Robot Path Planning in a Social Context

Stephan Sehestedt, Sarath Kodagoda and Gamini Dissanayake
ARC Centre of Excellence for Autonomous Systems (CAS)
The University of Technology, Sydney
Australia
Email: {S.Sehestedt, S.Kodagoda, G.Dissanayake}@cas.edu.au

***Abstract*—Human robot interaction has attracted significant attention over the last couple of years. An important aspect of such robotic systems is to share the working space with humans and carry out the tasks in a socially acceptable way. In this paper, we address the problem of fusing socially acceptable behaviours into robot path planning. By observing an environment for a while, the robot learns human motion patterns based on sampled Hidden Markov Models and utilise it in an Probabilistic Roadmap based path planning algorithm. This will minimise the social distractions such as going through someone else's working space (due to the shortest path), by planning the path through minimal distractions, leading to human-like behaviours. The algorithm is implemented in Orca/C++ with appealing results in real world experiments.**

***Index Terms*—Human robot interaction, path planning, motion models, learning, hidden markov models**

I. Introduction

Path planning is a fundamental ability for a robot to be able to efficiently perform tasks in its environment. Alongside with other abilities such as localisation and moving object tracking, it builds the basis for effective interaction with the environment.

Approaches to path planning can be classified as reactive or deliberative. Reactive methods do not use any prior knowledge and only the latest sensor readings are used to trigger reactive behaviours, resulting in a succession of short term plans [1]. Deliberative approaches use a previously built model of the (usually) static environment. As opposed to reactive methods, deliberative approaches provide globally optimal (or approximate) solutions at the cost of higher computational effort. It has been noted that in real world applications precomputed paths may be invalidated by the occurrence of unexpected events, or more precisely dynamic environments are poorly handled by approaches which assume a completely static surrounding [2]. Consequently, hybrid approaches and methods for replanning are pursued, where replanning/reacting is done using the latest sensor information [1] [2].

While technically solving the path planning problem, the aforementioned methods do not consider any a priori knowledge about motion patterns, which may limits their applicability. The term motion pattern refers to usually observed trajectories in an environment. Ideally such information can be used to improve global path planning and local replanning or reaction. We are only aware of very few publications in this area of interest. In one approach Kruse et al. [3] built models of dynamics in small office environments with the goal to avoid regions where the robot is likely to encounter moving objects. Thus, the probability of the robot having to react and replan could be minimised. In another approach, Bennewitz et al. [4] propose learning motion patterns of individual persons in an office to enable the robot to predict an identified persons motion for the robot to avoid collision based on the prediction.

Although these methods have appealing results, they approaches are minor steps towards the full potential of human robot interaction. Humans are excellent at planning paths. They do not only utilise the shortest path as criteria, but also consider social interaction in planning paths (among other information). For example, they prefer to walk along corridors rather than going through desk areas in an office environment although the latter could be the shortest path. By doing so, they create minimal distractions to the office occupants.

In our opinion, the ability to adapt to a human environment is essential for the acceptance of robots operating in shared spaces. Therefore, in this work we outline a method to learn generally occurring motion patterns in an office environment on-line with a mobile robot. The method is based on sampling from observations, and subsequent clustering resulting in a Sampled Hidden Markov Model (SHMM) can be derived. This model is rich in socially compatible path plans, which can then be exploited in robotics path planning. No offline learning is required and on-line learning can even be performed with partial observations.

This paper is organised as follows. In Section II we outline our approach to on-line learning of models of motion patterns with a mobile robot. In Sections III the proposed approach for using the previously learnt motion pattern models to plan paths more socially acceptable is introduced. Section IV presents experimental results with our IRobot Create robot. Finally, Section V concludes the paper.

II. Models of Motion Patterns

In this section we outline a methodology for representing motion patterns using Hidden Markov Models. The proposed model is learnt incrementally using a sampling algorithm. For more details of the proposed method, please refer to [5].

A. 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 [6]. 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.

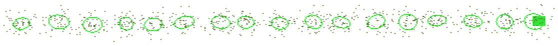


Fig. 1. The object (green rectangle) moved from the left to the right. The dark points denote samples generated from the tracker's prediction. The green ellipses denote the covariance 95% contours after weighing the samples according to the most recent observation of the target.

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

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

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

$$b_{ij} = P(v^{(i)} | s^{(j)}), 1 \leq i \leq M \quad 1 \leq j \leq N \tag{2}$$

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

$$\pi_i = P(q_1 = s^{(i)}), 1 \leq i \leq N \tag{3}$$

HMMs are used in a number of different fields with great success. In the following section we outline Sampled Hidden Markov Models for learning motion patterns.

B. Sampled Hidden Markov Models

Consider a person walking along a corridor from left to right. The person could be tracked and the samples of states could be easily determined as shown in Fig. 1. Note that in this example the samples are taken with a fixed frequency, thus forming clusters with inter-cluster distances corresponding to the speed of the tracked object.

To derive a Hidden Markov Model, each cluster can be used as an estimate of a state and the sequence of clusters defines the transitions.

$$S = s^{(i)} = \begin{bmatrix} \mu^{(i)} \\ \Sigma^{(i)} \end{bmatrix} \quad 1 \leq i \leq N \tag{4}$$

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 Fig. 2).

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

$$S_t = s_t^{(i)} = \begin{bmatrix} \mu_t^{(i)} \\ \Sigma_t^{(i)} \end{bmatrix} \quad 1 \leq i \leq N \tag{5}$$

However, for better readability the subscript t (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. The decision can be made based on the symmetric version of the Kullback-Leibler divergence (KLD) [7] representing $s^{(i)}$ and $s^{(j)-}$ as probability distributions.

$$KLD(s^{(i)} | s^{(j)-}), 1 \leq i \leq N \quad 1 \leq j \leq K \tag{6}$$

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 are changed accordingly. The threshold is computed based on the expected distance between consecutive states. To avoid growing computational effort with a growing model, we only calculate the KLD for clusters which are closely located in the $x - y$ plane.

1) 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

$$a_{ij} = \begin{bmatrix} \Delta T_{ij} \\ N_{ij} \\ P(s^{(j)} | s^{(i)}) \end{bmatrix} \tag{7}$$

where ΔT_{ij} is the time from the first observation of the transition until the last observed occurrence, N_{ij} 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 Fig. 2.

ΔT_{ij} together with N_{ij} is useful as a measure of traffic density, although this value is only significant with a reasonable number of observations. N_{ij} is needed to update the transition probability using new observations. Naturally, the probability is calculated as

$$P(s^{(j)} | s^{(i)}) = \frac{N_{ij}}{N_i} \tag{8}$$

With N_i being the sum of all observations of outgoing transitions of state i and N_{ij} denoting 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 was learnt already) the transition probabilities can be updated by just counting. However, if there is a model which was learnt over a long period of time with many observations, the adaptation of the transition probabilities may be very slow. To overcome this, it is possible to use the transitions per unit of time to calculate the transition probabilities according to

$$P(s^{(j)} | s^{(i)}) = \frac{N_{ij} / \Delta T_{ij}}{\sum_{k=0}^{n} N_{ik} / \Delta T_{ik}} \tag{9}$$

with n being the number of outgoing transitions from state i.

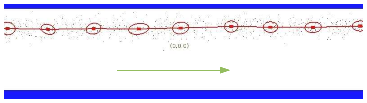


Fig. 2. States (red ellipses) are estimated using the underlying sample set.

2) Reducing the Dimensionality of the Model: In order to make the evaluation of the model computationally less demanding, it is possible to use the properties of the HMM to reduce the dimensionality. When sampling with a fixed frequency as proposed, the distance between the means of successive states encodes the average speed which was observed in the area and thus it is possible to represent a state as x, y, θ. Obviously, if an object is observed moving at a vastly different speed than the learnt model suggests, this object could belong to a different model. Thus, a distance measure for the speeds or binning replace the explicit use of v in the model. Furthermore, it is possible to use binning for the heading θ, reducing the dimensions of a single state to just x, y.

III. Path Planning in Populated Environments

In this section we present our approach to improve path planning using the learnt motion pattern model. The goal is to increase efficiency of the planner as well as to add basic social awareness without explicitly learning social rules.

A. Related Work

Here, we outline the most relevant publications in the context of this paper. It is to be noted that there are only a few published papers in this area of research.

Bennewitz et al. [4], [8] used an off-line learning procedure to obtain motion models of individual people in an office. These individual patterns are represented as HMMs with a fixed number of nodes and defined start and end locations. Once a person is identified, the respective model is used to predict that persons future position. Should the robot encounter a person it will attempt to predict the future pose of the person in order to give way. Thus, the model is used to improve local replanning and does not influence global planning. The authors argue that this leads to more compliant robot motion, which improves the robot's operation in an office environment. However, we argue that this way of reaction planning is not just compliant but recessive. Moreover, the model is not used for global path planning.

Kruse et al. [3] presented a similar approach representing trajectories stochastically by assigning a probability of occurrence according to a Poisson process. The resulting model is used for global path planning avoiding busy areas and to improve moving obstacle avoidance. They claim this makes the operation of the robot more efficient as less reactive behaviours are needed and the reactive behaviour itself is improved. However, it does not exploit the full potential of the idea as the prior knowledge is not used for improving interaction but to avoid interaction. One can even argue that in some environments it may decrease the robots performance, where the robot is supposed to be a co-worker.

B. Models of Motion Patterns in Path Planning

For a robot to efficiently and effectively play a part in a work place it has to comply with social behaviours of the co-wokers. People do not always plan their path along the shortest route, but take other considerations into account. E.g. in an office, one would usually avoid the paths through work stations to avoid distractions to colleagues. This is illustrated in Fig. 3, where a photo and a map of the open office space are shown with desk and corridor areas as outlined. Hence, there are situations where a robot should plan its motion in a compliant way rather than an avoiding way as given in [3].

Following a similar philosophy, we have included the previously learnt model of motion patterns into the path planning algorithm to make the robots operation more socially acceptable. The information that can be derived from the learnt motion patterns relevant for path planning are the motion patterns (trajectories) themselves and traffic density.

C. The A^* Algorithm

The A^* algorithm and its derivatives are a popular solution for the path planning problem [9] [1] [2]. It performs the best-first search on a grid which is precomputed using a collision detector with defined configuration values. More precisely a configuration space C is computed which contains all static obstacles in the d-dimensional space of the robot. C_{free} exists, which contains all collision free configurations. A path planning algorithm searches a path such that the path lies in C_{free}.

A^* is defined by the functions $g(x)$ which is the shortest path from start to goal by Euclidean distance (often called the path-cost function). Furthermore, $h(x)$ is used as a heuristic estimate of the length of the path and $f(x)$ is the sum of $g(x)$ and $h(x)$. The algorithm searches for a path using a priority queue, where the priority of node x is higher the lower its $f(x)$ is. Hence, it is called a best-first search.

This can be exploited to include prior information about dynamics in the environment. A cost function is used to evaluate the cost of a path with respect to a model of motion patterns

$$g_D(x) \sim D(x) \tag{10}$$

Where D denotes the learnt model of motion patterns. $g_D(x)$ returns a low value if node x is in an area of high traffic density and a high value if it is in an area of low traffic density. Consequently, instead of $g(x)$ int the standard $A*$-algorithm the function

$$G(x) = g(x) + g_D(x) \tag{11}$$

can be used to calculate $f(x)$. This cost function applies whenever the robot is supposed to prefer a commonly taken path. However, this may not always be a good solution. E.g. consider a robot with limited capabilities which should probably avoid busy areas or a robot whose has the task to empty all trash bins. Clearly eq. 10 should be reformulated to account for the requirements of a task the robot might have. Hence, a factor w is introduced

$$G(x) = g(x) + w * g_D(x), 0 \leq w \leq 1 \tag{12}$$

Where the factor w is chosen depending on the current requirements. Set to zero, the model will be ignored and paths are planned using normal A^*. If the robot prefers common paths w is set to 1 and any number in the range of 0 to 1 denotes whether it prefers the pure A^* distance or the combination with $g_D(x)$. Note that for some applications a robot may be required to avoid busy areas. For this we can change G(x) to be

$$\begin{aligned} G(x) = g(x) + w_g * g_D(x) + w_{\overline{g}} * \overline{g}_D(x), 0 \leq w_g \leq 1 \\ 0 \leq w_{\overline{g}} \leq 1 \end{aligned} \tag{13}$$

Where $\overline{g}_D(x)$ returns a low cost value for paths within low traffic areas. The factors w_g and $w_{\overline{g}}$ determine whether to prefer high or low traffic density areas or to ignore this information. Naturally, only one of the two values should be non-zero.

D. Probabilistic Roadmaps

The integration of Probabilistic Roadmap path planning (PRM) with the proposed model of motion patterns is straightforward and yields some interesting properties.

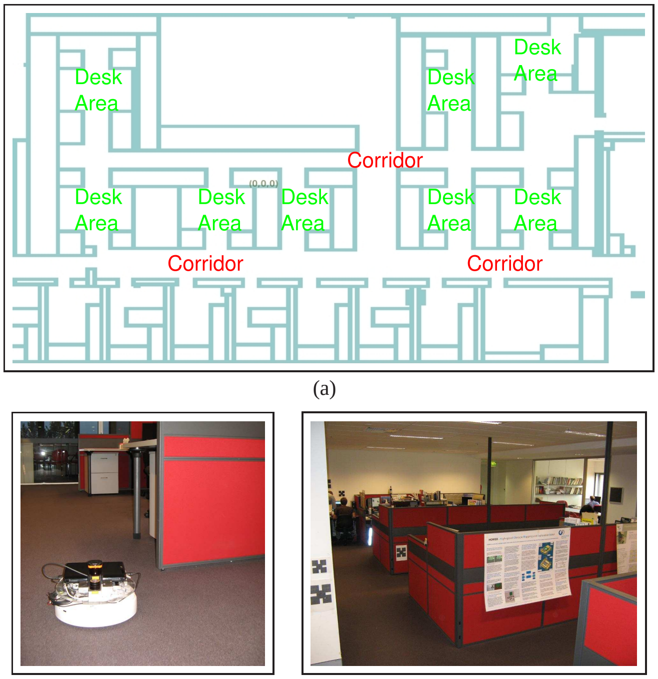
PRM was introduced as a method to overcome the issue of growing complexity in higher dimensions [10] [9]. The basic algorithm first constructs an undirected graph G, the roadmap, to then use this graph to solve path planning. The nodes of G are generated by random sampling and collision checking and path planning is done by traversing between nodes which are sufficiently close to each other. There are many publications presenting variations to the sampling step and collision checking in order to improve efficiency. Generally, PRMs have been applied successfully in different applications.

Once a graph is constructed, path planning can be done using A^* as described above. Consequently, when using the model of motion patterns D, at least part the graph G can be considered known and sampling can be restricted to unexplored areas, thus increasing efficiency. The more complete D is the less sampling needs to be done and hence, our motivation to use PRM.

IV. Experimental Results

In this section we present experimental results which were produced based on our IRobot Create platform. The robot carries a Hokuyo UTM-30LX laser and a small size notebook (eeePC) enabling it to navigate in the environment and observe motion (see Fig. 3(b) and 3(c)). Fig. 3(a) shows a map of an open office space where corridors and desk areas are labeled accordingly.

Fig. 4 presents the learning phase in which the robot observes motion in different parts of the office. In total the robot observed more than 60 trajectories in this experiment.



(a)

(b)

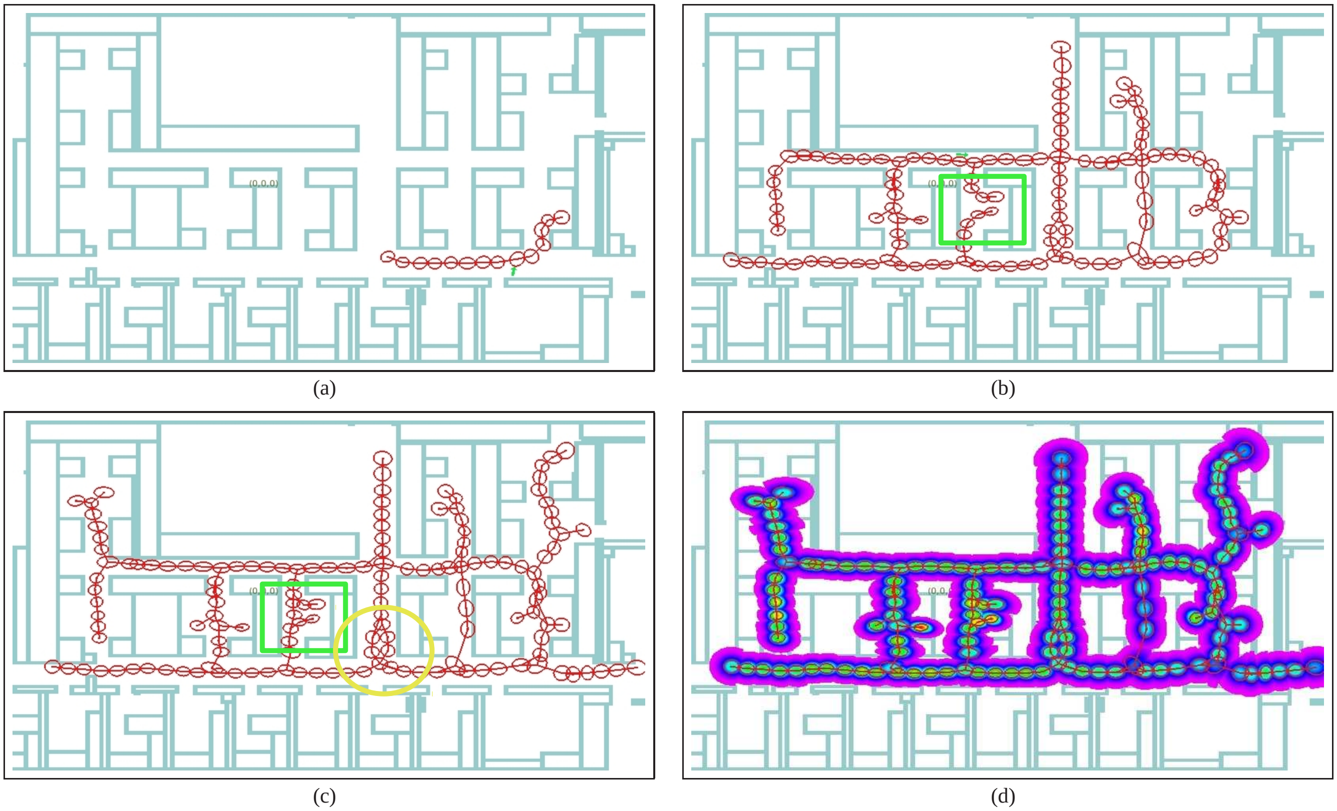
(c)

Fig. 3. a) The map. b) The IRobot Create in its environment. c) The office space.

The green arrow denotes the robot's position and heading, the red ellipses are the covariance ellipses of states in the SHMM and the red lines denote state transitions from the mean of one state to the mean of the next state. In the sequence of figures the evolution of the model can be seen from a single trajectory to a complex model of motion patterns. The model is correct with respect to the observations at any time. However, it is to be noted that there is no guarantee for completeness. This results from the fact that some patterns may not have been observed or missed due to tracking failure. This can be seen in Fig. 4(b), where there is a gap in the model (inside the green rectangle). Later in the experiment a trajectory was observed, which resulted in the gap being closed (see Fig. 4(c)). Also in in Fig. 4(c)), inside the yellow circle, an intersection is highlighted which seems to have a more complex structure than necessary. This stems from the fact that people do not turn 90° on the spot but behave like a non-holonomic vehicle, thus walking on an elliptic trajectory.

Furthermore, Fig. 4(d) adds a visualisation of the Gaussian distributions with which the states are represented. In some places an overlap with obstacles can be seen, because structure is not taken into account. This can result in very close proximity of a mean to a static obstacle in situations where the tracking algorithm produces inaccurate estimates. However, with more observations the affected part of the SHMM will be improved. Finally, The traffic density associated with every state also positively reflects the separation between desk areas and corridors as shown in Fig.5.

In the following we present our path planning results using



(a)

(b)

(c)

(d)

Fig. 4. a) The first observed trajectory in the model. The green arrow denotes the robots position. b) The model after the robot observed 7 trajectories. c) The model after observing 25 trajectories. d) The final model after observing more than 60 trajectories.

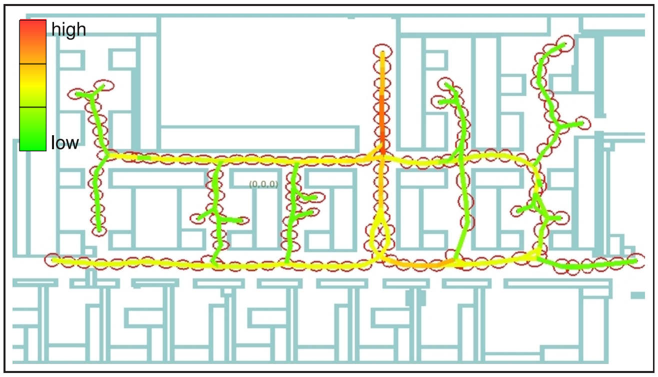


Fig. 5. The observed traffic density; colours range from green (low traffic density) to red (high traffic density).

the above presented model which the robot did learn. Examples are shown where the robot is supposed to avoid office spaces, i.e. a high value for w_g ind Eq. 13, and for comparison normal shortest path results are presented alongside. In the following figures start and goal locations are marked as such.

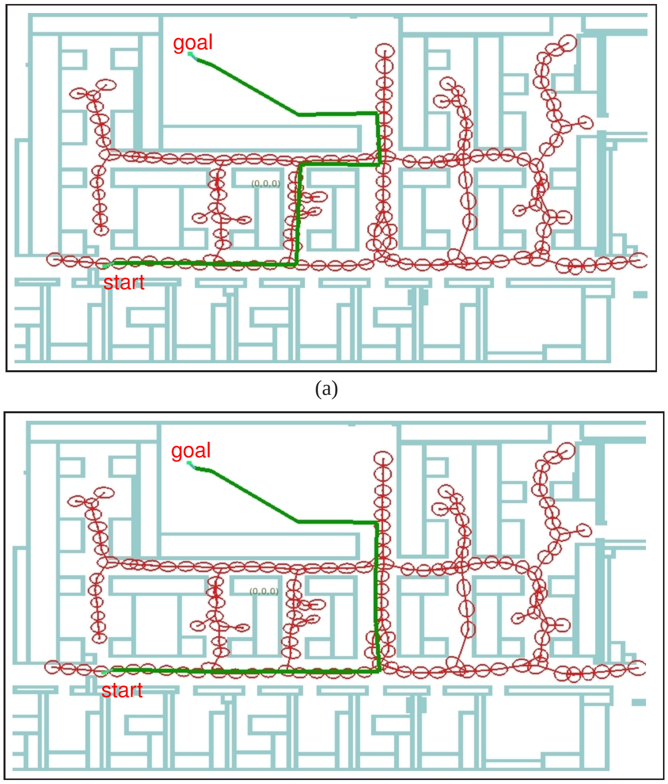
Fig. 6(a) shows a typical result of a planned path using only basic PRM without exploiting further knowledge. Since non of the possible paths is shorter than the others there is no preference. Thus, any of the possible routes will be chosen arbitrarily. In the shown example the chance (assuming randomness) of choosing a path which crosses a desk area is 75 percent. In contrast when using the proposed model of motion patterns the path will be consistently chosen to not cross any office spaces, as can be seen in Fig. 6(b).

Consider a robot which is supposed to plan a path, where the shortest path crosses a desk area. In that case basic PRM will always generate a path plan which would send the robot through this area, thus being a distraction to co-workers (see Fig. 7(a)). Setting a high value for w_g in Eq. 13 will guarantee that the robot avoids areas where it potentially disturbs workers. This can be seen in Fig. 7(a) where the robot actually plans a considerably longer path in order to stay out of the desk area.

However, the trade-off between shortest path and social consideration needs to be within reasonable limits. This is also assured given a sensible implementation of $g_D(x)$, as illustrated in Fig. 8. The detour is too long, resulting in a lower cost $f(x)$ for the shortest path. In a social context this mimics human behaviour, as the very same trade-off between shortest path and not disturbing colleagues is considered.

V. Conclusions

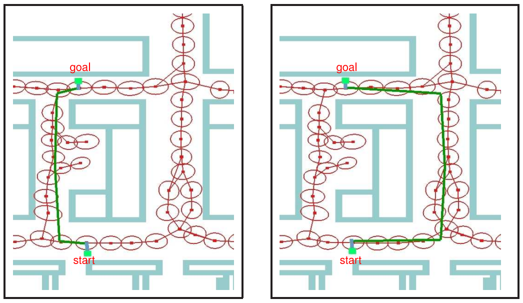
In this paper, a method was presented to allow a robot to plan paths considering social interactions. This philosophy is important as it will lead the robots to behave more like humans reinforcing human interactions. For this purpose, we



(a)

(b)

Fig. 6. a) A path generated using basic PRM. b) Path generated with regard to motion pattern model



(a)

(b)

Fig. 7. a) A path generated using basic PRM. b) Path generated with regard to motion pattern model, which is significantly longer than the shortest path.

have proposed a motion pattern learning algorithm to enhance the robot path planning algorithm. A weight is set to control the influence on path planning. Simulation results show that the robot, can plan paths with regard to a model of motion patterns avoiding certain areas where people want minimal interactions. It was shown how to achieve it using A^* path planning and Probabilistic Roadmaps, which integrate coherently with the motion pattern models.

Currently we are conducting more experiments with a robot in our office space. Future work involves the implementation of a replanning component which also uses the proposed model of motion patterns. We are particularly interested in determining whether a moving object needs to be avoided or can be followed as it may be following the same trajectory. Specifically, a method like the one presented by Müller et al. [11] may greatly benefit from additional prior information.

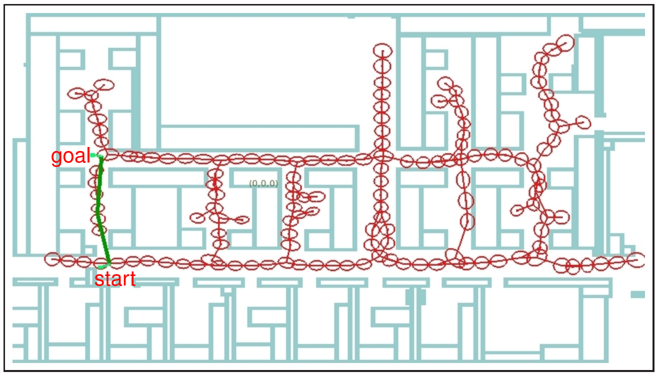


Fig. 8. The detour around the desk areas is too long. Hence the robot will take the path through the desk area (the shortest path).

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] T. Allen, A. Hill, J. Underwood, and S. Scheding, “Dynamic path planning with multi-agent data fusion - the parallel hierarchical replanner,” in *Robotics and Automation, 2009. ICRA ’09. IEEE International Conference on*, May 2009, pp. 3245 – 3250.

[2] M. Likhachev, D. Ferguson, G. Gordon, A. T. Stentz, and S. Thrun, “Anytime dynamic a*: An anytime, replanning algorithm,” in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.

[3] E. Kruse, R. Gutsche, and F. Wahl, “Acquisition of statistical motion patterns in dynamic environments and their application to mobile robot motion planning,” in *Intelligent Robots and Systems, 1997. IROS ’97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2, Sep 1997, pp. 712–717 vol.2.

[4] M. Bennewitz, W. Burgard, and S. Thrun, “Adapting navigation strategies using motions patterns of people,” in *Robotics and Automation, 2003. Proceedings. ICRA ’03. IEEE International Conference on*, vol. 2, September 2003, pp. 2000 – 2005.

[5] S. Sehestedt, S. Kodagoda, A. Alempijevic, and G. Dissanayake, “Efficient learning of motion patterns for robots,” in *The Australasian Conference on Robotics and Automation, December, 2009, (ACRA 2009)*, 2009.

[6] L. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proc. IEEE*, pp. 257 – 286, Feb. 1989.

[7] B. Fuglede and F. Topsoe, “Jensen-shannon divergence and hilbert space embedding,” in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, June 2004, p. 31 ff.

[8] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, “Learning motion patterns of people for compliant robot motion,” *International Journal of Robotics Research*, vol. 24, no. 1, 2005.

[9] S. M. LaValle and M. S. Branicky, “On the relationship between classical grid search and probabilistic roadmaps,” *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[10] R. Bohlin and L. Kavraki, “Path planning using lazy prm,” in *Robotics and Automation, 2000. Proceedings. ICRA ’00. IEEE International Conference on*, vol. 1, 2000, pp. 521–528 vol.1.

[11] J. Müller, C. Stachniss, K. O. Arras, and W. Burgard, “Socially inspired motion planning for mobile robots in populated environments,” in *International Conference on Cognitive Systems (CogSys’08)*, 2008.