Probabilistic Time-Dependent Models for Mobile Robot Path Planning in Changing Environments

Stefan Loibl Daniel Meyer-Delius Patrick Pfaff

***Abstract*— In the context of mobile robot path planning, a common strategy is to assume that the world is static and rely on heuristic approaches and obstacle avoidance to deal with the changes in the environment. When planning, not taking the potential changes of the environment into account usually leads to poor performances. In this paper we propose a probabilistic model that explicitly characterizes the traversability of the environment as a stochastic process. Furthermore, we present a path planning approach that exploits this traversability information to compute paths that minimize the expected travel time of the robot. Experimental results show that by explicitly modeling and reasoning about changes in the environment path planning performance can be improved.**

I. INTRODUCTION

Mobile robot systems must be able to deal with changes in the environment. While navigating, they need to avoid collisions and clear dynamic obstacles as well as replan their routes if the planned path is blocked. This reactive behavior, however, usually results in longer travel times. For this reason, to improve their navigation performance, mobile robots should be able to plan their paths trying to minimize the expected number of collision avoidance and replanning actions that may be encountered on the path. Consider, for example, the roadmap shown in Figure 1. When computing a pah from the *User* to the *Printer*, the shortest route would take the robot through the *Lab* which is frequently closed. To reduce the number of times that the robot has to replan because it found the lab closed, we could simply remove the path through the lab from the roadmap. This would cause the robot to always take a longer route independently of whether the lab is closed or not. A more founded approach would be to model the traversability of the path through the lab and decide whether to try the shortest route or not based on the expected traversability according to the model.

In this paper we propose a path planning approach that takes into account potential changes in the environment by explicitly modeling them in the roadmap. The goal is to find a path from an initial location to some goal location that minimizes the expected travel time. The road network is given as a directed graph where a node represents a location and an edge represents a path between two locations. We propose a model where each edge has an associated stochastic process that describes the traversability and the travel time at every time step of the underlying path. The path planning problem is formulated as a Markov decision process (MDP) where the states consist of joint locations and arrival times, and the cost function corresponds to the travel time. An optimal policy is found by exploiting the properties of our environment model and using dynamic programing.

All authors {stefan.loibl, daniel.meyer-delius, patrick.pfaff}@kuka.com are with the KUKA Laboratories GmbH, Ausgburg, Germany.

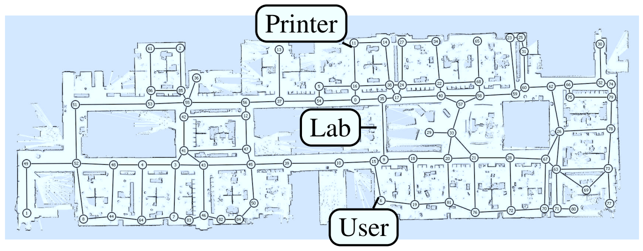

Fig. 1. The picture shows the grid map of an office environment extended with a roadmap. The shortest route from the *User* to the *Printer* would take the robot through the *Lab* which is frequently closed. The approach described in this paper explicitly takes the traversability of the *Lab* into account at the moment of planning a path.

The contribution of this work is twofold: a model of the environment that explicitly characterizes changes and a path planning approach that minimizes the expected travel time of the robot by explicitly taking into account potential changes in the environment. We propose a representation of the environment that probabilistically models the traversability and the travel time of the paths. Based on this model, we formulate the path planning problem as a MDP and describe an efficient approach for computing policies with minimum expected travel times. We evaluate our approach using synthetic and realistic data. The results show that path planning performance can be improved by explicitly modeling and reasoning about changes in the environment.

II. RELATED WORK

The path planning problem has been studied extensively in the past. Shortest paths in graphs with static costs can be found efficiently with Dijkstra's algorithm [1] or the heuristic A^*-Algorithm [2]. For partially known environments, Eyerich el al. [3] describe the stochastic variant of the Canadian Traveler's Problem, a path planning problem where some roads can be blocked by adverse weather. The objective is to find a policy with low expected travel cost. Similarly, the D^*-Algorithm described by Stentz et al. [4] finds shortest paths in partially known environments and is also able to cope with changing environments. If a discrepancy between the robots belief and the actual world is observed the costs are updated and the path is replanned. However, these changes are not taken into account beforehand.

Our work builds upon the work of Hall [5], who introduced the problem of finding the minimum expected travel time through stochastic time-dependent networks (STD networks). The travel time of each edge is a random variable whose distribution depends on the time. Several related problems and applications have been treated by Miller-Hooks and Mahamassani [6]. They defined efficient methods to find a least possible time path (i.e., a strategy yielding minimum travel time in the most favorable case). Pretolani [7] presented a directed hypergraph formulation of a STD network where the best strategy (time-adaptive route) corresponds to a minimum weight hyperpath and can be found in linear time with respect to the size of the hypergraph. He also discusses different criteria for ranking routes.

Several problems arise when using STD networks to model the environment of a mobile robot. In STD networks it is assumed that each node is related to a finite set of time steps at which it is possible to leave that node. Furthermore there is a finite time horizon t_{max} and it is possible to reach the goal node within t_{max}. Since a mobile robot can leave a node at any time step (provided that the current path is not blocked) it is hard to determine such a finite time horizon. One possibility would be to set it to a large number (it probably will not take a mobile robot several hours to move from one place in an office to another), but then it is not clear which cost has to be assigned to the nodes with time step t_{max} and to which other nodes they are connected. So although promising, STD networks can not directly be used to model the environment of a mobile robot. In spite of that our approach is closely related to STD networks.

Similar to our approach, Marthi [8] uses a known static topological map where edges can be either observed free or blocked and each edge state evolves according to a continuous time Markov chain. In the blocked case the state belongs to one of three types determining the average duration the edge stays blocked. Since the robot is not able to observe the type of the blocking obstacle the problem is modeled as a partially observable Markov decision problem (POMDP). The state space of a POMDP is exponential in the number of edges and in order to make the problem computationally tractable the so called abstract graph is introduced which reduces the number of edges in the graph. Whereas this approach can only be solved approximately our formulation of the problem as MDP permits an exact solution in polynomial time. The problem mentioned by Marthi [8], that by modeling the uncertainty as a MDP a robot trying to traverse a blocked edge would either leave immediately or wait forever when the edge stays blocked is solved by simultaneously learning the model's parameters while executing the plan and replanning in each node.

III. PLANNING IN CHANGING ENVIRONMENTS

Let $G = \langle V, E \rangle$ be a directed graph representing the roadmap of the environment, where V is a finite set of nodes (locations) and $E \subseteq V \times V$ a set of edges (roads). Each edge is annotated with a stochastic process characterizing the traversability of the edge at each time step and a probability distribution modeling the time needed to traverse the edge (given the edge is traversable). We formulate the path planning problem for G as an MDP $\langle S, A, P, c \rangle$, where

- $S = \{(i,t) \mid i \in V, t \in \mathbb{N}_0\}$ is the state space representing joint locations and arrival times t.
- $A(i,t) = N_G(i)$ is the action space for each state $(i,t) \in S$, where $N_G(i)$ corresponds to the set of all the neighbors of node i in graph G.
- $P\big((j,t') \mid (i,t), j\big)$ is the transition model that represents the probability of ending in location j at time t' after executing action j from location i at time t. It depends on the traversability of edge (i,j) and the travel time distribution. We assume that the only source of uncertainty in our transition model is the temporal component of the state, the spatial component is deterministic. Furthermore, our MDP is a so called left-right model since no transitions are allowed to states whose arrival times are lower or equal than the one of the current state.
- $c : S \times S \to \mathbb{N}_0 \cup \{\infty\}$ is the immediate cost function that represents the time needed for moving from one state to another. If the corresponding transition probability is zero, the cost is set to infinity.

Given a goal location $g \in V$, we are interested in computing an optimal policy π^* that minimizes the expected arrival time at the goal location for every state in S. The minimum expected arrival time $C^*(i,t)$ for state $(i,t) \in S$ is given by

$$C^*(i,t) = \min_{j \in A(i,t)} \sum_{t'=t+1}^{\infty} C^*(j,t') P\big((j,t') \mid (i,t), j\big) . \quad (1)$$

At a goal state (g,t), it is simply initialized to t, that is, $C^*(g,t) = t$.

A. Bounding the State Space

To efficiently compute the minimum expected arrival time using dynamic programming we assume that a time step t_m exists, such that $C^*(i, t_m + t) = C^*(i, t_m) + t$ for all $t \in \mathbb{N}_0$. This simply states that for time steps larger than t_m the time needed to reach the goal location does not depend on the current time, making the optimal policy time-independent.

Let X_t be a random variable representing the traversability of a given edge at time step t

$$X_t = \begin{cases} 0, & \text{if the edge is traversable at time } t, \\ 1, & \text{if the edge is not traversable at time } t. \end{cases}$$

We assume that the traversability of an edge is independent of the traversability of all other edges and model X_t as a first order homogeneous Markov process with transition matrix

$$A = \begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix} .$$

As t tends to infinity, X_t converges to a unique stationary distribution. The time needed to converge to this distribution is called mixing time. We follow the approach described in [9] to analytically compute the mixing time t_m^e for each edge e in E and define the mixing time for the complete graph as $t_m = max\{t_m^e \mid e \in E\}$.

B. Modeling the Travel Time of an Edge

We implicitly model the travel time for an edge $e = (i, j)$ (regardless of its traversability state) as a stochastic process Y_t that characterizes the probability of reaching the destination node j after a certain number of time steps. More specifically, $P(Y_t = t')$ is the probability of ending in location j at time t' when starting from location i at time t. The arrival time distribution $P(Y_t)$ is given by

$$P(Y_t) = \sum_{x \in \{0,1\}} P(X_t = x) P(Y_t \mid X_t = x) , \tag{2}$$

where $P(X_t)$ corresponds to the traversability state distribution of the edge and $P(Y_t \mid X_t)$ is the arrival time distribution of the edge conditioned on its traversability state. Note that $Y_t - t$ is the travel time of the edge. If the edge is traversable, we assume its travel time is time-independent and we use a (discrete) Beta distribution to model the conditional arrival time, that is, $P(Y_t \mid X_t = 0) \sim Beta(\alpha, \beta; a+t, b+t)$.

If the edge is not traversable, the conditional arrival time distribution is computed as

$$P(Y_t \mid X_t = 1) = \sum_{n=0}^{\infty} (p_{11})^n \, p_{10} \, P(Y_{t+n+1} \mid X_t = 0) . \tag{3}$$

The intuition behind (3) is that the state of the edge has to first change before the destination node can be reached. The expected arrival time for an edge at time step t is then given by

$$E\left[Y_t\right] = E\left[Y_t \mid X_t = 0\right] + \frac{P(X_t = 1)}{1 - p_{11}} , \tag{4}$$

where $E\left[Y_t \mid X_t = 0\right]$ represents the expected arrival time for the edge when traversable and $P(X = 1)/(1 - p_{11})$ is the expected number of time steps to wait for the edge to change its state from not traversable to traversable.

Since we are assuming that the only source of uncertainty in our transition model is the temporal component of the state, the arrival time distribution $P(Y_t)$ of an edge $e = (i, j)$ corresponds to the transition model between the nodes associated to the edge, that is

$$P\big((j,t') \mid (i,t), j\big) = P(Y_t = t') . \tag{5}$$

C. Minimum Expected Arrival Time

To compute the minimum expected arrival time $C^*(i,t)$ for each state in S we first compute the minimum expected arrival time $C^*(i,t_m)$ for each state at time t_m as

$$C^*(i,t_m) = \min_{j \in A(i,t)} \sum_{t=t_m+1}^{\infty} (C^*(j,t_m) + t) P(Y_{t_m} = t). \tag{6}$$

The equation above is obtained from (1) by replacing the motion model with the arrival time distribution of the corresponding edge and from the fact that $C^*(i,t_m+t) = C^*(i,t_m) + t$. Due to spatial constraints we omit the complete derivation of the formula, but after a series of straight forward algebraic operations, rearranging the indexes of the sums and applying the definition of the expected value for a random variable, we obtain the following formula for computing the expected arrival time for a state (i,t_m):

$$C^*(i,t_m) = \min_{j \in A(i,t)} C^*(j,t_m) + E\left[Y_{t_m}\right] . \tag{7}$$

As defined at the beginning of the section, for a goal location g, $C^*(g,t_m) = t_m$. Setting the cost of each edges to its expected travel times $E\left[Y_{t_m}\right] - t_m$ at time step t_m, Dijkstra's algorithm can be used to efficiently compute the expected arrival time at the goal location for all remaining non-goal states at time t_m.

Assuming that the expected arrival times at time $t = t_m$ have already been computed, the expected arrival times for time steps $t < t_m$ are computed according to (1) by iterating backwards in time, starting at time step $t_m - 1$. At each state (i,t) in S only the expected arrival times at time steps larger than t are needed, which are either already computed or given by $C^*(i,t_m+t) = C^*(i,t_m) + t$.

The value of the series (1) can be exactly computed by dividing it into a finite sum A and a series B, which can be computed analytically:

$$C^*(i,t) = \min_{j \in A(i,t)} \underbrace{\sum_{t'=t+1}^{T} C^*(j,t') P(Y_t = t')}_{A} + \underbrace{\sum_{t'=T+1}^{\infty} C^*(j,t') P(Y_t = t')}_{B} . \tag{8}$$

The time step T is the maximum between t_m and t^*+1 where t^* is the largest value in the support of $P(Y_t \mid X_t = 0)$.

For all summands of the first part A, the minimum expected arrival time is already computed. After a series of straight forward algebraic operations we obtain the following formula for computing the value of the second part B:

$$B = P(X_t = 1) P(Y_t = t^*+1 \mid X_t = 1) \frac{p_{11}^{T-t^*}}{1 - p_{11}} \cdot \left(C^*(j,t_m) + T + 1 + t - t_m + \frac{p_{11}}{1 - p_{11}}\right). \tag{9}$$

D. The Path Planning Algorithm

The pseudo-code of our path planning approach is shown in Algorithm 1. The mixing time t_m for the complete road map is computed in line 2. In line 3 the cost for the goal states is initialized. The optimal cost for all states at time step t_m is initialized to the stationary cost according to (7) in line 4. After that we go backwards from time step $t_{max} - 1$ to time step 0 and compute for each node that is not a goal the optimal cost $C^*(i,t)$ and policy $\pi^*(i,t)$ by comparing the cost of all neighbors $N_G(i)$ of the current node i in the graph G (lines 8 – 10).

The algorithm has a general time complexity of $O(|E| \cdot t_m^2)$, where $|E|$ is the number of edges. This can easily be seen from lines 5, 6, 8 and 9. Note that lines 6 – 15 can be run in parallel.

Algorithm 1: Main algorithm

```
input : The graph $G = (V, E)$ and the goal node $g \in V$
output: The optimal cost $C^* : V \times \{0, \ldots, t_m\} \to \mathbb{R}_0^+$ and
        the optimal policy $\pi^* : V \times \{0, \ldots, t_m\} \to V$
1  begin
2      $t_m := \max\{t_m^e \mid e \in E\}$;
3      $C^*(g,t) := t$ for all time steps $t \leq t_m$;
4      Compute $C^*(i, t_m)$ according to (7);
5      for $t = t_m - 1$ to 0 do
6          foreach $i \in V \setminus \{g\}$ do
7              $C^*(i,t) := \infty$;
8              foreach $j \in N_G(i)$ do
9                  $c := \sum_{t' \in \mathbb{N}_0} P(Y_t = t') C^*(j, t')$;
10                 if $c < C^*(i,t)$ then
11                     $C^*(i,t) := c$;
12                     $\pi^*(i,t) := j$;
13                 end
14             end
15         end
16     end
17 end
```

IV. EXPERIMENTAL EVALUATION

To evaluate our approach we performed experiments on different environment models. These models consist of two components: a directed graph and the traversability model of each edge in the graph. This traversability model, in turn, consists of a Markov process characterizing the traversability state of the edge over time and a Beta distribution corresponding to the travel time for the edge.

We generated random graphs out of regular grid networks. Starting from a random spanning tree on the grid, we randomly selected further edges from the underlying network to produce graphs with different edge densities.

Since the traversability model of an edge characterizes multiple different aspects of the underlying road, we defined three traversability models according to the edge's probability of being traversable and remaining traversable: rarely traversable with prolonged traversability (*semi-static*), frequently traversable with prolonged traversability (*semi-dynamic*), and frequently traversable with brief traversability (*dynamic*). Additionally, we also considered edges that were always traversable (*static*). In the *semi-static* case the state of an edge changes rarely, whereas in the *dynamic* case edge state transitions occur frequently. The model of an edge was assigned by randomly sampling from a discrete distribution over the four different traversavility models. The overall traversability of the environment was controlled by biasing this distribution in favor of one of the classes.

A. Path Planning In Changing Environments

Figure 2 compares the performance of different path planning approaches in different environment models. In particular, we compare our path planning approach (*STDN*) against different common approaches with respect to arrival times at the goal. We evaluated Dijkstra's algorithm using,

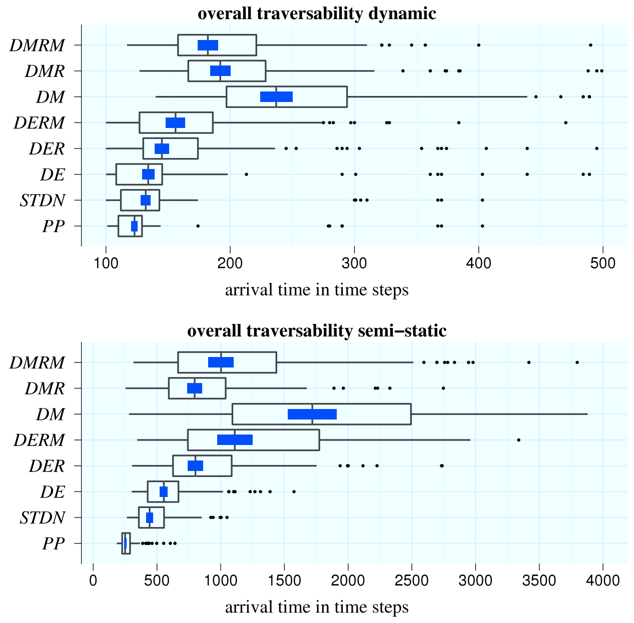

Fig. 2. Arrival times of different planners for the same environment but with different overall traversability. The planner *PP* has complete knowledge of the environment, *STDN* is our approach, the three planners *DE*, *DER*, and *DERM* use expected values of our model to compute shortest paths, and the planners *DM*, *DMR*, and *DMRM* use minimal travel times (*R* stands for replanning and *M* for memory, i.e. remembering untraversable edges).

as static costs of the edges, the expected arrival time $E[Y_{t_m}]$ at time t_m (*DE*) and the minimal arrival time (*DM*). If during execution, an edge of the path is not traversable, these two approaches wait until the edge changes its state. We also considered variations of the *DE* and *DM* approaches, namely *DER* and *DMR*, that did replan when a non-traversable edge was found. To mitigate the oscillating behavior typical of replaning approaches that don't keep record of their actions, we also considered extensions of the *DER* and *DMR* approaches, namely *DERM* and *DMRM*, that kept track of the traversability state of the edges. Finally, as base line, we considered a perfect planner (*PP*) that had complete knowledge of the environment at every point in time.

We assumed that at any given state, only the traversability of the outgoing edges could be observed. For the *STDN* Planner, the traversability of the edges is initially unknown. Additionally, we assumed that the traversability state of an edge did not change while it was being traversed.

For all environments we selected a start and goal node set at opposite corners of the underlying grid network and performed 30 runs for each approach. The boxplots in Figure 2 correspond to the arrival times at the goal for two random environments with different traversability models generated out of a 10×10 grid networks. The plots show the 95% confidence interval of the median of the arrival times. In *dynamic* environments, where state transitions occur frequently, the mixing time of the environment is in general rather small reducing the advantage of the *STDN* planner against that of the *DE* planner. A Wilcoxon rank-sum test revealed that the arrival times of our approach were significantly lower than the ones of all other approaches on a 1% level in *semi-static*

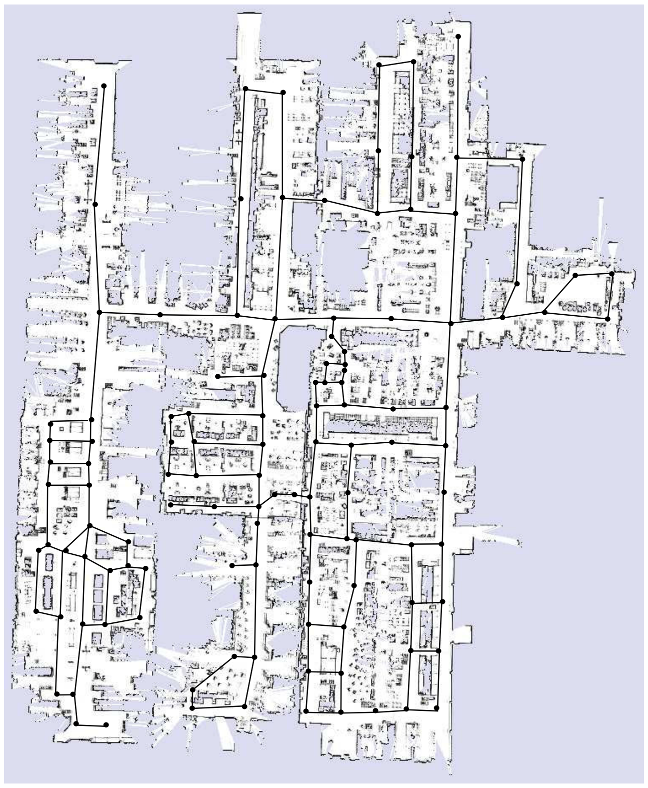

Fig. 3. Grid map and roadmap for the KUKA production site in Augsburg with a size of 300×200 meters.

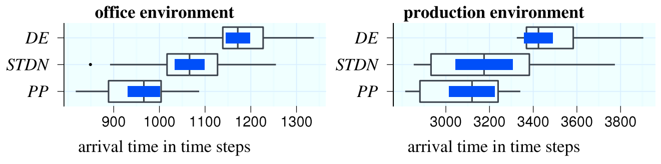

Fig. 4. Arrival times in the example environments office (left) and production hall (right) depicted in Figure 1 and Figure 3. We compare the performances of the perfect planner (*PP*), our approach (*STDN*), and the Dijkstra planner using expected values of our model (*DE*).

and in *semi-dynamic* environments. Consistent results were obtained on numerous other environments of different sizes, densities and dynamics.

Real World Topology

We also compare the performance of the different path planning approaches in more realistic environments. We manually created road networks using as reference a floor map of an office environment and an industrial production hall. The traversability models of the edges were assigned according to the typical dynamic of the underlying road. Figures 1 and 3 show the floor maps with the road network overlaid on top for these two environments. To make the scenario more realistic, instead of a single goal, a sequence of goal nodes was used in each run. The arrival times of the planners *PP*, *STDN*, and *DE*, at the last goal node are shown in Figure 4. Just as in the previous experiment the planning approach presented in this paper is significantly better than the Dijkstra planners.

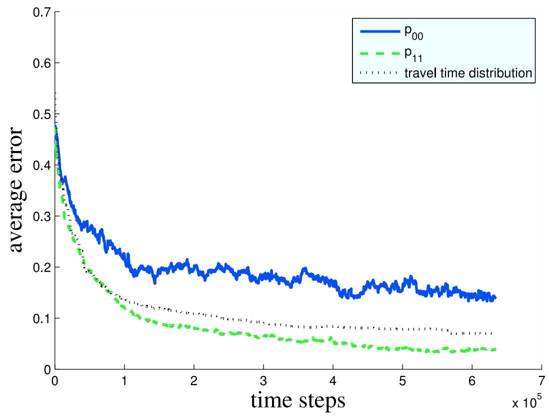

Fig. 5. Online learning of the model parameters. The curves correspond to the total variation distance for the state transition probabilities p_{00} and p_{11}, and the travel time distribution, averaged over all edges in the network.

TABLE I

ARRIVAL TIMES WHEN USING THE PARAMETER ESTIMATES OBTAINED AFTER 100 (*M1*), 20000 (*M2*), 40000 (*M3*), AND 60000 (*M4*) TIME STEPS.

	DMRM	*M1*	*M2*	*M3*	*M4*	*STDN*
mean	2057	2349	1939	1655	1655	1065
std.	718.30	1654.97	1584.07	1071.82	307.98	90.37

B. Estimating the Model Parameters

The parameters of our model can be learned from data collected while moving through the environment. The state transition parameters of the traversability model can be estimated online using the approach described in [10] and the parameters of the travel time distributions can be updated using the method-of-moments [11] approach.

The curves in Figure 5 show how the parameters of the model are learned while observing the environment. In simulation, we performed 3000 runs using our path planning algorithm in the office environment described in the previous experiment, each time using a randomly selected node as goal. The traversability state and state transition probabilities of each edge were set initially to a uniform distribution, and the travel time distribution was set to a point mass distribution at the minimum travel time in the support of the underlying travel time distribution.

The curves correspond to the total variation distance (see [12]) for the state transition probabilities p_{00} and p_{11}, and the travel time distribution, averaged over all edges in the network. The accuracy of the learned parameters for an edge not only depends on how often the edge is observed but also on how long it is observed. During the first runs, the parameter estimates are rather rough and the generated paths frequently result in longer waiting times at untraversable nodes. This, in turn, leads to frequent and longer observations of the adjacent edges. This explains the rapid decrease of the error at the beginning of the learning process. The better the parameter estimates, the less frequent and shorter the waiting times.

Table I shows the performance of our approach when using the parameter estimates obtained after 100 (*M1*), 20000 (*M2*), 40000 (*M3*), and 60000 (*M4*) time steps. We compared the arrival times against those obtained when using the ground truth model (*STDN*). We also compared the results of the *DMRM* approach - the best approach without any knowledge about the underlying dynamics.

As can be seen in the figure, the performance of our approach increases as the model parameters converge to the ground truth. Additionally it can be seen that the number of outliers (in both directions) also decreases. This outliers are caused by uninformed or "risky" decisions, that lead to short paths that are often untraversable.

C. Computational Time

TABLE II

RUNTIME OF OUR APPROACH IN MILLISECONDS FOR ENVIRONMENTS OF DIFFERENT SIZES AND USING DIFFERENT VALUES FOR t_m.

	$t_m = 1$	$t_m = 10$	$t_m = 100$	$t_m = 1000$
5×5	3 (±5)	14 (±4)	125 (±5)	2202 (±17)
10×10	17 (±4)	68 (±8)	593 (±7)	10227 (±156)
15×15	47 (±0.4)	168 (±8)	1530 (±182)	24616 (±1069)
20×20	95 (±5)	333 (±8)	2794 (±63)	46411 (±1003)

We also evaluated our approach in terms of computational time. The experiments where performed on a standard PC with a dual-core processor. Table II summarizes the the average runtime (and standard deviation) in milliseconds for 30 runs in environments of different sizes. Since the complexity of the algorithm depends largely on the value of the upper limit t_m for the mixing time, we also compared the computational times for different values of t_m. As expected, runtime of the algorithm scales linearly with the number of edges and quadratically with t_m. Truncating the value of mixing time is equivalent to increasing the convergence threshold at the moment of computing the mixing time (see [9]). Practically t_m represents the planning horizon: after t_m we assume that the stationary state of the environment has been reached. In general, the closer t_m is to its theoretical value, the better the performance of the planner. As reference for the computational times, the average runtime for the office (87 nodes) and production environments (118 nodes) for $t_m = 100$ were, 312 and 403 milliseconds respectively.

V. CONCLUSIONS

In this paper we presented an approach to model the environment based on the idea of road networks which probabilistically characterize the traversability and travel time of the roads. We described the traversability of each road as a first order homogeneous Markov process and used a discrete Beta distribution to model the travel times. This extension of the standard road networks commonly used for path planning allows us to represent relevant time-depended information of the environment.

Based on this model, we proposed a path planning approach that takes into account potential changes in the environment to minimizes the expected travel time. We formulated the path planning problem as a Markov decision process where the states consist of joint locations and arrival times. We first computed the minimum expected travel times in the model at its stationary state and then used dynamic programming to compute the expected travel times for all prior time steps.

We implemented and tested our approach using synthetic and realistic data. The results show that the information about the environment encoded in our model can lead to significant path planning improvements. The results also indicate that, by considering both the current and predicted state of the environment, our path planning algorithm outperformed approaches that assumed a static world. Finally we described an approach for learning the parameters of the model online, and showed that our approach was robust against inaccurate models parameters. Being able to learn the model parameters while navigating the environment together with the ability to explicitly model and reason about potential changes in the environment makes our approach particularly attractive for robotic systems that need to operate over extended periods of time.

ACKNOWLEDGMENT

This work has been partially supported by the European Commission under contract number FP7-260026-TAPAS.

REFERENCES

[1] E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerical Mathematics*, vol. 1, pp. 269–271, 1959.

[2] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100 –107, july 1968.

[3] P. Eyerich, T. Keller, and M. Helmert, "High-Quality Policies for the Canadian Traveler's Problem," in *Third Annual Symposium on Combinatorial Search*, AAAI. AAAI Publications, aug 2010, pp. 147–148.

[4] A. Stentz and I. C. Mellon, "Optimal and efficient path planning for unknown and dynamic environments," *International Journal of Robotics and Automation*, vol. 10, pp. 89–100, 1993.

[5] R. W. Hall, "The Fastest Path through a Network with Random Time-Dependent Travel Times," *Transportation Science*, vol. 20, no. 3, pp. 182–188, Aug. 1986.

[6] E. D. Miller-Hooks and H. S. Mahmassani, "Least Expected Time Paths in Stochastic, Time-Varying Transportation Networks," *Transportation Science*, vol. 34, no. 2, pp. 198–215, May 2000.

[7] D. Pretolani, "A directed hypergraph model for random time dependent shortest paths," *European Journal of Operational Research*, vol. 123, p. 2000, 1998.

[8] B. Marthi, "Robust navigation execution by planning in belief space," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[9] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," in *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, Toronto, Canada, July 2012.

[10] G. Mongillo and S. Deneve, "Online learning with hidden markov models," *Neural Comput.*, vol. 20, no. 7, pp. 1706–1716, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1162/neco.2008.10-06-351

[11] L. P. Hansen, "Large sample properties of generalized method of moments estimators," *Econometrica*, vol. 50, no. 4, pp. 1029–54, July 1982. [Online]. Available: http://ideas.repec.org/a/ecm/emetrp/v50y1982i4p1029-54.html

[12] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times*. American Mathematical Society, 2006.