf{P} , the Fisher information [4], etc. Our method provided here is suitable for the problems using each of these error bounds.

III. THE SOLUTION TO THE PROBLEM

A. The approximate model

From the SLAM algorithm in Section II-A, it can be seen that the evolution of the covariance matrix \mathbf{P} depends on the model of the robot, the model of the observation, the previous covariance matrix, the control action, the number and the distribution of the landmarks, and so on. The evolution of the error bound p_k is even more complicated and it is impossible to obtain a clear formula for it in general. Furthermore, even if the clear formula can somehow be obtained, it is impossible to use the model to calculate the optimal control law because the computation complexity is not acceptable.

Fortunately, we can keep the data of the error bounds at each time step during the application of SLAM algorithm. Under the assumption that the key factors of the environment (e.g. the density of the beacons) do not change very much along the path, we can assume that the error bound p_{k+1} mainly depends on the control action u_k and the previous error bound p_k . Moreover, numerical methods (e.g. the nonlinear regression methods [2]) can be used to obtain an approximate model of the evolution of the error bound (under each control action) using the data from SLAM algorithm. We assume the model of the error bound evolution is the following.

Model of p_k : For $0 \leq i \leq m$, when the control action is \bar{u}^i , the dynamics of p_k is

$$p_{k+1} = f_i(p_k) + w_i, \quad (p_0 = p_0) \quad (9)$$

where f_i are given (one scalar variable, nonlinear) functions, w_i is used to characterize the model uncertainty which satisfies

$$-w_{i0} \leq w_i \leq w_{i0} \quad (10)$$

(w_{i0} is a nonnegative constant number).

Since d_k is the distance from the robot location to the destination at time k , the dynamics of d_k can also be obtained easily for different control actions (velocity and direction).

Model of d_k : For $0 \leq i \leq m$, when the control action is \bar{u}^i , the dynamics of d_k is

$$d_{k+1} = g_i(d_k), \quad (d_0 = d_0) \quad (11)$$

where g_i are given functions.

B. Dynamic Programming Equation

We have shown that the problem is a time optimal control problem with state constraints $p_k \leq \delta$, $k \geq 0$. Now we use Dynamic Programming technique to solve the problem.

We define the **value function**

$V(p, d)$ – the minimal time taken to reach the destination when the current error bound is p , and the current distance from the robot location to the destination is d , under the constraints $p_k \leq \delta$, $k \geq 0$. (12)

In fact, the value of V at the point $(p_0, d_0) - V(p_0, d_0)$, is the minimal time needed to reach the destination from the starting point with the initial error bound p_0 (considering the worst case of the uncertainty w_i).

If the robot is currently at the ending point, then the minimal time needed to reach the ending point is 0. So the value function V needs to satisfy the boundary condition

$$V(p, 0) = 0, \forall p \leq \delta. \quad (13)$$

Note that because of the constraints $p_k \leq \delta$, when the current error bound is p_k , the control actions resulting in $p_{k+1} > \delta$ are not allowed actions. At time step k , we are only allowed to use the control actions such that $p_{k+1} \leq \delta$. So we define the **set of admissible control actions** by

$$\mathbf{U}(p_k) = \{\bar{u}^i \in \mathbf{U} : f_i(p_k) + w_i \leq \delta, \forall -w_{i0} \leq w_i \leq w_{i0}\}. \quad (14)$$

By the typical dynamic programming principle, the value function satisfies

$$V(p_k, d_k) = \min_{u_k \in \mathbf{U}(p_k)} \max_{w_k} \{T(u_k) + V(p_{k+1}, d_{k+1})\}. \quad (15)$$

Using the models of p_k and d_k , we can obtain the **Dynamic Programming Equation**

$$V(p, d) = \min_{\bar{u}^i \in \mathbf{U}(p) - w_{i0} \leq w_i \leq w_{i0}} \max_{\{T(\bar{u}^i) + V(f_i(p) + w_i, g_i(d))\}}. \quad (16)$$

C. Computation of the value function

The value function $V(p, d)$ and the optimal control action can be solved numerically using the numerical method of solving dynamic programming equations. Notice that the value function is a function of 2 scalar variables, the computation complexity is acceptable.

We only need to solve the value function within the following range: $0 \leq p \leq \delta$, $0 \leq d \leq d_0$.

The following is an algorithm that can be used to compute the value function and the optimal control action.

Algorithm:

- Step 1. Let $n = 0$. Choose a small tolerance $\epsilon > 0$.
- Step 2. Initialize the value function $V_0(p, d)$, e.g. let $V_0(p, d) = 0$ for all $0 \leq p \leq \delta$, $0 \leq d \leq d_0$. (V_0 needs to satisfy the boundary condition $V(p, 0) = 0$, $\forall p$)

- Step 3. Compute function V_{n+1} using the dynamic programming recursion

$$V_{n+1}(p, d) = \min_{\bar{u}^i \in \mathbf{U}(p) - w_{i0} \leq w_i \leq w_{i0}} \max_{\{T(\bar{u}^i) + V_n(f_i(p) + w_i, g_i(d))\}}. \quad (17)$$

- Step 4. If there exists a pair p, d , $0 \leq p \leq \delta$, $0 \leq d \leq d_0$, such that

$$|V_{n+1}(p, d) - V_n(p, d)| > \epsilon, \quad (18)$$

then let $n = n + 1$, go to Step 3. If not, $V_{n+1}(p, d)$ is an approximation of the value function that satisfies the dynamic programming equation (16).

- Step 5. The optimal controller $u^*(p, d)$ can be obtained simultaneously in the iteration. We just need to save the \bar{u}^i that achieve the minimum in (17).

After obtaining the value function $V(p, d)$ and the optimal control law $u^*(p, d)$ (off line), in the real implementation, we compute the p_k (using SLAM algorithm) and d_k (using (11)), and apply the control law $u^*(p_k, d_k)$ at each time step k .

IV. AN EXAMPLE

A. The problem and the model

An example, we consider a corridor environment (Level 5, Building 2, UTS, Figure 1 (a)). 21 beacons are placed along the corridor with uniform distribution. The distance from the starting point to the ending point is 26 meters.

We use the upper bound of the diagonal elements in the submatrix \mathbf{P}_{rr} in (4) as the error bound p_k . It should be noted that the robot can not revisit the old beacons behind it unless it rotates. So this error bound will be increasing continuously if the robot keeps going forwards (even if with a very slow speed). In order to guarantee the error bound in the SLAM is within a given bound, we have to make the robot rotate and go backward from time to time. This makes the total time to reach the end of the corridor very long. Also, it is crucial to decide when the robot rotates and how fast the robot moves.

The control action in this situation is the speed and the direction in which the robot moves. In particular, we consider the set of possible control actions

$$\mathbf{U} = \{\bar{u}^1, \bar{u}^2, \dots, \bar{u}^9\} \quad (19)$$

where

- \bar{u}^1 – go forward at 5cm/s for 15 seconds
- \bar{u}^2 – go forward at 7.5cm/s for 15 seconds
- \bar{u}^3 – go forward at 10cm/s for 15 seconds
- \bar{u}^4 – go forward at 15cm/s for 15 seconds
- \bar{u}^5 – go backward at 5cm/s for 15 seconds
- \bar{u}^6 – go backward at 7.5cm/s for 15 seconds
- \bar{u}^7 – go backward at 10cm/s for 15 seconds
- \bar{u}^8 – go backward at 15cm/s for 15 seconds
- \bar{u}^9 – rotate 180° at 5°/s (takes 36 seconds)

So, the time needed to finish the control actions are

$$T(\bar{u}^i) = 15s, \quad i = 1, \dots, 8, \quad T(\bar{u}^9) = 36s. \quad (20)$$

The robot can not go backward if it is facing forward; and it can not go forward if it is facing backward. So the set of admissible control actions depends on not only the error bound p , but also the direction of the robot. In order to take this into account, we introduce a sign for the error bound variable p , $p \geq 0$ indicates robot is facing forward and $p < 0$ indicates robot is facing backward. Now the error bound constraints become

$$|p_k| \leq \delta, \quad \forall k \geq 0.$$

Using the data obtained from the SLAM process (simulation) when the robot is moving at different speeds (and rotating), we get the approximate model of the dynamics of p by nonlinear regression methods [2] (we assume $f_i(p)$ is of the form $p + \alpha_i |p|^{\beta_i}$ when $|p| \geq 0.0005$, for $i = 1, \dots, 8$, we also assume $w_{i0} = 0$, for $i = 1, \dots, 9$)

$$\begin{aligned} f_1(p) &= \begin{cases} p + 0.0061|p|^{0.47}, & p \geq 0.0005 \\ 0.0005, & 0 \leq p < 0.0005 \end{cases} \\ f_2(p) &= \begin{cases} p + 0.0098|p|^{0.49}, & p \geq 0.0005 \\ 0.0005, & 0 \leq p < 0.0005 \end{cases} \\ f_3(p) &= \begin{cases} p + 0.013|p|^{0.47}, & p \geq 0.0005 \\ 0.0005, & 0 \leq p < 0.0005 \end{cases} \\ f_4(p) &= \begin{cases} p + 0.0205|p|^{0.47}, & p \geq 0.0005 \\ 0.0005, & 0 \leq p < 0.0005 \end{cases} \\ f_5(p) &= \begin{cases} p + 0.0092|p|^{0.56}, & p \leq -0.0005 \\ -0.0005, & 0 \geq p > -0.0005 \end{cases} \\ f_6(p) &= \begin{cases} p + 0.0175|p|^{0.6}, & p \leq -0.0005 \\ -0.0005, & 0 \geq p > -0.0005 \end{cases} \\ f_7(p) &= \begin{cases} p + 0.0187|p|^{0.56}, & p \leq -0.0005 \\ -0.0005, & 0 \geq p > -0.0005 \end{cases} \\ f_8(p) &= \begin{cases} p + 0.03025|p|^{0.58}, & p \leq -0.0005 \\ -0.0005, & 0 \geq p > -0.0005 \end{cases} \\ f_9(p) &= -p. \end{aligned}$$

Since d is the distance from the robot to the destination, the dynamics of d can be obtained easily as

$$\begin{aligned} g_1(d) &= d - 0.05 \times 15 = d - 0.75 \\ g_2(d) &= d - 0.075 \times 15 = d - 1.125 \\ g_3(d) &= d - 0.1 \times 15 = d - 1.5 \\ g_4(d) &= d - 0.15 \times 15 = d - 2.25 \\ g_5(d) &= d + 0.05 \times 15 = d + 0.75 \\ g_6(d) &= d + 0.075 \times 15 = d + 1.125 \\ g_7(d) &= d + 0.1 \times 15 = d + 1.5 \\ g_8(d) &= d + 0.15 \times 15 = d + 2.25 \\ g_9(d) &= d. \end{aligned} \quad (21)$$

When $p \geq 0$, the set of the admissible control actions is

$$\mathbf{U}(p) = \{\bar{u}^i \in \{\bar{u}^1, \bar{u}^2, \bar{u}^3, \bar{u}^4, \bar{u}^9\} : |f_i(p)| \leq \delta\}. \quad (22)$$

When $p < 0$, the set of the admissible control actions is

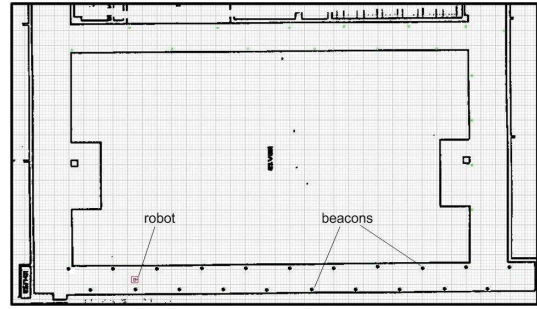
$$\mathbf{U}(p) = \{\bar{u}^i \in \{\bar{u}^5, \bar{u}^6, \bar{u}^7, \bar{u}^8, \bar{u}^9\} : |f_i(p)| \leq \delta\}. \quad (23)$$

B. Compute the value function and optimal control law

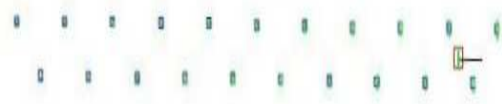
Now we compute the value function using the algorithm provided in Section III-C.

We set

$$\delta = 0.015, \quad d_0 = 26(m), \quad p_0 = 0. \quad (24)$$



(a) The environment – corridor



(b) Map obtained by SLAM algorithm

Fig. 1. The environment and the map obtained

The range of computation is: $-\delta \leq p \leq \delta, 0 \leq d \leq d_0$. We choose the number of grids $N = 100$, discretize $p \in [-\delta, \delta], d \in [0, d_0]$ as $p^1, \dots, p^N, d^1, \dots, d^N$.

Using the algorithm in Section III-C (we choose the tolerance $\epsilon = 0.1$ in Step 1), we obtain the value function $V(p_i, d_j)$, $1 \leq i, j \leq N$ and the optimal control law $u^*(p_i, d_j)$, $1 \leq i, j \leq N$ numerically as shown in Figure 2 (it takes 14 iterations for the value function to converge). It can be seen from Figure 2 that when the distance d is fixed, the larger the error bound p is, the longer time is needed to reach the destination; when the error bound p is fixed, the longer the distance d is, the longer time is needed to reach the destination.

C. Simulation

We did the simulation with a real-time SLAM algorithm realization using player/stage. Figure 1 (a) is the world map representing the corridor. A laser scanner was used which makes observations every 0.3 second (the range of the laser is 20 m and the scanning angel is 180° with resolution of 0.5 degree).

The optimal control law obtained in Section IV-B was implemented in the SLAM simulation. At each step k , the error bound p_k was obtained from the SLAM algorithm, the distance d_k was obtained using (11), and the optimal control law $u^*(p_k, d_k)$ was applied.

It took 1220s for the robot to reach the destination, the final map is given in Figure 1 (b). (The error bound is required to be within 0.015). The difference between the estimation of the robot location and the true robot location (obtained by truthproxy) is given in Figure 3.

Before the minimal time robot motion control technique was developed, in order to guarantee that the error bound in the SLAM is lower than the bound 0.015, we fixed the robot speed at 15 cm/s, and let the robot rotate and go backward once the error was above 0.015, then let it rotate again and go forward when the error bound was reduced

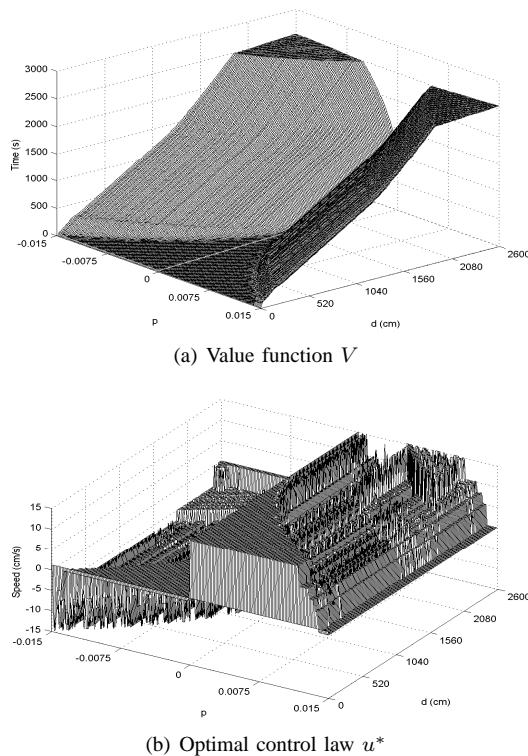


Fig. 2. Value function and Optimal control law

to 0.009. The simulation shows that the robot took longer time (1409s) to complete the SLAM task using this method than using the minimal time optimal control law.

V. CONCLUSION AND FURTHER REMARKS

In this paper, the problem of minimal time robot motion control in SLAM is considered where the robot has an destination point. An approximate one dimensional model of the evolution of estimation error is used and a global sub-optimal control law is developed using dynamic programming techniques.

In the case when the environment changes significantly, one approximate model for the evolution of the error bound p may not be enough though there is an uncertainty term w_i . One possible way of dealing with this situation is to obtain several different one dimensional models (with uncertainties) by doing experiments in different environments, and then compute the optimal control laws for the different models (offline). In real application, the obtained estimation error data in SLAM algorithm will be used to match the different models from time to time, and the control law of the matched model will be used as the real control law.

The method provided in this paper is suitable for any reliable SLAM algorithm because the approximate one dimensional model is obtained using the data from the SLAM algorithm. It is also possible to apply our method to some other problems (not only SLAM) in robot activities, such as robot information gathering, navigation and exploration, etc. The only thing needed is to change the error bound

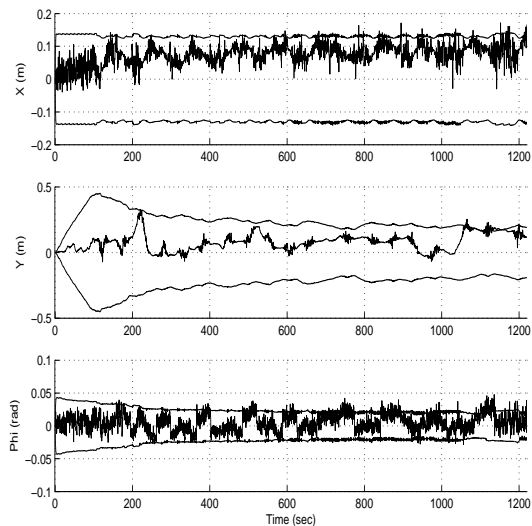


Fig. 3. The estimation error of the robot location (x, y, ϕ) and the 95% confidence limit

variable p in our method into some other performance criteria.

Further research work includes how to divide a general SLAM problem into several steps where each step has a destination point, and how to couple the obstacle avoidance with the motion control techniques to obtain less conservative results, etc.

ACKNOWLEDGMENT

This work is supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

REFERENCES

- [1] A. A. Makarenko, S.B. Williams, F. Bourgault, and H. F. Durrant-Whyte, "An Experiment in Integrated Exploration". 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2-4 October 2002, Lausanne, Switzerland.
- [2] D. M. Bates and D.G. Watts, "Nonlinear Regression Analysis and Its Applications", John Wiley & Sons, 1988.
- [3] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csobor, "A solution to the simultaneous localization and map building (SLAM) problem", *IEEE Trans. on Robotics and Automation*, vol. 17, no.3, pp. 229-241, 2001.
- [4] H. Feder, J. Leonard, and C. Smith, "Adaptive mobile robot navigation and mapping", *International J. of Robotics Research*, vol. 18, no. 7, pp. 650-668, 1999.
- [5] D. Fox, W. Burgard, and S. Thrun, "Active Markov localization for mobile robots", *Robotics and Autonomous Systems*, vol. 25, pp. 195-207, 1998.
- [6] F. Michaud, and M.J. Mataric, "Learning from history for adaptive mobile robot control", *Proc. of the 1998 IEEE/RSJ Int. Conference on Intelligent Robotics and Systems*, pp. 1865-1870, Victoria, B.C., Canada, October 1998.
- [7] N. J. Nilsson, "Principles of Artificial Intelligence", Tioga Publishing Company, 1980.
- [8] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics", in *Autonomous Robot Vehicles*, I.J. Cox and G.T. Wilfon, Eds, New York: Springer Verlag, 1990, pp. 167-193.
- [9] A. Stentz, "Optimal and Efficient Path Planning for Partially-Known Environment". In Proceedings IEEE International Conference on Robotics and Automation. 1994.
- [10] B. Yamauchi, "A frontier based approach for autonomous exploration". In IEEE International Symposium on Computational Intelligence in Robotics and Automation. Monterey, CA, USA, 1997.