

---

# A Markov-Chain Monte Carlo Approach to Simultaneous Localization and Mapping

---

**Péter Torma**  
GusGus AB  
Hungary

**András György**  
Machine Learning Res. Group  
MTA SZTAKI, Hungary

**Csaba Szepesvári**  
Dept. of Computing Sciences  
University of Alberta, Canada

## Abstract

A Markov-chain Monte Carlo based algorithm is provided to solve the Simultaneous localization and mapping (SLAM) problem with general dynamics and observation model under open-loop control and provided that the map-representation is finite dimensional. To our knowledge this is the first provably consistent yet (close-to) practical solution to this problem. The superiority of our algorithm over alternative SLAM algorithms is demonstrated in a difficult loop closing situation.

## 1 Introduction

Simultaneous localization and mapping (SLAM) is a well-known, intensively studied problem of robotics (Thrun et al., 2005). In the classical problem a mobile robot is moving in an unknown environment with a known control sequence. It is assumed that there are some landmark points in the environment which the robot is able to sense and that the robot follows its control with some uncertainty. The goal of the SLAM task is to incrementally build a consistent map of the robot’s environment while simultaneously determining its location within this map.

SLAM has been formulated and solved as a theoretical problem in a number of different settings (Durrant-Whyte and Bailey, 2006; Bailey and Durrant-Whyte, 2006). The two most successful approaches are based on Kalman filtering and particle filtering. As usual, the Kalman filter based approaches (such as EKF-SLAM by Cheeseman and Smith, 1986 and UKF-

SLAM by Smith et al., 1990) have the obvious drawback that they assume linear or linearized dynamics and observation model, and Gaussian noise. The other successful direction, called FastSLAM (Montemerlo et al., 2002, 2003), uses Rao-Blackwellized particle filtering. Although, for the first sight, FastSLAM provides a general, consistent solution, it actually fails to solve the problem reliably as it is discussed by (Bailey et al., 2006). Moreover, the solution given by FastSLAM is not general, as the landmark locations are represented by Gaussians in each particle.

Consequently the SLAM problem does not have a consistent and robust solution in case of arbitrary dynamics and observation model. In this paper we give a solution to this problem based on a Markov-chain Monte Carlo (see, e.g., Andrieu et al., 2003) approach under open-loop control for finite-dimensional maps.

The main challenge in SLAM is the so-called loop-closing problem. This is the situation when the robot observes a landmark that has not been seen for a long time. As the robot moves, its uncertainty in its earlier positions and the landmark locations increases. When a landmark that has been seen earlier is observed again (“a loop is closed”), in theory, this could be used to refine all previous estimates to a great extent. Assuming a prior over the landmarks, this can be made more precise: In normal steps the posterior distribution over past states and the landmarks usually gets more diffuse over time, while at loop closing suddenly it becomes very well concentrated.

As we consider general models, no closed form representation of the posterior is possible (as in the linear-Gaussian case). A standard solution then is to use particle filtering, where the underlying distribution is represented by weighted samples (Doucet et al., 2001). However, when loop closing happens, the broad, uncertain posterior that is represented by the weighted particles should be transformed into a peaky (concentrated) posterior. This is only possible if at least some particles “hit” the region of the peak. Since the state-space of the SLAM problem is huge (and grows with

time), any practical number of particles might prove to be too few. As a result, FastSLAM and other particle filter methods using a bounded number of particles is determined to fail on some SLAM problem (Bailey et al., 2006). Some improvements are achievable if the map learning is separated from the estimation of the path of the robot, as done by Martinez-Cantin et al. (2007a,b) using marginal particle filters.

Here we explore an alternative solution. A very elegant way of representing arbitrary distributions is to use the stationary distributions of stable Markov chains. The theory of Markov-chain Monte Carlo (MCMC) methods gives recipes to construct Markov chains with any desired stationary distributions (see, e.g., Andrieu et al., 2003). In this paper we will show that the solution of the general SLAM problem is possible with MCMC methods: we will construct a *provably consistent* yet (close-to) *practical* method to sample from the SLAM posterior. The main advantage of our method is that MCMC is able to efficiently track the change in the posterior even in the case of loop closing. We will argue that the special structure of the SLAM problem makes it possible to implement the algorithm efficiently.

To the best of our knowledge prior to this work MCMC has only been used for solving sub-tasks of the SLAM problem (see, e.g., Kaess and Dellaert, 2005; Ranganathan and Dellaert, 2005), but it has not been used to attack the SLAM problem itself. The idea most closely related to ours is to use the standard MCMC algorithm to sample marginals in a Bayes-net (see, e.g., Pearl, 1987). However the SLAM-problem generates a very special Bayes-net, and hence in our case a more efficient solution is possible than the one based on the standard algorithms.

The rest of the paper is organized as follows. The problem is defined in Section 2. The MCMC-SLAM algorithm is defined and analyzed in Sections 3 and 4. Efficiency issues are considered in Section 5 and experiments are reported in Section 6. The paper closes with conclusions in Section 7.

## 2 Preliminaries

Consider a dynamical system whose states  $x_0, x_1, x_2, \dots, x_n$  evolve in the  $d$ -dimensional Euclidean space  $\mathbb{R}^d$ , where the transition between  $x_t$  and  $x_{t+1}$  is defined by some conditional probability density function  $p(x_t|x_{t-1})$  for  $t = 1, \dots, n$  and (without the loss of generality)  $x_0 = 0$ .<sup>1</sup> At time  $t = 0, \dots, n$ ,  $M$  random variables  $z_t^i, i = 1, \dots, M$ , with conditional

probability density functions  $p(z_t^i|x_t, \theta_i)$  are observed. Here  $\theta_i \in \mathbb{R}^d$  are some static parameters. The variables  $z_t^i$  are assumed to be conditionally independent given the object states  $x_{0:n} = (x_0, x_1, \dots, x_n)$  and the static parameter set  $\theta_{1:M} = (\theta_1, \dots, \theta_M)$ .<sup>2</sup> The task is to estimate  $x_{0:n}$  and  $\theta_{1:M}$ .

The system defined above can be viewed as a natural generalization of the SLAM problem. A slight difference in the notation is that in the SLAM problem the state transition  $p(x_t|x_{t-1})$  is governed by a known control signal  $u_t$ , and  $p(x_t|x_{t-1}) = p(x_t|x_{t-1}, u_t)$  for some conditional density function  $p(x_t|x_{t-1}, u_t)$ . Note that this formulation implies that the number of parameters,  $M$ , to be estimated is fixed and given. Another assumption is that the control should not depend on the observations – an assumption that is typical in the system identification literature (see, e.g., Katayama, 2006) but is admittedly restrictive. Extensions of this work to the cases not covered by these assumptions are left for future work.

In this paper we take the Bayesian approach to system identification: we assume that a known prior distribution  $p(\theta_{1:M})$  is available which  $\theta_{1:M}$  is sampled from before the state of the system is reset to  $x_0$ . The problem is to determine, or just sample from the *posterior density*  $p(x_{0:n}, \theta_{1:M}|z_{0:n})$  of the past state variables and the parameters. The next statement gives a trivial formula for the posterior. Given the assumptions, the proof is trivial and is hence omitted.

**Proposition 1.** *The posterior density of  $x_{0:n}$  and  $\theta_{1:M}$  given the observations  $z_{0:n}$  can be computed as*

$$p(x_{0:n}, \theta_{1:M}|z_{0:n}) = C(z_{0:n})p(\theta_{1:M}) \left( \prod_{t=0}^{n-1} p(x_{t+1}|x_t) \right) \prod_{t=0}^n \prod_{i=1}^M p(z_t^i|x_t, \theta_i)$$

where  $C(z_{0:n})$  is a normalizing constant that depends on  $z_{0:n}$  only.

To sample from the posterior, we transform the problem to what we call *an inference graph problem*.

## 3 Inference graphs and the generalized SLAM problem

Consider a graph  $G = (V, E)$ , where  $V$  is the set of vertices and  $E$  is the set of edges. Assume that each vertex  $v \in V$  is labeled with a  $k_1$ -dimensional real vector  $l(v) \in \mathcal{V} \subset \mathbb{R}^{k_1}$  and also assume that for each edge  $e \in E$  there is a labeling function  $s_e : \mathcal{V} \times \mathcal{V} \rightarrow$

pressed in the notation  $p(x_t|x_{t-1})$ .

<sup>2</sup>Throughout the paper we use the notation  $a_{k:l} = (a_k, a_{k+1}, \dots, a_l)$  for any sequence.

<sup>1</sup>To simplify the notation we will denote all densities with  $p$ . The actual random variables  $p$  corresponds to will be clear from the arguments of  $p$ . Also, controls are sup-

$\mathbb{R}^{k_2}$ , and a probability density function  $J_e$  on the image space of  $s_e$ . In what follows, for simplicity, and without loss of generality, we assume that  $k = k_1 = k_2$ .<sup>3</sup> Then  $G$  equipped with  $l$ ,  $(s_e)_{e \in E}$ , and  $(J_e)_{e \in E}$  is called an *inference graph*.

We will assume that the edge labeling functions are invertible in both of their parameters, that is, for each  $e \in E$  and vertex labels  $l_1, l_2 \in \mathcal{V}$ , there are functions  $s_{e,1}^{-1}$  and  $s_{e,2}^{-1}$  such that  $s_{e,1}^{-1}(s_e(l_1, l_2), l_2) = l_1$  and  $s_{e,2}^{-1}(l_1, s_e(l_1, l_2)) = l_2$ .

The vertex labeling function  $l$  defines a labeling on the edges which, by slightly abusing notation, we shall also denote by  $l$ :  $l(e) = s_e(l(v), l(v'))$  for  $e = (v, v')$ . This shows how to go from node labels to edge labels. We can also ask if the reverse is possible. That is, given arbitrary labels  $l(e), e \in E$ , is there a vertex labeling function  $l$  such that  $l(e) = s_e(l(v), l(v'))$  for all  $e \in E$  where  $e = (v, v')$ ? If this holds, we call the edge labeling *consistent*.

In what follows we assume that the label of an arbitrary fixed vertex  $v_0$  is set to the zero vector; then the vertex and edge labels uniquely determine each other (provided that the edge labels are consistent) because of the invertibility of the edge labeling functions  $s_e$ .

The *posterior density* of a consistent (edge/vertex) labeling  $l$  on the inference graph  $G$  is defined as

$$\begin{aligned} \pi(l) &= p(l(e), e \in E) = p(l(v), v \in V) \\ &= C \prod_{e=(v,v') \in E} J_e(s_e(l(v), l(v'))) \end{aligned} \quad (1)$$

where  $C$  is a suitable normalizing constant.

The inference graph sampling problem is to sample consistent edge labels from  $\pi$ . Notice that the set of consistent edge labels is a low-dimensional manifold  $\mathcal{M}$  in  $\mathbb{R}^{k \times |E|}$  specified by the consistency constraints. This makes sampling from  $\pi$  difficult as the sampler must stay on the manifold  $\mathcal{M}$ . Before deriving the sampling procedure, we show how to transform the SLAM problem to such an inference graph problem.

### 3.1 The SLAM inference graph

We define an inference graph  $G$  for the parameter estimation problem considered in Section 2. For each state  $x_t, t = 0, \dots, n$  and parameter  $\theta_i, i = 1, \dots, M$  we define a vertex in the graph: vertices corresponding to states (parameters) will be denoted by  $v_{x,t}$  (resp., by  $v_{\theta,i}$ ). Two vertices are connected by an edge if their underlying variables are connected via some con-

ditional distribution in the definition of the system. That is, the edges of  $G$  are defined by

$$\begin{aligned} E &= \{(v_{x,t-1}, v_{x,t}), t = 1, \dots, n\} \\ &\cup \{(v_{x,t}, v_{\theta,i}), t = 0, \dots, n, i = 1, \dots, M\}. \end{aligned}$$

Note that in this way all states are connected to all parameters (as needed for an unambiguous definition of the posterior). What happens when in a SLAM problem some landmarks ( $\theta_k$ ) are not “visible” in certain states? To address this, the observation space would contain a special signal that indicates “no information” so that all landmarks can generate some observation in all time steps. Note that the graph grows with  $n$ : at each time step a new vertex and  $M+1$  new edges are introduced. However, for simplicity, here we concentrate on the case when  $n$  is fixed; issues related to incremental implementation of our algorithm will be discussed in Section 5.2.

The vertex labels are defined as  $l(v_{x,t}) = x_t$  and  $l(v_{\theta,i}) = \theta_i$ . Furthermore, we assume that the edges have a common labeling function  $s$ , that is, an edge  $(v, v') \in E$  has label  $l(e) = s(l(v), l(v'))$ : for an edge  $(v_{x,t-1}, v_{x,t})$  the label is  $s(x_{t-1}, x_t)$ , and for an edge  $(v_{x,t}, \theta_i)$  the label is  $s(x_t, \theta_i)$ . Also recall that  $s$  has to be invertible in both of its variables.

We assume that the conditional distributions  $p(x_t|x_{t-1})$  and  $p(z_t^i|x_t, \theta_i)$  depend on  $(x_{t-1}, x_t)$  and  $(x_t, \theta_i)$  via  $s$  only:

**Assumption 1.** *There exist functions  $K_t$  such that*

$$K_t(s(x_{t-1}, x_t)) = p(x_t|x_{t-1}),$$

*and there exist functions  $r_t$  such that*

$$r_t(z_t^i|s(x_t, \theta_i)) = p(z_t^i|x_t, \theta_i).$$

Finally, we need to define the density functions  $J_e$  for each edge. Let

$$J_e(s) = \begin{cases} \frac{K_t(s)}{\int K_t(s') ds'}, & \text{if } e = (v_{t-1}, v_t); \\ \frac{r_t(z_t^i|s)}{\int r_t(z_t^i|s') ds'}, & \text{if } e = (v_t, \theta_i), \end{cases}$$

where we assume that the integrals above are finite.<sup>4</sup>

The next statement shows the connection between the posterior densities of the labeling on the graph  $G$  and the original problem.

**Proposition 2.** *Under Assumption 1 the posterior density of the dynamical system is given by*

$$p(x_{0:n}, \theta_{1:M}|z_{0:n}) = C_1(z_{0:n})p(\theta_{1:M})\pi(l)$$

<sup>3</sup>Our results would still apply even if the dimensions of the label spaces were dependent on the edge/vertex labeled.

<sup>4</sup>Note that the general MCMC-SLAM algorithm to be defined later only needs proportions of the form  $J_e(s)/J_e(s')$ , hence computing the normalizing integrals is unnecessary.

where

$$\pi(l) = C_2(z_{0:n}) \prod_{e=(v,v') \in E} J_e(s(l(v), l(v')))$$

and  $C_1(z_{0:n})$  and  $C_2(z_{0:n})$  are suitable normalizing constants that depend on  $z_{0:n}$  only.

*Proof.* First notice that since  $s$  is invertible and  $x_0 = 0$ , the edge labels  $\{l(e), e \in E\}$  and the variables  $(x_{0:n}, \theta_{1:M})$  uniquely determine each other. Now, it follows from Proposition 1 and Assumption 1 that

$$\begin{aligned} p(x_{0:n}, \theta_{1:M} | z_{0:n}) &= C(z_{0:n}) p(\theta_{1:M}) \cdot \\ &\cdot \left( \prod_{t=0}^{n-1} K(s(x_{t+1}, x_t)) \right) \prod_{t=0}^n \prod_{i=1}^M r(z_t^i | s(x_t, \theta_i)) \\ &= C_1(z_{0:n}) p(\theta_{1:M}) \pi(l) \end{aligned}$$

for some constant  $C_1(z_{0:n})$ .  $\square$

It follows from the above result that if we can sample efficiently from the posterior underlying the inference graph then we can sample from the posterior by accept-reject sampling (Andrieu et al., 2003) provided the density of the prior distribution  $p(\theta_{1:M})$  is bounded by some constant  $L > 0$ . That is, first we sample an edge labeling  $l$  from  $\pi(l)$ , and determine  $x_{0:n}$  and  $\theta_{1:M}$  from  $l$ . Then  $(x_{0:n}, \theta_{1:M})$  is accepted with probability  $p(\theta_{1:M})/L$ . The following theorem is a trivial consequence of Proposition 2 and the standard result on accept-reject sampling (Andrieu et al., 2003).

**Theorem 1.** *Assume there is an  $L > 0$  such that  $p(\theta_{1:M}) \leq L$  for all possible values of  $\theta_{1:M}$ . Then in the above sampling procedure,  $(x_{0:n}, \theta_{1:M})$  follows the posterior distribution  $p(x_{0:n} = \cdot, \theta_{1:M} = \cdot | z_{0:n})$ .*

Note that if  $p(\theta_{1:M})$  is uniform on its support then the accept-reject procedure always accepts (with the choice of  $L \equiv p(\theta_{1:M})$ ). Hence our goal is to sample from  $\pi(\cdot)$ . We use MCMC to do so.

## 4 MCMC for inference graphs

MCMC is a strategy for generating samples from a distribution that is hard to sample from otherwise. The idea is to construct an irreducible and aperiodic Markov chain with kernel  $Q$  that has the desired density  $p$  as stationary density. This is easy to achieve if the Markov transition kernel  $Q$  satisfies the detailed balance equation

$$p(y')Q(y|y') = p(y)Q(y'|y).$$

It is the spectral structure of  $Q$  that determines the convergence rate of the resulting chain (Andrieu et al., 2003). Many MCMC algorithms of various character are known in the literature. In this paper we use

the Metropolis-Hastings algorithm. The algorithm is parametrized by a transition kernel  $Q(y'|y)$ , called the *proposal distribution*. Starting from an arbitrary state  $y_0$  the algorithm generates a sequence  $\{y_\tau\}$  as follows:

1. At time  $\tau$  generate a new sample  $y$  according to the conditional distribution  $Q(y|y_{\tau-1})$ .
2. Accept the move with probability

$$A(y|y_{\tau-1}) = \min \left\{ 1, \frac{p(y)Q(y_{\tau-1}|y)}{p(y_{\tau-1})Q(y|y_{\tau-1})} \right\},$$

that is, in this case let  $y_\tau = y$ , otherwise let  $y_\tau = y_{\tau-1}$ .

Under very general conditions on  $Q$  the generated sequence will converge, in distribution, to  $p$  (Tierney, 1998).

In what follows we will show how to construct a Metropolis-Hastings algorithm to sample labelings from the posterior density underlying a general inference graph.

As we mentioned earlier, sampling a labeling from the posterior (1) of an inference graph  $G$  is hard because the densities are defined in terms of the edges while the consistency of a labeling is defined via the vertices.

To address this issue we propose the following procedure. Fix a spanning tree  $T$  of  $G$ . Next:

1. randomly choose an edge  $e$  of  $T$ ;
2. choose a new label for this edge from some distribution;
3. change the vertex labels on  $T$  to make the labeling consistent while keeping all edge labels except for that of the chosen edge  $e$ ;
4. adjust the edge labels on the edges outside of  $T$  to make the labeling consistent on  $G$ .

It is easy to see that almost any edge labeling is consistent on a tree (the only constraint is that if the labels of two vertices are equal then the edges that connect these vertices to a third vertex must have the same label), and any consistent labeling on a spanning tree  $T$  uniquely determines a consistent labeling on  $G$ .

When the label of an edge  $e$  of the spanning tree  $T$  is changed, we make the labeling consistent in the tree in the following way: Let  $T_{e,1}$  and  $T_{e,2}$  be the two subtrees obtained by deleting  $e$  from  $T$ . To make the choice unique, we assume that  $v_0 \in T_{e,1}$  (recall that the label of  $v_0$  is fixed at zero). Then, for any  $v \in T_{e,1}$  the labels are kept unchanged, that is,  $l'(v) = l(v)$ , while for  $v \in T_{e,2}$  the labels  $l(v)$  are changed to make the new labels consistent on  $T$  (that this can be done and in a unique manner follows from the fact that  $s_e$  is invertible for any  $e \in E$ ). Finally, we recompute the labels of those edges that have one vertex in  $T_{e,1}$  and another one in  $T_{e,2}$ .

Define the proposal distribution for the labeling  $l$  as

$$Q_T(l'|l) = \sum_{e \in E} p_T(e|l) Q_T(l'|l, e), \quad (2)$$

where, for all consistent  $l$ ,  $p_T(e|l)$  is a user-chosen distribution supported on the edges of  $T$ , that is,  $p_T(e|l) > 0$  if  $e \in T$  and  $p_T(e|l) = 0$  if  $e \notin T$ . Further,  $Q_T(l'|l, e)$  is defined as follows: If  $e$  is not an edge of  $T$ , then  $Q_T(l'|l, e) = 0$  for all labelings  $l, l'$ . For any edge  $e \in T$ ,  $Q_T(l'|l, e)$  is defined through the following procedure: Sample  $l'(e)$  according to the distribution  $J_e$ . For any other edge  $e' \in T$ ,  $e' \neq e$ , let  $l'(e') = l(e')$ . For any vertex  $v \in V$ , the label  $l'(v)$  is defined in the unique way that results in a consistent labeling for  $l'$  on  $T$  while keeping the edge labels and  $l'(v_0) = 0$  fixed. Note that in this way we can only change the label of a vertex  $v$  if  $v \in T_{e,2}$  and  $l'(v) = l(v)$  if  $v \in T_{e,1}$ . Finally, for any edge  $(v, v')$  let  $l'((v, v')) = s_{(v, v')}(l'(v), l'(v'))$ . For this  $l'$ ,  $Q_T(l'|l, e) = J_e(l'(e))$ .

**Theorem 2.** *Let the proposal distribution  $Q_T(l'|l)$  be defined by (2), and define the acceptance probability as*

$$A_T(l'|l) = \min \left\{ 1, \frac{\pi(l') Q_T(l|l')}{\pi(l) Q_T(l'|l)} \right\}.$$

*if  $l$  and  $l'$  differ on the edges of  $T$  in at most once, and let  $A_T(l'|l) = 0$  otherwise. Then the resulting MCMC converges to the posterior distribution  $\pi$ .*

*Proof.* We start with a few trivial observations: (i) the support of  $Q_T(\cdot|l, e)$  contains consistent labelings only; (ii) for almost all  $l'$  (with respect to  $Q_T(\cdot|l, e)$ ) there is an edge  $e_{l, l'} \in T$  (dependent on  $l$  and  $l'$ ) such that

$$Q_T(l'|l) = p_T(e_{l, l'}|l) Q_T(l'|l, e_{l, l'}); \quad (3)$$

(iii)  $Q_T(l'|l) > 0$  if and only if  $Q_T(l|l') > 0$ ; (iv) the resulting Markov chain is irreducible. Here (ii) follows since given any consistent  $l$ , for almost all  $l'$ ,  $l'$  and  $l$  differ on exactly one edge of  $T$  with probability 1. Calling this edge  $e_{l, l'}$  yields (3). Then it follows from the general state-space Metropolis-Hastings theorem<sup>5</sup> due to Tierney, 1998 (cf. Theorem 2 and the following discussion) that the resulting Markov chain converges to the stationary distribution  $\pi$ .  $\square$

## 5 Efficiency

The efficiency of this algorithm depends on several factors, such as the accept-rate (more generally, the spectral structure of the chain) and the cost of a single step.

<sup>5</sup>Note that standard textbook versions are not applicable since the support  $\mathcal{M}$  of the Markov kernel is a low-dimensional manifold.

### 5.1 Label preserving transformations

The complexity of computing the acceptance probability depends mainly on the number of likelihood evaluations needed to compute the acceptance probability. It follows from the results of Section 4 and (1) and (3) that if  $l$  and  $l'$  are two successive edge labelings and  $e_{l, l'}$  is the unique edge they differ on in  $T$  then

$$A_T(l'|l) = \min \left\{ 1, \frac{p_T(e_{l, l'}|l')}{p_T(e_{l, l'}|l)} \prod_{e \notin T} \frac{J_e(l'(e))}{J_e(l(e))} \right\}.$$

As the set of edges of  $G$  not belonging to  $T$  can be large, below we derive an alternate form to the above that needs fewer evaluations of  $J_e$ .

This is done under the assumption that the labeling functions  $s_e$  have some nice structure which allows to localize the changes in the edge labels. A function  $g: \mathcal{V} \rightarrow \mathcal{V}$  is called a *label preserving transformation* for the inference graph  $G$  equipped with  $l$  and  $\{s_e\}$  if  $s_e(l_1, l_2) = s_e(g(l_1), g(l_2))$  for any labels  $l_1, l_2 \in \mathcal{V}$  and edge  $e \in E$ .

Assume that the label  $s$  of  $e = (v, v') \in T$  is changed to  $s'$  so that the new label of  $v'$  is obtained by a label preserving transformation  $g(\cdot) = g(\cdot, s, s')$ . Then the new (consistent) labels for any  $\hat{v} \in T$  can be computed as

$$l'(\hat{v}) = \begin{cases} l(\hat{v}), & \text{if } \hat{v} \in T_{e,1}; \\ g(l(\hat{v})), & \text{if } \hat{v} \in T_{e,2}. \end{cases} \quad (4)$$

Furthermore, the label preserving condition implies that for any edge  $\hat{e} = (v, v') \in T_{e,2}$ ,

$$l'(\hat{e}) = s_{\hat{e}}(g(l(v)), g(l(v'))) = s_{\hat{e}}(l(v), l(v')) = l(\hat{e}).$$

Clearly, for  $\hat{e} \in T_{e,1}$  we have  $l'(\hat{e}) = l(\hat{e})$ . Thus, the edge labels have to be recomputed only for those edges, different to  $e$ , which have one vertex in  $T_{e,1}$  and one in  $T_{e,2}$ . Denote this set of edges by  $\hat{E}(T_e)$ . In summary, the new edge labels for all  $\hat{e} = (v, v') \neq e$  can be computed as

$$l'(\hat{e}) = \begin{cases} s_{\hat{e}}(l'(v), l'(v')), & \text{if } \hat{e} \in \hat{E}(T_e); \\ l(\hat{e}), & \text{otherwise.} \end{cases} \quad (5)$$

One step of the resulting MCMC algorithm is shown in Figure 1. It follows easily from (1) and (3) that the computational requirements of the acceptance probability are reduced significantly:

**Theorem 3.** *If the edge labeling functions  $s_e$  allow the application of label preserving transformations  $g$ , then the acceptance probability can be computed as follows:*

1. Draw randomly an edge  $e$  of the spanning tree  $T$  according to the law  $p_T(e|l)$ , and split  $T$  into the subtrees  $T_{e,1}$  and  $T_{e,2}$ .
2. Sample a new label  $l'(e)$  from  $J_e$  and find a label preserving transformation  $g$  for which  $l'(e) = s_e(v, g(v'))$  where  $v \in T_{e,1}$  and  $v' \in T_{e,2}$  are the vertices of  $e$ .
3. Update the vertex labels according to (4) and the edge labels according to (5).
4. Accept the move with probability  $A_T(l'|l)$  given by (6).
5. If the move is accepted change all  $l$  to  $l'$ .

Figure 1: One step of the efficient MCMC-SLAM algorithm with an arbitrary spanning tree  $T$  and label preserving transformations.

$A_T(l'|l) = 1$  if  $l(e) = l'(e)$  for all edges  $e \in T$ . If  $l$  and  $l'$  agree in all but one edge of  $T$ , then

$$A_T(l'|l) = \min \left\{ 1, \frac{p_T(e_{l,l'}|l')}{p_T(e_{l,l'}|l)} \prod_{e \in \hat{E}(T_{e_{l,l'}})} \frac{J_e(l'(e))}{J_e(l(e))} \right\} \quad (6)$$

where  $e_{l,l'}$  is the unique edge in  $T$  where  $l$  and  $l'$  differ.

For structured graphs containing many edges corresponding to “no information”, such as a real SLAM graph, the computational savings can be large.

## 5.2 Implementation and heuristics

As we have seen, in the simplest version of the MCMC-SLAM algorithm all vertices in  $T_{e_{l,l'},2}$  and all edges that have at least one vertex in  $T_{e_{l,l'},2}$  have to be relabeled. This is costly. If a label preserving transformation can be applied, it suffices to relabel only the vertices in  $T_{e_{l,l'},2}$  and the edges in  $\hat{E}(T_{e_{l,l'}})$  (and, obviously,  $e_{l,l'}$ ). Since  $J_e(\cdot)$  needs to be evaluated exactly for those edges that are relabeled, this way we can greatly reduce the computational needs of the algorithm. Note that label preserving is naturally satisfied in most practical SLAM problems and the gain is usually significant.

Consider now the problem when the algorithm is used not with a fixed number of observations but in an online situation when observations arrive sequentially and after each observation we need samples from the posterior. In this case the size of the SLAM graph (the number of edges) grows linearly both in time  $n$  and the number of landmarks  $M$ , and so the algorithm’s computational complexity is  $O(n^2MN)$ , where  $N$  is

the number of MCMC moves per round. In a typical SLAM graph only a bounded number of landmarks are visible from each state, and so the number of edges corresponding to “real” observations grows only as  $O(n^2)$ , hence the computational complexity becomes  $O(n^2N)$ . One idea (not explored here) is to use an adaptive number of MCMC moves or mix the algorithm with some approximate algorithm. Another idea is to store only the edge labels and compute the vertex labels only when needed. This way only  $|\hat{E}(T_{e_{l,l'}})|$  edge labels must be recomputed after each accepted MCMC move. Exploiting the problem’s geometry, usually an update can be implemented efficiently using the labels of edges in a small vicinity of the edge to be updated.

Other heuristics may help to speed up the convergence of the MCMC sampling. For example, choosing  $p_T(e|l)$  such that edges with “good” labels are picked rarely can help. One way to do this is to set  $p_T(e|l) \propto 1/J_e(l(e))$  (this is the choice used in the experiments below). The choice of the spanning tree also influences the cost, and several heuristics (not discussed here) can be applied to improve performance.

As already noted one can also combine the MCMC approach with some other “fast” ( $O(n)$ ) algorithm, like an algorithm based on particle filters. The chain’s burn-in time is expected to be short if the chain is started from the state encoded by a randomly selected particle. This idea is similar to that explored by Lee (2001).

## 6 Experiments

We tested the above MCMC sampling algorithm on a standard SLAM problem (Thrun et al., 2005): A simulated robot moves around the plane under some deterministic control and continuously observes the distances and viewing angles to some landmarks. The robot follows the control only with some uncertainty, and our goal is to estimate the robot’s trajectory, as well as the positions of the landmarks based on these observations and the control sequence.

The robot’s state is described by its position  $(x_t, y_t)$  on the plane and a heading parameter  $\varphi_t$  (all measured relative to the robot’s original pose). The robot is controlled by specifying the pair  $(v_t, \omega_t)$ . The robot’s state  $(x_t, y_t, \varphi_t)$  evolves according to

$$\begin{aligned} x_t &= x_{t-1} - \frac{v_t}{\omega_t} \sin \varphi + \frac{v_t}{\omega_t} \sin(\varphi + \omega_t \Delta t), \\ y_t &= y_{t-1} + \frac{v_t}{\omega_t} \cos \varphi - \frac{v_t}{\omega_t} \cos(\varphi + \omega_t \Delta t), \\ \varphi_t &= \varphi_{t-1} + \omega_t \Delta t + \gamma_t \Delta t, \end{aligned}$$

where  $\gamma_t$  is a robot-specific rotation-noise parameter and  $\Delta t$  is the duration of a time step.

As said before, the robot observes the distance and viewing angle of the landmarks. Let the position of the  $i$ th landmark be  $\theta_i = (\theta_{i,1}, \theta_{i,2})$ . If the robot is close to the landmark, it observes  $z_t^i = (d_t^i, \varphi_t^i)$ , where

$$\begin{aligned} d_t^i &= \sqrt{(\theta_{i,1} - x_t)^2 + (\theta_{i,2} - y_t)^2} + \epsilon_{\sigma_r}, \\ \varphi_t^i &= \arctan\left(\frac{\theta_{i,2} - y_t}{\theta_{i,1} - x_t}\right) - \varphi_t + \epsilon_{\sigma_\phi}, \end{aligned}$$

where  $\epsilon_{\sigma_r}$  and  $\epsilon_{\sigma_\phi}$  are zero-mean Gaussian random variables with respective standard deviations  $\sigma_r$  and  $\sigma_\phi$ . If the robot is far from the landmark, the robot’s observation for the  $i$ th landmark is  $z_t^i = \#$ , meaning “no-information”. The probability of no information is solely determined by the distance  $d_t^i$ .

For this problem a suitable edge labeling function is defined by  $s(x_{t+1}, x_t) = x_{t+1} - x_t$  and

$$s(x_t, \theta_i) = \begin{pmatrix} x_{t,1} - \theta_{i,1} \\ x_{t,2} - \theta_{i,2} \\ \arctan\left(\frac{\theta_{i,2} - y_t}{\theta_{i,1} - x_t}\right) - \varphi_t \end{pmatrix},$$

the latter of which gives the distance vector together with the viewing angle of the landmark position  $\theta_i$  from state  $x_t$ . It is easy to see that  $s(\cdot, \cdot)$  is invertible. Furthermore, it is easy to verify that any Euclidean transformation  $g$  is label preserving with these labeling functions.

Thus, we can apply our efficient MCMC sampling algorithm of Section 5. In the algorithm at each time instant we use a trivial spanning tree of the SLAM-graph that includes all the edges between state-nodes. Further, for each landmark node, the state-landmark edge which connects the landmark node to the state where the landmark was seen from for the first time is included.<sup>6</sup> We use  $p_T(e|l) \propto \frac{1}{J_e(l(e))}$  as discussed in Section 5, and a uniform prior for the landmark positions, obviating the need for the accept-reject step.

We compared our algorithm, MCMC-SLAM, to FastSLAM 2.0 (Montemerlo et al., 2003) and EKF-SLAM (Cheeseman and Smith, 1986) (for the latter two we used the implementations of Bailey, all algorithms run in Matlab) on two artificial examples with the above described dynamics and observation model. We used online, sequential computations (which is disadvantageous for our algorithm unless one is interested in estimating the unknowns in each time step). The first “small” problem is taken from Bailey, while in the second, “large” problem the landmarks are arranged in an

<sup>6</sup>Our choice of spanning tree is not optimal in the sense that usually the densities underlying the dynamics are less concentrated than those of the observation model. Hence guessing these links well is less important. Our choice is dictated by simplicity.

equidistant grid and the robot follows a circular path with a radius 8 times bigger than the grid parameter. MCMC-SLAM was used with 1, 10, and 100 MCMC proposals per time step, while FastSLAM 2.0 was used with 100, 1000, and 10000 particles. The average mean-squared errors of the estimated landmark positions and the cumulative state errors, together with the running times slightly after the loop closing, averaged over 100 runs, are given in Table 1. EKF-SLAM is fast, but it is highly inaccurate, in terms of both the landmark and state errors. FastSLAM 2.0 is more accurate, but its accuracy is limited and improves only moderately when run with more particles. In this example, MCMC-SLAM outperforms FastSLAM 2.0 if an accurate solution is needed and the algorithms are given enough time.

Figure 2 shows the results of a typical run of the algorithms for the “small” problem before and after a loop closing happens. Noticeable is that after loop closing our algorithm’s landmark (and trajectory) estimates are significantly more accurate than the estimates computed by the other algorithms.

## 7 Conclusions

We considered the SLAM problem with general dynamics and observation model. We gave an MCMC based algorithm, MCMC-SLAM, which, to our knowledge, is the first provably consistent algorithm with close-to practical run-time. Our experiments have shown that the new algorithm is superior than previous state-of-the-art algorithms if high accuracy estimates are needed. Problems for future work include how to improve the run-time, e.g., by combining the algorithm with particle filtering or by making the sampling part adaptive.

## Acknowledgment

This work was supported in part by the PASCAL2 Network of Excellence under EC grant no. 216886, the Hungarian Scientific Research Fund and the Hungarian National Office for Research and Technology (OTKA-NKTH CNK 77782), NSERC, the Alberta Ingenuity Centre for Machine Learning and iCore.

## References

- C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, pages 5–43, 2003.
- T. Bailey. SLAM simulations. [http://www-personal.acfr.usyd.edu.au/tbailey/software/slam\\_simulations.htm](http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm).
- T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part II State of the art. *Robotics and Automation Magazine*, 13:108–117, 2006.

	“Small” problem			“Large” problem		
	time	landmark error	state error	time	landmark error	state error
MCMC-SLAM 1	580.945	98.312	92.7304	4675.93	217.921	189.6
MCMC-SLAM 10	781.555	16.231	15.4099	5796.37	95.8177	81.0959
MCMC-SLAM 100	2859.77	6.90057	6.89335	18624.0	18.3329	15.2871
FastSLAM2 100	49.857	118.953	16720.0	198.671	196.821	91655.8
FastSLAM2 1000	484.262	65.3241	16499.6	1875.87	79.6746	91108.0
FastSLAM2 10000	4172.91	14.2413	16375.8	18562.2	29.4634	90983.8
EKF-SLAM	7.09411	170.335	197.881	35.3062	378.363	154.242

Table 1: Performance of MCMC-SLAM, FastSLAM 2.0 and EKF-SLAM on two artificial problems.

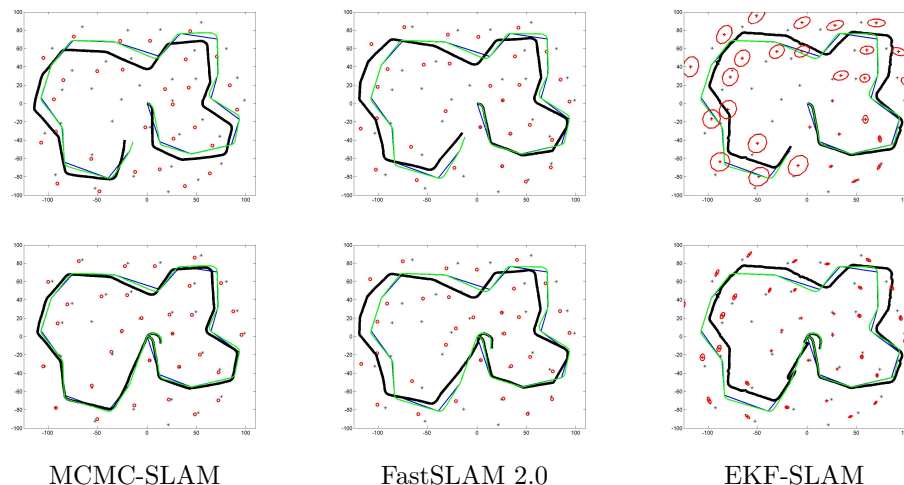


Figure 2: The performance of our MCMC-SLAM, FastSLAM 2.0, and EKF-SLAM on the “small” problem before and after a loop is closed. The blue (green/lightest) line represents the intended (resp., actual) robot trajectories, the thick black line is the estimated trajectory. Landmark positions are shown as stars, and the red dots (ellipses for EKF-SLAM) represent the estimated landmark positions.

- T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM algorithm. In *ICRA*, pages 424–429, 2006.
- P. Cheeseman and R. Smith. On the representation and estimation of spatial uncertainty. *International Journal of Robotics*, pages 56–58, 1986.
- A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York, 2001.
- H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping (SLAM): Part I The essential algorithms. *Robotics and Automation Magazine*, 13:99–110, 2006.
- M. Kaess and F. Dellaert. A Markov Chain Monte Carlo approach to closing the loop in SLAM. In *ICRA*, pages 643–648, 2005.
- T. Katayama. *Subspace Methods for System Identification*. Springer-Verlag, 2006.
- N. Lee, D.S. Chia. The sequential MCMC filter: formulation and applications. In *Statistical Signal Processing*, pages 30–33, 2001.
- R. Martinez-Cantin, J. Castellanos, and N. de Freitas. Multi-robot Marginal-SLAM. In *IJCAI Workshop on Multi-Robotic Systems for Societal Applications*, 2007a.
- R. Martinez-Cantin, N. de Freitas, and J. A. Castellanos. Analysis of particle methods for simultaneous robot localization and mapping and a new algorithm: Marginal-SLAM. In *ICRA*, pages 2415–2420, 2007b.
- M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI*, pages 593–598, 2002.
- M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *IJCAI*, pages 1151–1156, 2003.
- J. Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32:245–257, 1987.
- A. Ranganathan and F. Dellaert. Data driven MCMC for appearance-based topological mapping. In *Robotics: Science and Systems*, pages 209–216, 2005.
- R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer, 1990.
- S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.
- L. Tierney. A note on Metropolis-Hastings kernels for general state spaces. *The Annals of Applied Probability*, 8(1):1–9, 1998.