# Efficient Learning of Motion Patterns for Robots

**Stephan Sehestedt, Sarath Kodagoda, Alen Alempijevic and Gamini Dissanayake**
ARC Centre of Excellence for Autonomous Systems
The University of Technology, Sydney
Australia
{s.sehestedt, s.kodagoda, a.alempijevic, g.dissanayake}@cas.edu.au

**Abstract**

In this work we present a novel approach to learning dynamics of an environment perceived by a mobile robot. More precisely, we are interested in general motion patterns occurring in the environment rather than object dependent ones. A sampling algorithm is used to update a sample set, which represents observed dynamics, using the Bayes rule. From this set of samples a Hidden Markov Model is learnt online, which allows fast and efficient matching and prediction in the learnt model. Such models are useful for a number of tasks such as path planning, localisation and compliant motion. The approach is validated through simulation as well as experiments.

## 1 Introduction

Mobile robotic systems are often deployed in controlled, often static, environments, as it is difficult to handle highly dynamic situations efficiently. There are a number of techniques to deal with some of the dynamics, such as door state estimation [Avots *et al.*, ] or to avoid dynamics in the observations. This can be achieved by pointing the sensors in a favourable direction as done on Minerva [Thrun *et al.*, ] or by using tracking algorithms so that other components of the robotic system will have minimum interference with moving objects [Roy *et al.*, ].

These techniques were successfully employed and are good measures to enable robust localisation and mapping. However, this leaves valuable information almost completely ignored. Having a priori knowledge about dynamics in a given environment could be used for localisation as well as path planning and other tasks.

Thus far, most work in the field of learning dynamics was done in video surveillance, where a number of assumptions can be used, which ease the problem. Common assumptions are the presence of a stationary observer; always seeing complete trajectories from start to end and tracked objects having similar, sometimes constant, speed. For examples and further references see [Makris and Ellis, 2001], [Swears *et al.*, ] and [Govea *et al.*, ].

In the domain of mobile robotics these prerequisites cannot be guaranteed. Therefore, researchers are focussing on the development of learning algorithms based on environment monitoring. However, so far we are only aware of publications where static observers are utilised.

Bennewitz et al. [Bennewitz *et al.*, ] developed a method to learn a model of dynamics in an office environment which was used for a mobile robot. Even so, the learning phase relies on stationary sensors and the observation of complete paths and learning is done offline. Also in this work start and end points of a path must be learnt and it is assumed that motion always happens between those points (e.g. object starts at start point A and ends at end point B).

Vasquez et al. [Govea *et al.*, ] propose growing Hidden Markov Models (GHMM) to incrementally learn motion patterns in an area. This allows for online learning but it is still assumed that the sensor is mounted in a fixed location, which limits the usefulness of the proposed method for mobile robots.

In this work we propose a novel approach to learning typical motion patterns (dynamics) in a given environment based on a sampling methodology. In this way the model can be learnt incrementally and can in principle be used on a mobile platform. Furthermore, a method is proposed to extract a Hidden Markov Model which can be used for efficient evaluation of the model of dynamics.

The remainder of this publication is organised as follows. In Section 2 we present our method to learn dynamics by sampling. Section 3 then introduces the reader to Hidden Markov Models after which our approach to learning a Hidden Markov Model of motion patterns is described. Furthermore, we present results in sections 4 and 5 and finally discuss our conclusions and outline of future work.

## 2 Learning a model of motion patterns

In this section we introduce our proposed methodology for learning motion patterns. We want to build a model of generally occurring motions in a known environment. In the following the existence of an object tracker is assumed, from which we expect to get a position of a moving object and a prediction of where this object will be in the next time step. For readability the method will be formulated for a single target tracker as the extension to multiple targets is straight forward.

### 2.1 Object Tracking

For object tracking we use an implementation of an interacting multiple model tracking algorithm [Blom and Bar-Shalom, ]. To simplify the description and notation in the following sections, we assume the tracking algorithm, as outlined below (also see Figure 1).

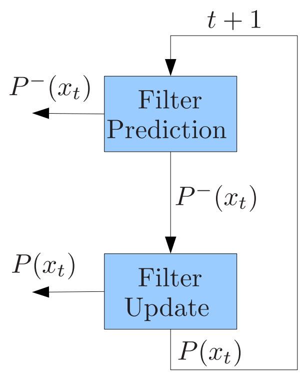


Figure 1: Diagram of the tracking algorithm

In every time step $t$ the tracking algorithm produces a single estimate $x_t$ of a moving objects position.

$$P(x_t) = P(x_t|x_{t-1}, z_t) \tag{1}$$

with $x_t$ being the state of the tracked object and $z_t$ the observation at time $t$.

The tracker also reports the predicted position of the tracked object.

$$P^-(x_t) = P(x_t|x_{t-1}) \tag{2}$$

Estimate and prediction should be reported as mean $\mu$ and covariance $\Sigma$.

### 2.2 Motion Sampling

A probability distribution $P(D)$ exists which represents the occurrence of motion in any point of space. This can be expressed as

$$P(D) = P(x, y, \theta, v) \tag{3}$$

where $x - y$ denotes a coordinate in the 2D plane, $\theta$ is an angle and $v$ denotes the speed of a motion. Thus, $P(D)$ is the joint probability of the simultaneous occurrence of a particular $x-y-\theta-v$. Arguably, this distribution can be complicated and therefore, an approximation is needed which can be learnt. Naturally, learning can be done by observing the system and trying to extract a model and its parameters. For observing the aforementioned moving object tracking is used in conjunction with localisation.

Hence, $P(D)$ can be approximated by a model $P(D_t)$ at time $t$ using the observers state $o_t$ and observations $z_t$.

$$P(D) \approx P(D_t|o_{1:t}, z_{1:t}) \tag{4}$$

where $D_t$ incorporates all observations up to time $t$. Clearly this distribution is complex and thus, it is difficult to learn the parameters. Therefore, we propose to approximate this probability distribution by a set of weighted samples

$$X_t = \langle x_t^{(i)}, \omega_t^{(i)} \rangle, \quad 1 \leq i \leq N \tag{5}$$

with $X_t$ being the sample set at time $t$, consisting of $N$ samples $x_t^{(i)}$ and $\omega_t^{(i)}$ being the weight of the $i$-th sample. Each sample represents a probability of the occurrence for a certain kind of motion.

Hence each sample $x_t^{(i)}$ is defined as

$$x_t^{(i)} = \begin{bmatrix} x^{(i)} \\ y^{(i)} \\ \theta^{(i)} \\ v^{(i)} \end{bmatrix} \tag{6}$$

We combine the sample sets $X_{1..t}$ in

$$D_t = \langle X_{1..t} \rangle \tag{7}$$

Hence, we obtain the approximation

$$D_t \approx P(D) \tag{8}$$

This approximation can be updated using the Bayes rule, utilising the prediction (eq. 2) and update (eq. 1) from the tracking algorithm. With new observations the sample set will grow in order to incorporate new information. To make sure that no redundant information is kept and to control the number of samples a subsampling procedure is used. The result of this procedure can be seen in Fig. 2(a), where one object moving from the left to the right was observed. Fig. 2(b) shows the sample set after a number of similar observations.

Obviously, such a model can grow to a substantial size using a lot of memory and facilitating the model for matching and prediction of observed motion would be computationally expensive and non trivial. Therefore, we use the sampled distribution to incrementally learn and update a Hidden Markov Model as described in the following section.

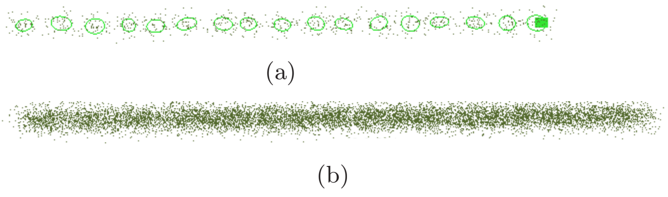


Figure 2: a) The object (green rectangle) moved from the left to the right. The dark points denote samples generated from the trackers prediction. The green ellipses denote the covariance after weighing the samples according to the most recent observation of the target b) The sample set $D_t$ after more objects were observed.

## 3 Efficient Representation

To decrease computational complexity and memory footprint, we derive a Hidden Markov Model (HMM) from the sampled distribution. This part first introduces the reader to HMMs and then describes how to incrementally learn and update an HMM, which will approximate $D_t$. Finally, model matching and prediction will be discussed.

### 3.1 Hidden Markov Models

A Hidden Markov model (HMM) is a statistical model that represents a system as a directed graphical model. Here we briefly outline HMMs following the notation used by Rabiner [Rabiner, ]. HMMs are defined by $N$ states of a system $S = s^1, s^2, ..., s^N$, together with the observation symbols $V = v^1, v^2, ..., v^M$ with $M$ being the number of symbols.

A state transition probability distribution $A = a_{ij}$ is given as

$$a_{ij} = P(q_{t+1} = s^{(j)} | q_t = s^{(i)}), 1 \leq i \leq N \quad 1 \leq j \leq N \tag{9}$$

Furthermore, the observation probabilities in state $j$, $B = b_{ij}$ are formulated as

$$b_{ij} = P(v^{(i)} | s^{(j)}), 1 \leq i \leq M \quad 1 \leq j \leq N \tag{10}$$

Finally, the initial state distribution $\pi = \pi_i$ is defined as

$$\pi_i = P(q_1 = s^{(i)}), 1 \leq i \leq N \tag{11}$$

Most HMM frameworks highly depend on prior knowledge of the topology of the model and learning is with previously obtained data. There is no easy way to update these models over time [Rabiner, ]. Thus, these implementations are not suitable for the application at hand. There are numerous, usually application dependent, derivatives of hidden Markov Models reported in the literature and we will briefly refer to the ones most relevant to the presented work.

The idea of using HMMs to model dynamics is not new, however, comparatively few publications are found in the realm of mobile robotics. The use of a hierarchy of HMMs to describe motion patterns on different levels was proposed in [Liao *et al.*, ]. However, the topology is given and learning is done offline. Vasquez et al. [Govea *et al.*, ] propose Growing Hidden Markov Models, which can be learnt incrementally, so the topology is also learnt online. Still, the assumptions that are made, prevent their use in practical mobile robotic applications. In particular, complete trajectories must be observed, meaning objects always have to be seen from the start of the path to the very end and the observer is in a fixed location.

In contrast, in the following section we will present an approach which allows to efficiently learn and update an HMM over time, which does not assume full observability of trajectories.

### 3.2 Deriving a Hidden Markov Model

**Sampling the states**

When creating the model, we do make the assumption that when an object moves the performed motion at time $t$ is independent from $t-1$ in order to be able to use a first order Markov Model. Using the sampling algorithm described in Section 2.2 we obtain samples with a fixed frequency with the result that we get a sequence of clusters of samples.

These clusters can be interpreted as estimates of the states the object has been in and hence, we use the sample statistics to define the estimated states $s^{(i)}$

$$S = s^{(i)} = \begin{bmatrix} \mu^{(i)} \\ \Sigma^{(i)} \end{bmatrix} \quad 1 \leq i \leq N \tag{12}$$

with $\mu^{(i)}$ being the mean, $\Sigma^{(i)}$ is the covariance of the $i$-th sample cluster and $N$ is the number of states. A 2D projection of a state can be seen as the covariance ellipse in $x$ and $y$ (see Figure 3).

Whenever another moving object is observed in a region where a model was previously learnt, the statistics of states are updated by combining the according sample clusters. Therefore, the time dependency needs to be incorporated into the definition of a state,

$$S_t = s_t^{(i)} = \begin{bmatrix} \mu_t^{(i)} \\ \Sigma_t^{(i)} \end{bmatrix} \quad 1 \leq i \leq N \tag{13}$$

However, for better readability this time dependency will be omitted in the following.

By observing a moving object, the resulting cluster can be seen as the $j$-th state $s^{(j)-}$ in the path of the object. The superscript "$-$" means that this is either a new state or may add new information to an already existing state. To decide whether the robot observed something new or got new information concerning the existing model, we regard $s^{(i)}$ and $s^{(j)-}$ as probability distributions which we can compare using a symmetric version of the Kullback-Leibler divergence (KLD)

$$\begin{aligned} KLD(s^{(i)}|s^{(j)-}), 1 \leq i \leq N \\ 1 \leq j \leq K \end{aligned} \tag{14}$$

where $K$ is the number of clusters from the tracked objects path. If the KLD is below an adaptive threshold, the sample clusters are combined and the state statistics change accordingly. To avoid growing computational effort with a growing model, we only calculate the KLD for clusters which are closely located in the $xy$ plane.

**Learning transitions**

The sequence of clusters also implies the sequence of transitions. Thus, we define transitions between the states we just learnt, resulting in a transition matrix $A$. Each individual transition $a_{ij}$ is defined as

$$a_{ij} = \begin{bmatrix} \Delta T \\ N_o \\ P(s^{(j)}|s^{(i)}) \end{bmatrix} \tag{15}$$

where $\Delta T$ is the time from the first observation of the transition until the last observed occurrence, $N_o$ denotes the number of times the transition was observed and $P(s^{(j)}|s^{(i)})$ is the transition probability from state $s^{(i)}$ to state $s^{(j)}$. An example for a resulting model after an object was observed for some time can be seen in Figure 3.

$\Delta T$ together with $N_o$ is useful as a measure of traffic density. Of course, this value only gets interesting with a reasonable number of observations. $N_o$ is needed to update the transition probability using new observations. Naturally, the probability is calculated as

$$P(s^{(j)}|s^{(i)}) = \frac{\sum_{j=0}^{k} N_{ij}}{N_i} \tag{16}$$

With $k$ being the number of outgoing transitions from state $i$, $N_i$ is the sum of all observations of outgoing transitions and $N_{ij}$ denotes the number of times one particular transition $a_{ij}$ was seen.

It is to be noted that once the robot observes an object moving along an already known path (i.e. a model is learnt already) the transition probabilities can be updated by just counting. This phenomenon is illustrated in Figure 5 in Section 4.

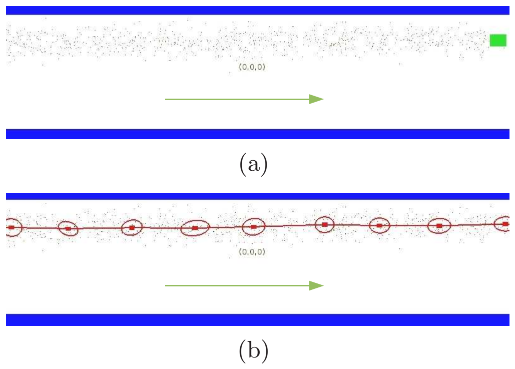


Figure 3: a) Objects (green rectangle) were observed moving from the left to the right along the wall, and samples (dark dots) were generated accordingly. b) From the resulting sample set $D_t$, states (red ellipses) are learnt.

## 3.3 Evaluating the Model

The learnt HMM can be used for predicting future poses of moving objects and matching observed trajectories to the model. This can be achieved using standard HMM algorithms, however, considering the computational complexity it is preferable to evaluate the model locally.

Matching an observed trajectory to a learnt model is done by only taking nearby states into account. For this smaller part of the model the forward algorithm [Rabiner, ] can be utilised. Prediction can be done by estimating in which state of the learnt model the object is most likely. From there the transition probabilities can be used for predicting into the future. Since the model of dynamics also encodes velocities, it is possible to predict future poses and speed.

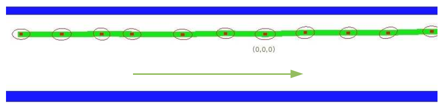
When evaluating bigger parts of the model, it may be necessary to introduce an additional normalisation step as two paths which lead to the same goal are not guaranteed to have the same number of states. In any case, the further the algorithm predicts into the future the less confident it is about the estimate.

## 4 Simulation Results

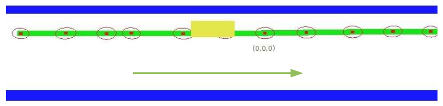
**Model dynamics**

In the first experiment we present the dynamic behaviour of our model of dynamics. Assume an observer sees people moving down a hallway and builds a model as shown in Figure 4(a). In this example the red ellipses denote the states, the green edges are the transitions between states and the green arrow below the model indicates the direction of the transitions.

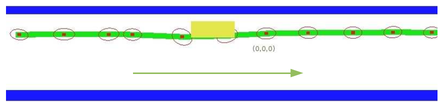
At some point an obstacle is put into the learnt path, so that people now have to slightly alter their trajectory (Figure 4(b)). In Figure 4(c) to 4(e) it can be seen how the model is adjusted, given the new information. In fact, the mean and covariance of the affected states changes significantly.



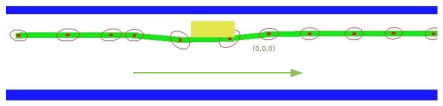
(a)



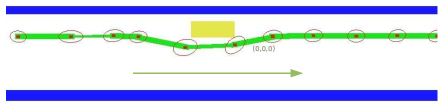
(b)



(c)



(d)



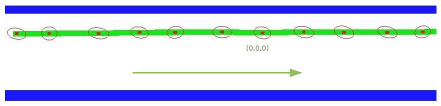
(e)

Figure 4: a) Shows the initially learnt model. Red ellipses show the states ,the green lines denote transitions and the green arrow indicates the direction of the modelled motion. b-e) After an obstacle was introduced in the originally learnt path, objects move around this obstacle and the model changes by shifting the mean of some states.

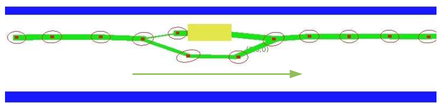
We repeated the same experiment, but changed the location of the obstacle such that people had to change the trajectory even more (Figure 5(a)). The result is that a new path is added to the model (Figure 5(b)). After enough observation, we can also see that the new path now has a higher probability, which is indicated by the thickness of the edge between the states.

### The 5 o'clock model

Suppose we have an observer overlooking most of the office space shown in Figure 6. The coloured arrows denote trajectories of people, who will leave the office around 5pm, which the observer will see in the following simulation.



(a)



(b)

Figure 5: a) Shows the initially learnt path. The thickness of the green transition lines denotes the probability of a transition. b) Again an obstacle was introduced. In this example the change in path is bigger, so a new path is added to the model.

Figure 7 shows the learnt model after 25 observed trajectories. The red ellipses represent the state covariances and the red edges are the transitions. Each transition has a direction, which is indicated by the green arrows. The model is learnt online, meaning that no postprocessing is taking place and trajectories are incorporated into the model as they are observed.

Then an obstacle is put into a path, which can be seen in Figure 8 as a yellow rectangle. Now people have to choose different ways. As they do this the new paths are learnt (having a yellow arrow to indicate the new paths diretcions) and become part of the model.

It can be seen that new paths can easily be included into the model, as well as states can shift their mean and change the covariance, according to new information. Figure 8(b) highlights part of the model. In this figure the thickness of the red edges indicates the transition probability and it can be seen that the transition from A to E, which was learnt before the obstacle was introduced is lower than the transition probability A to B.

It can also be seen that the transition from C to D became less likely than the transition C to F after the obstacle blocked the way for a while. Thus, again highlighting the dynamic properties of the proposed modelling method.

Finally, Figure 6 shows the raw sample set from which the above presented model was built.

## 5 Experimental Results

In early experiments we used a stationary observer with a position estimate. In Figure 9(b) the estimated position of the observer is shown as a red triangle which also indicates its heading. The sensor is a Sick LMS 291 laser scanner.

In Figure 9 and 10 the green arrow denotes the di-

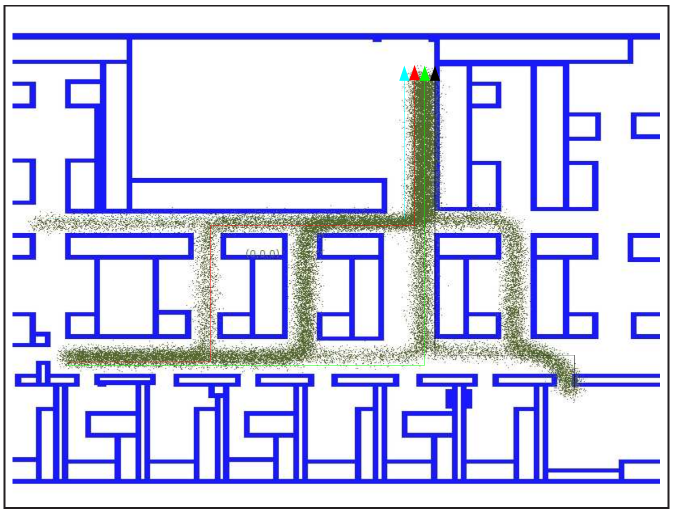


Figure 6: Part of the map of our office space (appr. 18x22m). The coloured lines show which pathes are observed. The observer can overlook most of the area. The green points represent the sample set from which the HMM in Figure 8(a) is learnt.

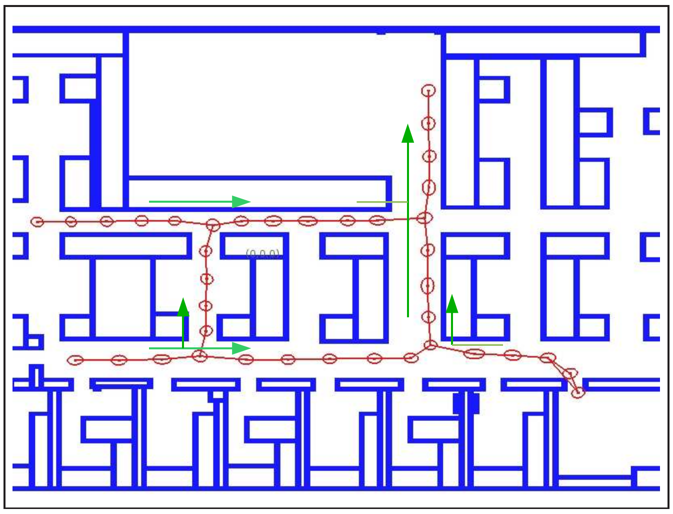


Figure 7: The motion model after 25 observed trajectories. The states of the HMM are shown as red covariance ellipses with a rectangle denoting the mean. The state transitions are shown as red edges, the green arrows indicate the models direction.

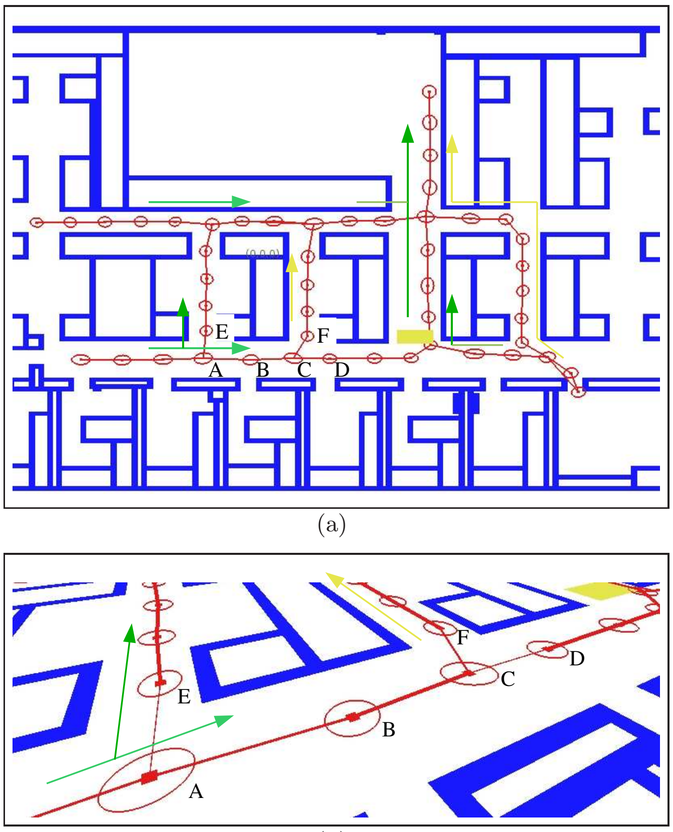


Figure 8: a) An obstacle (yellow rectangle) was inserted into the environment, blocking a path. As objects have to choose different trajectories now, the model adapts accordingly. The yellow arrows denote the models direction where new paths were added. The green arrows indicate the models directions where it was learnt before the obstacle was introduced. b) Close up view on part of the model. The transition probabilities are indicated by the width of the edges.

rection of motion of the objects that were tracked. Note that the obstacle marked with an "A" is just a low bench which did not obstruct the robots vision.

In the experiment people walked past the robot and turned left, following the green arrow. In Figure 9(a) the object was not immediately detected, resulting in a slightly shorter initial model. In subsequent observations the tracker picked up objects earlier and thus new states were added to the model (see Figure 9(a)).

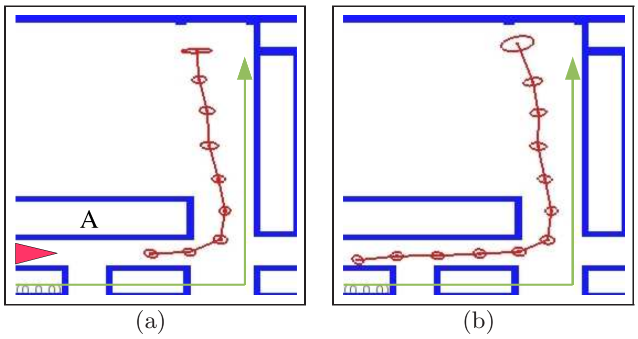


Figure 9: a) The observer (red triangle) overlooks most of the shown area. Objects pass the robots position following the green arrow. The red ellipses and edges show the initial model of motion patterns. b) Due to new information new states are added.

To verify the simulation results shown in Figure 5 we let objects deviate from the learnt path, thus forcing the algorithm to extend the model as seen in Figure 10. Hence, it can be seen that the results from the simulation experiment are also valid for real world experiments.

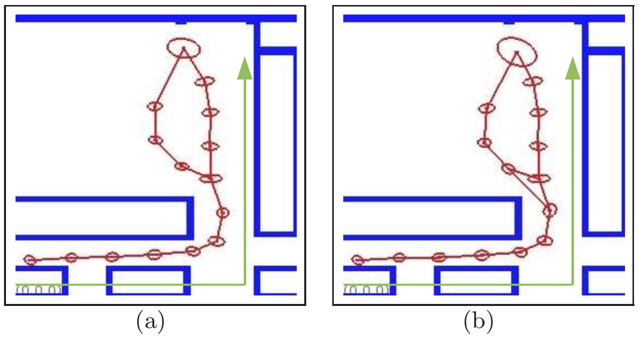


Figure 10: a and b) Objects deviate from the learnt model forcing the algorithm to learn a new path.

## 6 Conclusions

In this publication we presented a novel approach to learning models of motion patterns. A sampling algorithm is proposed which allows for online learning and updating a Hidden Markov Model. It was shown that motion patterns in larger areas can be learnt and new paths can be easily added to the model.

Such a model can be used for various tasks, such as path planning, moving object tracking and many more, which we are currently embarking on. Furthermore, in the near future we expect to scale such a model to 3D space without having to substantially increase the computational effort and memory demand.

**Acknowledgements**

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

[Avots *et al.*, ] D. Avots, E. Lim, R. Thibaux, and S. Thrun. A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments.

[Bennewitz *et al.*, ] M. Bennewitz, W. Burgard, and S. Thrun. Learning motion patterns of persons for mobile service robots.

[Blom and Bar-Shalom, ] Henk A.P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients.

[Govea *et al.*, ] Dizan Alejandro Vasquez Govea, Thierry Fraichard, and Christian Laugier. Incremental learning of statistical motion patterns with growing hidden markov models.

[Liao *et al.*, ] Lin Liao, Dieter Fox, and Henry A. Kautz. Learning and inferring transportation routines.

[Makris and Ellis, 2001] Dimitrios Makris and Tim Ellis. Finding paths in video sequence. In *Proceedings of the British Machine Vision Conference*, pages 263–272, 2001.

[Rabiner, ] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition.

[Roy *et al.*, ] Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal navigation – mobile robot navigation with uncertainty in dynamic environments.

[Swears *et al.*, ] Eran Swears, Anthony Hoogs, and A. G. Amitha Perera.

[Thrun *et al.*, ] Sebastian Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, Frank Dellaert, Dieter Fox, D. Haehnel, Chuck Rosenberg, Nicholas Roy, Jamieson Schulte, and D. Schulz. Minerva: A second generation mobile tour-guide robot.