Safe Mobile Robot Motion Planning for Waypoint Sequences in a Dynamic Environment

Janko Petereit, Thomas Emter and Christian W. Frey
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB
Karlsruhe, Germany
{janko.petereit, thomas.emter, christian.frey}@iosb.fraunhofer.de

***Abstract*—Safe and efficient path planning for mobile robots in dynamic environments is still a challenging research topic. Most approaches use separate algorithms for global path planning and local obstacle avoidance. Furthermore, planning a path for a sequence of goals is mostly done by planning to each next goal individually. These two strategies generally result in sub-optimal navigation strategies.**

In this paper, we present an algorithm which addresses these problems in a single combined approach. For this purpose, we model the static and dynamic risk of the environment in a consistent way and propose a novel graph structure based on a state $\times$ time $\times$ goal lattice with hybrid dimensionality. It allows the joint planning for multiple goals while incorporating collision risk due to dynamic and static obstacles. It computes hybrid solutions which are part trajectory and part path. Finally, we provide some results of our algorithm in action to prove its high quality solutions and real-time capability.

I. Introduction

Mobile robots are more and more employed in manufacturing tasks, respectively, industrial environments in general. There is a strong trend away from automated guided vehicles (AGV) and towards fully autonomous robots. The working area of the robot is no longer restricted to small, well defined regions; instead, the robot is explicitly allowed to share its working area with human co-workers. It may even be desired to actively collaborate with humans to accomplish a given task. All industrial environments share the property that they are not static but may change over time. For example, deposited boxes or parked forklifts require the robot to constantly sense its environment. This is especially true in the presence of dynamic obstacles like humans or other mobile systems.

The successful application of mobile robots requires not only a safe but also an efficient operation. Typically, the robot needs to travel between several working stations in order to accomplish its task. In general, it is beneficial if the robot is able to determine the optimal orientation to arrive at the next goal by itself depending on the succeeding goals and the instantaneous configuration of the environment.

Most planning approaches utilize two separate algorithms for the global path planning and local obstacle avoidance. Two global path planning algorithms which have been successfully applied for the DARPA Urban Challenge are the Hybrid A* algorithm [1] and the planning in state lattices [2]. Both algorithms search for an optimal path by successively concatenating short motion primitives until reaching the goal. The Hybrid A* algorithm operates simultaneously in a continuous and discrete state space, which results in the reachable set forming a tree in the state space. On the contrary, state lattice planners utilize specifically generated motion primitives which restrict the motion of the system to regular discrete points in the state space, giving these state lattice algorithms their name.

In their original version, these algorithms were restricted to the planning of merely *kinematically* feasible paths. However, in order to guarantee smooth trajectories for the mobile robot and to be able to operate efficiently in a dynamic environment, it is advisable to extend the planning to *dynamically* feasible paths. Of course, this results in an increased dimensionality of the search space and higher computational burden. In the past, especially probabilistic approaches like RRTs [3] and PRMs [4] have become established for high-dimensional planning problems. However, generally, these algorithms seek to find *any* path from the start to the goal and provide only guarantees on probabilistic completeness. Furthermore, it is difficult to incorporate additional information like terrain quality into the planning. To overcome these limitations, state lattice planners have been extended to search spaces with higher dimensionality to plan optimal dynamically feasible paths [5].

A. Problem Statement

Especially in crowded dynamic environments like a collaborative manufacturing scenario, a global path planning, which is unaware of dynamic obstacles, combined with a subsequent local obstacle avoidance leads to sub-optimal results. Therefore, it is important to consider dynamic obstacles already during the global path planning phase. Particularly in narrow spaces, in which robots and humans work side by side, motion planning of mobile robots must also seek to minimize the collision risk as much as possible. This applies to the close vicinity of the robot (risk of collision due to dynamic obstacles) as well as to the farther regions (parts of the environment with increased collision risk, e.g., near static obstacles or common routes of human co-workers).

Another challenge for mobile robots in a manufacturing setting is the overall efficiency of the executed work sequence. Commonly, the task of the robot may contain a sequence of multiple future goal regions. However, generally, neither the optimal position nor the optimal heading for reaching the next goal region may be known in advance, in fact, the optimal way of reaching the next goal region is highly dependent on the

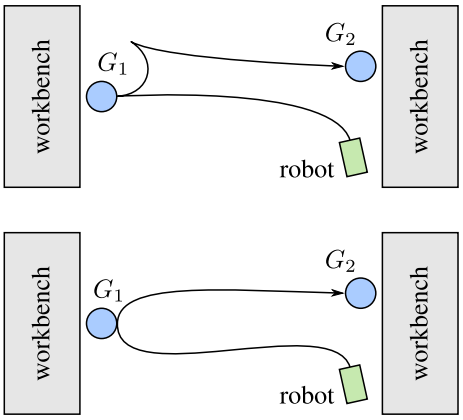


Fig. 1. Top: Greedy planning to goal region G_1 while ignoring G_2 results in an unnecessary turning maneuver on the subsequent way to goal region G_2. Bottom: Optimal arrival at G_1 due to forward-looking planning.

subsequent goal region (see Fig. 1). Therefore, an optimal path planning strategy must be able to explicitly take multiple goal regions into account.

B. Proposed Solution

In this paper, we present a novel concept which solves the path planning problem in the context of narrow spaces potentially containing many dynamic objects. In order to keep this complex task computational tractable, we employ a state $\times$ time lattice with variable dimensionality to exploit the fact that the required accuracy of the solution in the near range of the robot is higher than it is in farther regions. Our planning algorithm provides a solution of hybrid nature. In the immediate future, it explicitly contains a time component, thus providing a *trajectory*. With progressing time, we relax the requirements on the solution, so that it represents merely a *kinematically* feasible path. Furthermore, the planning will be done collectively for the next two goal regions to allow the robot to arrive at the first goal region in an optimal manner with respect to its subsequent travel.

C. Related Work

Our planning concept is closely related to time-bounded lattices [6], however, the algorithm proposed therein directly reduces to a simple 2D grid search (which ignores any kinematic constraint) after an initial planning phase in a full-dimensional state $\times$ time lattice. Planning with adaptive dimensionality has also been investigated in the past [7] but it is restricted to mere *path* planning – a time parametrization is not considered. To cope with moving obstacles, PRMs have been extended to state $\times$ time spaces as well [8] but still suffer from only probabilistic guarantees. Hierarchical planners [9] offer good responsiveness in the close vicinity of the robot but their solutions are not globally optimal in a rapidly changing environment as the quality depends on the performance of the global planner.

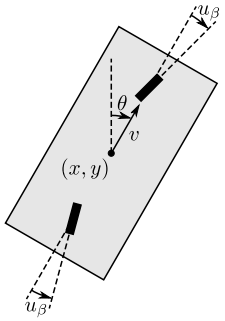


Fig. 2. Simplified kinematics of a mobile robot with four-wheel steering

D. Robot Motion Model

To describe the motion of the mobile robot, we use the following system of differential equations, which can be derived from the kinematics depicted in Fig. 2.

$$\dot{x} = v \cos\theta \tag{1}$$
$$\dot{y} = v \sin\theta \tag{2}$$
$$\dot{\theta} = c\, v \tan u_\beta \tag{3}$$
$$\dot{v} = u_a \tag{4}$$

The variables x and y denote the robot's position, θ its orientation and v its translational velocity. The constant c is twice the inverse of the wheelbase. The system's inputs are given by the translational acceleration u_a and the steering angle u_β. Of course, the system equations could be extended to include further derivatives like jerk or steering velocity. However, the chosen (x, y, θ, v) state representation suffices for the simulation of smooth trajectories, and additionally allows for a very efficient solution of the differential equations (1)–(4). This is due to the fact, that if the inputs u_a and u_β are held constant during a time interval T, the system of differential equations can be analytically integrated resulting in the following time-discrete equations.

$$x_{k+1} = x_k + \begin{cases} \dfrac{\sin\theta_{k+1} - \sin\theta_k}{c \tan u_\beta} & \text{if } u_\beta \neq 0 \\ \left(\frac{1}{2} u_a T + v_k\right) T \cos\theta_k & \text{if } u_\beta = 0 \end{cases} \tag{5}$$

$$y_{k+1} = y_k + \begin{cases} \dfrac{\cos\theta_k - \cos\theta_{k+1}}{c \tan u_\beta} & \text{if } u_\beta \neq 0 \\ \left(\frac{1}{2} u_a T + v_k\right) T \sin\theta_k & \text{if } u_\beta = 0 \end{cases} \tag{6}$$

$$\theta_{k+1} = \theta_k + \left(\tfrac{1}{2} u_a T + v_k\right) c\, T \tan u_\beta \tag{7}$$
$$v_{k+1} = v_k + u_a T \tag{8}$$

These recurrence equations can be used for the exact simulation of the robot's motion for an arbitrary time interval of length T. We will exploit this extensively for the generation of the control set for the high-dimensional lattice (see section III-A).

II. Collision Risk

For navigation in an environment with increased collision risk, it is crucial to model this risk appropriately and explicitly consider it during the path planning. For our algorithm we take the risk for collisions with static obstacles p_s as well as the risk caused by moving obstacles p_d into account.

A. Static Obstacles

To enable a safe, yet still efficient navigation in narrow spaces, we model static obstacles like walls in a probabilistic manner. For this purpose, we first compute the Euclidean distance transform of the grid map which represents the environment using the algorithm described in [10]. Afterward, we calculate a collision risk for each cell of the map by

$$p_s = \begin{cases} 1 & \text{if } d \leq \rho \\ e^{-\alpha(d-\rho)^2} & \text{else} \end{cases} . \tag{9}$$

The variable d corresponds to the distance to the next blocked cell resulting from the Euclidean distance transform (multiplied by the cell size for a conversion to meters). ρ is the radius of the robot, i.e., half of its diagonal. The parameter α controls the decay rate of the risk with increasing distance from the obstacle. For example, a decay rate of $\alpha = 4/\mathrm{m}^2$ lets the collision risk drop to approximately 2 % after 1 m.

Additionally, further zones of the environment with increased collision risk (e.g., a priori known routes commonly used by forklifts) can be easily integrated into this framework by directly adjusting the collision risk of the corresponding cells.

B. Dynamic Obstacles

There are a lot of strategies for taking collisions with dynamic obstacles into account. In [11] an approach is presented in the context of evacuees which occupy a fixed personal space around them; the future position of the moving obstacles is assumed to be exactly computable. However, in a real scenario, the future trajectories of dynamic obstacles cannot be determined in advance. Instead, they have to be predicted on the basis of past measurements with explicitly taking into account the uncertainty of the underlying prediction model. Therefore, it is inevitable to incorporate this probabilistic description of future trajectories into the path planning. For this purpose, we used the approach presented in [6] and refined it using some beneficial approximations.

The algorithm assumes that there is a predictor available which propagates the state and uncertainty of the dynamic obstacle over time. This could be achieved using off-the-shelf tracking algorithms like a Kalman filter with a simple constant velocity model, but more sophisticated approaches like multi-hypothesis techniques are also possible.

We assume that the probability of being at a certain location at a given time can be described using a two-dimensional normal distribution. The collision risk p_d depends on the position, orientation, size, shape, and uncertainty of the dynamic obstacle as well as on the position and orientation of the robot. Even if all obstacles were assumed to have the same shape, the collision probability would thus depend on a 10-dimensional parameter vector. As a multi-variate normal distribution cannot be integrated analytically over an arbitrary region, a 10-dimensional look-up-table would be necessary, which is intractable. Therefore, we decided to approximate the collision probability by introducing the following simplification. We reduce the dynamic obstacle to a point and in exchange assume that the robot has a circular shape with radius r that is the sum of the robot's and obstacle's half diagonals. With this assumption, the orientations of the obstacle and the robot become irrelevant and the collision probability depends only on the relative position $\delta x, \delta y$ of obstacle and robot, the propagated uncertainty of the position of the obstacle $\mathbf{\Sigma}$, and the radius r, which still results in six parameters. In order to avoid the generation of a 6-dimensional look-up-table (which would be only possible for a relatively coarse quantization), we, in contrast to [6], compute the collision probability on the fly during the path planning for all visited points by numerically solving the following integral using an approximate trapezoidal integration scheme.

$$p_d = \iint_R \frac{1}{2\pi\sqrt{|\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}\begin{bmatrix} x & y \end{bmatrix} \mathbf{\Sigma}^{-1} \begin{bmatrix} x \\ y \end{bmatrix}\right) dx\, dy \tag{10}$$

$$R = \left\{(x, y) : (x - \delta x)^2 + (y - \delta y)^2 \leq r^2\right\} \tag{11}$$

Once calculated, we store the probability in a 3-dimensional cache with (x, y, t) as its key since the collision risk is exclusively dependent on the position and time of the expanded nodes during each path planning iteration.

C. Overall Collision Risk

Having computed the two separate collision probabilities p_s and p_d, we can calculate the combined collision probability

$$p = 1 - (1 - p_s)(1 - p_d) . \tag{12}$$

assuming that the events of a collision are independent. Depending on the length of a motion primitive, this procedure might be repeated for several points along the corresponding trajectory, which leads to the following overall collision risk for the execution of a specific motion primitive.

$$p_m = 1 - \prod_i (1 - p_i) \tag{13}$$

In the presence of multiple dynamic obstacles, the collision probabilities are combined by computing the integral (10) for each obstacle j and the combined collision probability (12) becomes

$$p = 1 - (1 - p_s) \prod_j (1 - p_{d,j}) . \tag{14}$$

III. Trajectory/Path Planning

For efficient planning of maneuvers in narrow busy spaces, we present a novel algorithm that bases upon a state × time × goal lattice with variable dimensionality. The algorithm consists of three parts, which are described in detail in the following sections.

1) Construction of the state $\times$ time lattice with variable dimensionality defined by the corresponding motion primitives (precomputed off-line).
2) Augmentation of search space with a "goal" dimension, resulting in a state $\times$ time $\times$ goal lattice.
3) Graph search in the lattice, which provides a hybrid solution consisting of a full-dimensional trajectory in the near future and a kinematically feasible path in the far future.

A. State $\times$ Time Lattice Construction

First, the dimensionality of the full-dimensional lattice L_0 has to be determined. In accordance with the system model (1)–(4), we chose the dimensions to be x, y, θ, v, and t, which is sufficient for the generation of smooth trajectories. Now, the corresponding motion primitives which describe the transition between the points of the lattice, have to be generated.

There is a vast variety of approaches for the construction of these motion primitives. Bicchi et al. have shown in [12] which conditions a system and its inputs must satisfy so that its reachable set forms a lattice. However, for mobile robot applications this generally has the disadvantage of non-uniform heading discretization. Pivtoraiko et al. [2] use the approach presented by Kelly and Nagy [13], which is based on curvature polynomials, which approximate the vehicle motion.

We chose a completely different approach which enables a very efficient generation of high-dimensional motion primitives for the system model (1)–(4) while still providing exact solutions. For this purpose, the time-discrete system model (5)–(8) is used to run a large number of simulations of the robot's motion for randomly sampled piecewise constant inputs u_a and u_β.

For all motion primitive candidates that end at a state which belongs to the same point in the lattice, only the motion primitive which minimizes the quantization error is preserved. In an additional post processing step we drop all motion primitives that can be reconstructed from shorter motion primitives.

As translational invariance of the state $\times$ time lattice is only given for the x, y, and t dimensions, all the steps described above have to be performed for each combination of θ and v to generate a unique corresponding set of motion primitives. However, if the number of discrete orientations is chosen to be a multiple of 4, one can take advantage of the rotational symmetry, which considerably reduces the computational effort.

After having constructed the full-dimensional lattice L_0 with the corresponding set of motion primitives M_0, we project L_0 and M_0 onto the (x, y, θ) sub-space to obtain the low-dimensional lattice L_1 and its corresponding set of still kinematically feasible motion primitives M_1. Finally, we define transitions from L_0 to L_1 by connecting all states $(x, y, \theta, v, t) \in L_0,\ t > t_0$ with the states reachable by the motion primitive $m \in M_1$ that starts at the corresponding projected state $(x, y, \theta) \in L_1$. The parameter t_0 determines for which time horizon dynamic obstacles should be considered during the planning. This threshold can be either a fixed value or computed adaptively depending on the robot motion and the prediction of the motion of the dynamic obstacles (cf. [6]).

B. Augmentation with "Goal" Dimension

In section I-A, we already mentioned the importance of considering at least the next two goal regions during the planning to enable the robot to determine its optimal orientation at the next goal region. This can only be achieved by explicitly adding the goal regions to the search space. In the following we build on the concept which we proposed in [14] but formulate it in a more general way. For this purpose, we extend both L_0 and L_1 with an additional "goal" dimension. Thus, we obtain the lattice L_0' containing (x, y, θ, v, t, G) states and the lattice L_1' containing (x, y, θ, G) states. The discrete state variable $G \in \{G_i : i = 1, \ldots, n\}$ assigns each state to the segment that leads to the respective goal region. In practice, $n = 2$, i.e, planning for the next two goal regions, is usually sufficient. Furthermore, we define the transitions between the segments as follows. If a state with $G = G_k, k \in \{1, \ldots, n-1\}$ falls into the corresponding goal region, i.e., $(x, y) \in G_k$, we set the end state of the motion primitive to belong to G_{k+1}. Using this procedure, we obtain the new sets of motion primitives M_0' and M_1'.

C. Graph Search

The search space is completely defined by the procedure described in the previous section. The corresponding graph is constructed during the search as needed using the motion primitives from M_0' and M_1'.

As on the one hand especially the first seconds of the search space are relatively high-dimensional but on the other hand a fast generation of (possibly sub-optimal) trajectories is required to allow a safe navigation in the presence of dynamic obstacles, anytime graph search algorithms are particularly well suited for this type of problem.

For our implementation of the presented algorithm, we have chosen to use the ARA* algorithm (Anytime Repairing A*, [15]). It first starts a weighted A* search to find a valid, but possibly ϵ-sub-optimal solution. For this purpose, it uses a heuristic, which is inflated by a factor ϵ, to determine the order of the node expansion. If a solution is found and there is still enough computation time left, the search starts again using a decreased inflation factor ϵ. This step may be repeated several times. To speed up planning, ARA* uses intermediate results from the previous iteration. The integration of such an anytime graph search with our sequence of lattices is straightforward.

As especially in a highly dynamic environment a fast replanning is very important, it would be interesting to explore the potential of employing a replanning algorithm like D* Lite [16]. However, this is a rather challenging task because all these replanning algorithms generally search backwards from the goal to the robot's current position. This conflicts with our approach, which uses the accumulated time of a node since the start to determine the transition between two lattices. Of course, this time is not available when searching backwards.

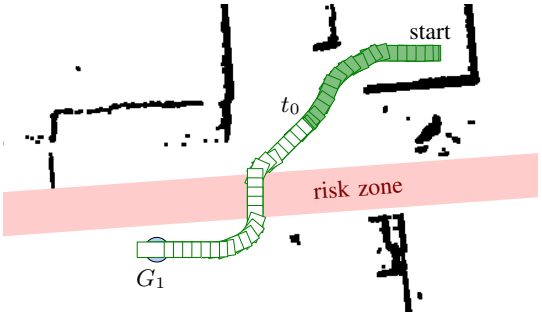


Fig. 3. Consideration of static risk zones. The planner crosses the risk zone as fast as possible instead of choosing the shorter diagonal path. The dark-green tiles of the planning result correspond to states from L'_0, i.e., a *trajectory* with time component; the light tiles correspond to the mere *path* part, i.e., states from L'_1.

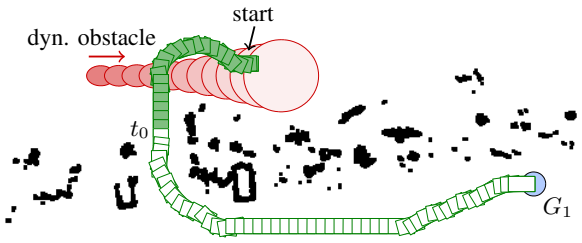


Fig. 4. Consideration of risks due to dynamic obstacles. The robot (green) faces an oncoming obstacle (red) and plans an avoidance maneuver according to the predicted motion of the obstacle. Again, the dark-green tiles correspond to the trajectory part, the light tiles to the path part. The red ellipses depict the obstacles uncertainty, the lightness increases with time.

D. Cost Function

We split the cost $g(s')$ for reaching the state s' into two separate parts, namely, travel and risk cost. First, we compute a travel cost $c_m(s, s')$ for the used motion primitive that connects the state s' with its predecessor s. $c_m(s, s')$ consists of the traveled distance $d_m(s, s')$ and the used time $t_m(s, s')$ weighted by a factor λ_t.

$$c_m(s, s') = d_m(s, s') + \lambda_t \, t_m(s, s') \tag{15}$$

The weighting factor λ_t can be chosen from a wide range of reasonable values. For $\lambda_t < 1$ the cost due to the traveled distance outweighs the cost associated with the needed time. If λ_t is chosen to be much larger than 1, the traveled distance becomes negligible and the focus is shifted to a time-optimal planning.

The first component of the overall cost $g(s')$ is the accumulated travel cost

$$c(s') = c(s) + c_m(s, s') \; . \tag{16}$$

The second component is the overall collision risk so far along this path in the search space

$$r(s') = 1 - (1 - r(s))(1 - p_m(s', s)) \; . \tag{17}$$

Finally, the total cost $g(s')$ is calculated as a weighted sum using a high weighting factor λ_r.

$$g(s') = c(s') + \lambda_r \, r(s') \tag{18}$$

The weighting factor λ_r is used to tune the admissible detour for reducing the collision risk. The choice of λ_r is clearly application-specific and my be adapted online according to the currently encountered situation and environment.

IV. Results

We have implemented the individual components of the presented algorithm in C++ and evaluated them on an Intel® Xeon® E3-1270 CPU using previously recorded map data. For the sake of clarity, in this section, we will provide the results for our proposed algorithm separately for the individual features. However, of course, in reality, all these components operate simultaneously.

A. Collision Risk

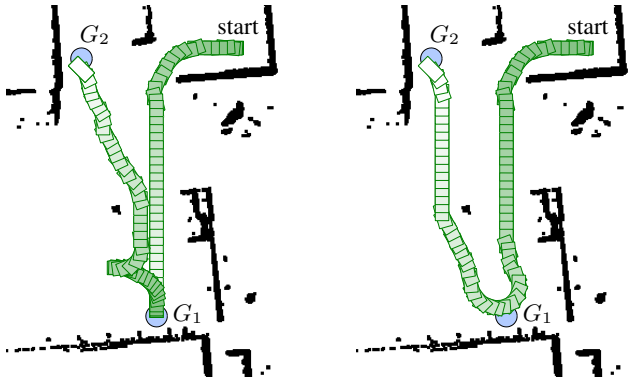
The first scenario (Fig. 3) demonstrates how static risks are taken into account. Each static obstacle (depicted in black) is surrounded by a safety zone according to II-A. This makes the planning trajectory pass through the middle of the small passage right after its start position. Furthermore, an additional risk zone (which could be a busy forklift route) is added to the map with a low but relevant collision probability for each instantaneous position. As one would expect, the planned path crosses the risk zone as fast as possible to reduce the overall collision risk, even though the chosen path is longer than the direct one cutting the risk zone diagonally.

In the second scenario (Fig. 4) the robot faces an oncoming dynamic obstacle. The position and uncertainty of the dynamic obstacle is predicted using the prediction step of a Kalman filter with a simple constant velocity model. The picture shows the propagated error ellipses with 90 % confidence. Due to the impending collision, our algorithm plans an avoidance maneuver to reduce the risk of collision.

Obviously it is important for the planner to explicitly take the time dimension into account to correctly incorporate the motion of dynamic obstacles. As planning a complete trajectory from the start to the goal G_1 in L'_0 (i.e., the (x, y, θ, v, t, G) space) would be computational expensive, the planning reduces to a planning in L'_1 (i.e., the (x, y, θ, G) space) after the time threshold t_0. Exactly this situation is depicted in Fig. 4. The dark-green tiles of the planning result correspond to the *trajectory* part, the light tiles to the mere *path* part. Dynamic obstacles are only considered during the trajectory part, i.e., until reaching the time t_0. This scheme of a search space with hybrid dimensionality allows for an efficient planning of even long paths while still guaranteeing safe navigation in the presence of dynamic obstacles.

B. Planning to Multiple Goal Regions

From the scenario in Fig. 5 it becomes evident that greedy planning considering only the next goal region G_1 leads to sub-optimal results. The robot cannot continue to drive forwards once having reached G_1; instead, it has to back off, turn around, and then drive to the next goal region G_2.



(a) separate planning (b) combined planning

Fig. 5. Difference between separate and combined planning for two goal regions. As expected, the combined planning is superior to the separate planning as it can avoid the unnecessary turning maneuver.

The lightness of the tiles in the figure increases with time during one planning iteration. Thus, one can easily see the two separate planning phases in Fig. 5a. For the sake of clarity, the transition from the trajectory part to the path part is not shown here.

In Fig. 5b the path, that results from a combined planning to the next two goal regions G_1 and G_2, is shown. The robot plans its path in an optimal forward-looking fashion only touching G_1 tangentially. Once again, the lightness of the tiles encodes the progressing time of the trajectory/path for the instantaneous planner iteration.

The computation time for planning the continuous trajectory/path from the start via G_1 to G_2 is shown in Fig. 6 depending on the time threshold t_0. The anytime nature of the ARA* algorithm makes it possible to quickly generate an initial solution and iteratively improve it if there is any computation time left. Even for the complex scenario in Fig. 5b and a quite low inflation factor $\epsilon = 1.3$, our algorithm is able to generate a solution in less than 40 ms. The computation time increases with the threshold t_0 and, of course, with each ARA* iteration. In practice, there are only marginal differences between the resulting trajectory/path when reducing ϵ to 1.1 or 1.0, although only the latter guarantees optimality. For this scenario the planner is able to generate trajectories/paths with 10 Hz for a setup of $\epsilon = 1.1$ and $t_0 = 4$ s.

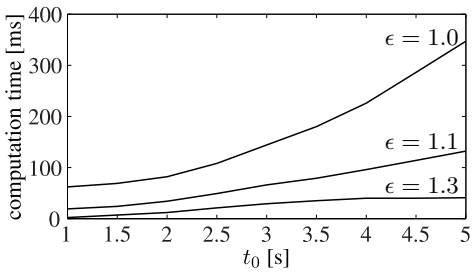


Fig. 6. Computation time for the trajectory/path planning of the scenario depicted in Fig. 5b.

V. Conclusion

In this paper, we have presented a novel algorithm that explicitly takes into account dynamic obstacles as well as subsequent goals in order to plan optimal collision-free global paths. For this purpose, we provided a way to consistently model the risk for collisions with static and dynamic obstacles. We constructed a state × time × goal lattice with hybrid dimensionality using efficiently simulated motion primitives. Finally, we applied the ARA* algorithm to our graph structure to efficiently generate hybrid trajectory/path solutions.

References

[1] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[2] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially contrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[3] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 387–400, 2001.

[4] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[5] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[6] A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.

[7] K. Gochev, B. Cohen, J. Butzke, A. Safonova, and M. Likhachev, "Path planning with adaptive dimensionality," in *Proceedings of the Symposium on Combinatorial Search*, 2011.

[8] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[9] R. A. Knepper, S. S. Srinivasa, and M. T. Mason, "Hierarchical planning architectures for mobile manipulation tasks in indoor environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010.

[10] C. R. Maurer, Jr., R. Qi, and V. Raghavan, "A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 265–270, 2003.

[11] T. Ohki, K. Nagatani, and K. Yoshida, "Collision avoidance method for mobile robot considering motion and personal spaces of evacuees," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[12] A. Bicchi, A. Marigo, and B. Piccoli, "On the reachability of quantized control systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 4, pp. 546–563, 2002.

[13] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *The International Journal of Robotics Research*, vol. 22, no. 7–8, pp. 583–601, 2003.

[14] J. Petereit, T. Emter, C. W. Frey, T. Kopfstedt, and A. Beutel, "Application of Hybrid A* to an autonomous mobile robot for path planning in unstructured outdoor environments," in *Proceedings of the 7th German Conference on Robotics (ROBOTIK 2012)*, 2012.

[15] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* search with provable bounds on sub-optimality," in *Proceedings of the Conference on Neural Information Processing Systems*, 2003.

[16] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics and Automation*, vol. 21, no. 3, pp. 354–363, 2005.

Year:
2013

Author(s):
Petereit, Janko; Emter, Thomas; Frey, Christian

Title:
Safe mobile robot motion planning for waypoint sequences in a dynamic environment

DOI:
10.1109/ICIT.2013.6505669 (http://dx.doi.org/10.1109/ICIT.2013.6505669)

Details:
Institute of Electrical and Electronics Engineers -IEEE-; IEEE Industrial Electronics Society -IES-:
IEEE International Conference on Industrial Technology, ICIT 2013. Vol.1 : 25-28 February 2013, Cape Town, Western Cape, South Africa
New York, NY: IEEE, 2013
ISBN: 978-1-4673-4567-5 (Print)
ISBN: 978-1-4673-4568-2 (Online)
ISBN: 978-1-4673-4569-9
pp.181-186