Online motion planning for autonomous vehicles in vast environments*

Tim Mercy[1], Erik Hostens[2] and Goele Pipeleers[1]

Abstract—**Nowadays, the potential of autonomous vehicles for order picking and material transport in vast environments with large amounts of obstacles is only exploited to a limited extent. In order to realize free, time-optimal motion of autonomous vehicles through such complex environments, this paper presents a novel motion planning approach. The approach combines a global path planner with a local trajectory generator. The global planner finds a path through the complete environment, taking only the stationary obstacles into account. The local trajectory generator computes a detailed trajectory in a local frame around the global path, accounting for both stationary and moving obstacles. This trajectory is parameterized as a spline, and is obtained by solving an optimal control problem. In order to always include the latest information about the environment, the optimal control problem is solved online with a receding horizon. The paper demonstrates the potential of the proposed method with extensive numerical simulations. In addition, it presents an experimental validation in which a *KUKA youBot* moves through an obstructed environment. To facilitate the numerical and experimental validation of the presented method, it is embodied in a user-friendly open-source software toolbox.**

I. Introduction

Vast, complex and uncertain environments are common in modern industry: docks, workshops and eCommerce warehouses contain large amounts of stationary obstacles like racks or walls, and moving obstacles like (human-operated) vehicles or humans walking around. While autonomous vehicles are becoming increasingly important for order picking and material transport in such environments, they are currently strongly restricted in their operation. To ensure safe operation, the vehicles are constrained to move over tracks that are separated from humans, and their velocity is limited. The selected vehicle tracks mainly consist of connections of straight lines, and generally, only one-way traffic is allowed in corridors. In addition, when there is a collision risk, the vehicle is stopped immediately. These restrictions severely restrain the potential of autonomous vehicles. Nowadays, the interest in a more flexible navigation approach is growing, in order to increase the performance of these systems. This poses challenges to the vehicle localization, safety, and in particular, to the motion planning.

*This work benefits from KU Leuven-BOF PFV/10/002 Centre of Excellence: Optimization in Engineering (OPTEC), from the project G0C4515N of the Research Foundation-Flanders (FWO-Flanders), from the KU Leuven Research project C14/15/067: B-spline based certificates of positivity with applications in engineering, and from the IWT ICON project 'Sitcontrol' of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) and Flanders Make.

[1]These authors are members of DMMS lab, Flanders Make - the strategic research centre for the manufacturing industry and of the MECO Research Team, Department of Mechanical Engineering, KU Leuven, BE-3001 Leuven, Belgium. [2]Flanders Make, Celestijnenlaan 300D Bus 4027, BE-3001 Leuven, Belgium. `tim.mercy@kuleuven.be`

Motion planning deals with the computation of collision-free trajectories that steer a vehicle from its current position to its destination, while taking into account the vehicle's motion model and its kinematic limits. This problem naturally translates into an optimal control problem (OCP). Solving this problem in one step returns the optimal motion trajectory and the corresponding controls to steer the vehicle over the trajectory. This is called a coupled motion planning approach, and is generally only feasible for simple environments. Therefore, a decoupled approach is often adopted, splitting the complicated problem in a path planning and a path following phase [1]. The path planning first computes a collision-free path as a collection of position setpoints (without any timing information), herein largely ignoring the system kinematics and dynamics. Afterwards, a path following method computes the controls to steer the vehicle along the computed path, taking the system limitations into account. Various techniques have been developed to solve the path planning, including graph based methods such as A* [2] and D*-lite [3], random sampling based methods like Rapidly exploring Random Trees (RRT) [4] and Probabilistic Road Maps (PRM) [5], methods based on artificial potential fields [6] and [7], and optimization based methods such as Trajopt [8] and ITOMP [9]. The path following phase often uses Model Predictive Control (MPC) [10] to steer the system along the computed path, as MPC can explicitly account for the system limitations.

While the decoupled approach is an adequate way of addressing the complexity of the motion planning problem in vast environments, it involves two major drawbacks. The first drawback is that it leads to suboptimal results, since not all information is combined into a single problem. The second drawback is that the majority of path planning methods can either only handle stationary obstacles, or require the trajectories of the obstacles in the environment to be fully known a priori. In the latter case, a complete replanning is needed when an obstacle deviates from its predicted trajectory, an obstacle leaves the environment, or a new obstacle shows up. Therefore, current decoupled approaches are suboptimal and inflexible.

In order to mitigate these drawbacks, this paper presents a novel optimization based approach to efficiently compute motion trajectories through vast, complex and uncertain environments. It combines a global path planner with a local trajectory optimizer: the global planner uses a path planning method (e.g. the A*-algorithm) to find a rough path through the environment, taking only the stationary obstacles into account. The local planner computes a trajectory through local frames around the global path, taking into account both stationary and moving obstacles, as well as the vehicle's motion model and its kinematic limits. To this end it solves an OCP, in which only the locally relevant obstacles are included. Uncertainties in the environment require solving this problem online with a receding horizon. To efficiently solve the OCP at every time sample, the spline-based approach of [11] and [12] is adopted. The resulting trajectory generation method outperforms current decoupled approaches in both optimality, since it allows (substantial) deviation from the global path, and flexibility, since it can account for uncertain obstacle motions.

The potential of the presented method is demonstrated by extensive numerical simulations, in combination with an experimental validation on a *KUKA youBot* moving through an obstructed environment. To facilitate modeling, simulating and exporting motion planning problems to practical setups, the method is embodied in OMG-tools, a user-friendly open-source Python toolbox [13].

Section II describes the general OCP and its online implementation. Section III shows how the scheduler combines a global planner and a local trajectory generator to reduce the complexity of the optimization problems. In addition, it explains how to keep these optimization problems small-scale, by using a spline parameterization. Afterwards, Section IV validates the presented method by showing numerical simulation examples together with an experimental validation. Finally, Section V concludes the paper.

II. Problem formulation

The presented motion planning approach aims at solving a global optimal control problem to obtain the collision-free trajectory that moves the vehicle as fast or as energy efficient as possible through an environment, while taking into account the vehicle's motion model and its kinematic limits. This OCP is described more in detail below. For the sake of clarity, the paper focuses on a holonomic vehicle with a fixed orientation. However, the presented approach, can also handle vehicles with a varying orientation, as well as alternative kinematic models such as a differential drive or a bicycle (see example in Section IV-A).

Considering a holonomic vehicle with a fixed orientation and perfect velocity control on its wheels, the motion trajectory is given by:

$$q(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix},$$

in which $x(t)$ and $y(t)$ are the position as a function of time. The inputs $u(t)$, being the setpoints for the wheel velocity controllers are given by the corresponding velocities $\dot{q}(t)$.

The computed trajectory must steer the system from its current position q_0 to its destination q_T. In addition, it is often desired to impose specific initial and final inputs. Hence, the following boundary conditions are included in the OCP:

$$\begin{aligned} q(0) &= q_0 \quad q(T) = q_T, \\ u(0) &= u_0 \quad u(T) = u_T. \end{aligned}$$

In this equation T represents the total motion time that is required to reach the goal state. In addition, the holonomic vehicle's velocity and acceleration bounds must be respected at all times:

$$\begin{aligned} \dot{q}_{\min} &\leq \dot{q}(t) \leq \dot{q}_{\max}, \quad \forall t \in [0, T], \\ \ddot{q}_{\min} &\leq \ddot{q}(t) \leq \ddot{q}_{\max}, \quad \forall t \in [0, T]. \end{aligned}$$

Since the environment contains (moving) obstacles, collision avoidance constraints are essential to obtain a collision-free motion trajectory. To account for obstacle movement, the constraints include a linear motion model for each obstacle. This model uses the current estimate of the obstacle position p_i and velocity v_i to predict the obstacle position p_i^{pred} as a function of time:

$$p_i^{\text{pred}}(t) = p_i + t \cdot v_i.$$

The collision avoidance constraints themselves express that the shape of the vehicle must not overlap with the shape of any of the N obstacles:

$$\text{dist}(\text{veh}(q(t)), \text{obs}_i\ (p_i^{\text{pred}}(t))) \geq \epsilon, \quad \forall t \in [0, T],$$

where $i \in [1, N]$ and ϵ is a safety factor to ensure that the vehicle keeps a safe distance from obstacles, if possible. In general, these constraints are nonlinear and non-convex, which highly complicates the OCP. A very similar constraint imposes that the vehicle needs to stay within the boundaries of the environment:

$$\text{dist}(\text{veh}(q(t)), \text{border}) \geq \epsilon, \quad \forall t \in [0, T].$$

Finally, the goal is to obtain time-optimal trajectories. To be able to optimize the motion time T, the problem is reformulated using a dimensionless time $\tau = \frac{t}{T}$. Combining all elements leads to the following OCP:

$$\begin{aligned} \underset{q(\cdot), T}{\text{minimize}} \quad & T \\ \text{subject to} \quad & q(0) = q_0, \quad q(1) = q_T, \\ & u(0) = u_0, \quad u(1) = u_T, \\ & u_{\min} \cdot T \leq u(\tau) \leq u_{\max} \cdot T, \\ & \ddot{q}_{\min} \cdot T^2 \leq \ddot{q}(\tau) \leq \ddot{q}_{\max} \cdot T^2, \\ & \text{dist}(\text{veh}(q(\tau)), \text{obs}_i\ (p_i^{\text{pred}}(\tau))) \geq \epsilon, \\ & \text{dist}(\text{veh}(q(\tau)), \text{border}) \geq \epsilon \\ & i \in [1, N], \quad \forall \tau \in [0,\ 1]. \end{aligned} \tag{1}$$

Since the environment is uncertain, new obstacles may appear and present obstacles may change their movement direction or velocity. As a consequence, the prediction of the obstacle behavior will differ from reality. To handle these uncertainties, the proposed method solves (1) with a receding horizon, like in MPC. At every time sample the vehicle and environment states are measured, allowing insertion of the most recent information in the problem. An important consequence is that the complicated OCP must be solved online during the vehicle movement, which requires a low solution time. The following section describes how this can be achieved by combining a global planner with a local trajectory generator, in addition to using a B-spline parameterization for the trajectories.

III. Methodology

In vast environments with numerous obstacles, problem (1) becomes very hard to solve. This is due to the long time horizon that must be considered in order to reach the destination, in combination with the non-convex collision avoidance constraints, leading to many local minima. Furthermore, the behavior of moving obstacles near the end of the trajectory would influence the complete motion trajectory. Due to uncertainties this is not beneficial, since the situation may have changed completely when the vehicle approaches its goal position. To deal with these problems, the presented approach uses a scheduler to combine a local trajectory generator with any global path planner that returns a path as a sequence of accessible positions (waypoints).

Section III-A elaborates how the scheduler handles the complex OCP, by combining the global planner and local trajectory generator to split the problem over different frames. Afterwards, Section III-B focuses on the computation of the frames themselves. Finally, Section III-C explains how to obtain a small-scale local trajectory generation problem inside a frame, using a spline parameterization.

A. Scheduler

Algorithm 1 gives an overview of the procedure that the scheduler follows to compute a motion trajectory through a complex environment.

First, the user selects an overall goal and a frame type (Section III-B). Afterwards, the scheduler computes a so-called local frame, which comes down to selecting a smaller environment inside the total environment. For this computation, the scheduler first asks the global planner for a path, consisting of a sequence of waypoints, that connects the vehicle position at the time of frame creation, with its desired goal position. The local frame is built around this global path, however, the exact creation method depends on the selected frame type (Section III-B). Inside the frame, the vehicle has a subgoal, being the last reachable waypoint of the global path inside the frame (taking into account the vehicle dimensions). Using the part of the global path inside the frame, the scheduler guesses the motion time that is required to reach the subgoal and determines which moving obstacles will pass through the frame during that time period, allowing to add only the relevant moving obstacles. Finally, the frame is converted into an OCP of the form (1) and is solved by the local trajectory generator. The solution consists of the vehicle trajectories $q(t)$, inputs $u(t)$ and motion time T_{frame} required to move through the frame. The vehicle follows these trajectories over a certain time period Δt, after which the information about the environment is updated. When a new obstacle crosses the frame or a previously added obstacle left the frame, the obstacles of the OCP are updated, keeping the same frame borders.

Algorithm 1 Scheduler combining global path planner and local trajectory generator

```
1:  **Input:**
2:  set frame type: square or max_waypoints
3:  set overall goal position
4:  **Repeat** every $\Delta t$
5:      **if** frame is not valid or no frame
6:          generate global path
7:          create new frame + local goal position
8:      **else**
9:          update information from environment
10:         solve local OCP: get new $q(t)$, $u(t)$ and $T_{\text{frame}}$
11:         follow $q(t)$ for $\Delta t$, using $u(t)$
12: **Until** goal position reached
```

At each time step, the scheduler checks the progress of the vehicle motion through the frame. As long as the vehicle has not made enough progress, the frame remains valid (Section III-B). When a new frame is required, the scheduler first asks the global planner for a new path, starting at the vehicle's current position. This is necessary because moving obstacles may have caused the vehicle to strongly deviate from the originally computed path. This procedure is repeated until, finally, the vehicle reaches its overall goal position.

B. Frame types

The way the scheduler computes a new frame and decides upon its validity depends on the frame type. Currently, two frame types are considered: (i) a square frame with a fixed size; and (ii) a frame including as many waypoints as possible, without containing any stationary obstacle, as shown in Figure 1.

When computing a square frame of fixed size, the square is first centered around the current vehicle position, and is afterwards shifted in the direction of the global path. This frame stays valid until the vehicle traveled over a specified percentage of the initially computed local trajectory. Afterwards, a new square frame is computed, using the same procedure, until the vehicle can reach the goal position inside the current frame.

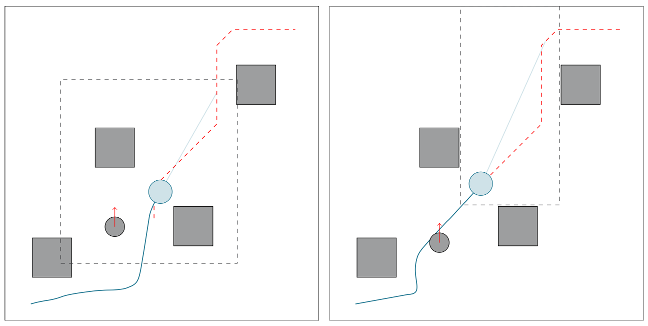
(a) Fixed size square (b) Maximum number of waypoints, no stationary obstacles

Fig. 1: Illustration of frame types

When computing a frame of the second type, the scheduler first includes as many waypoints of the global path as possible in a rectangle of minimal size, which contains no stationary obstacles. Afterwards, it scales up this rectangle in all directions, until it hits the borders of either an obstacle, or the overall environment. In addition, the scheduler repeats this procedure to calculate a second frame, using the goal position inside the first frame as a starting position. Therefore, both frames will always overlap in a certain region. The first frame stays valid until the vehicle enters the region of overlap. At that point, the scheduler switches to the second frame and computes the subsequent frame.

Both frame types have their advantages and drawbacks. When selecting a square frame of a fixed size (Figure 1a), in some cases small parts of obstacles are included in the frame, complicating the optimization problem, while being irrelevant for the obtained motion trajectory. On the other hand, this approach works well in unstructured environments, and obstacles that are moving around the corner are easily taken into account. When selecting the second frame type (Figure 1b), easy local motion planning problems are obtained. This results in lower average and maximum solving times in the receding horizon implementation. Another advantage is that this type of frame is well-suited in structured environments, since it selects large corridors. The downside is that the average time to compute a new frame is slightly higher compared to the one for a frame with fixed size.

For both frame types, the local trajectory generation problem is solved with a receding horizon. As a consequence, the resulting OCP needs to be solved online, during the vehicle movement. This requires a low solving time. The following section describes how this can be achieved by means of B-splines.

C. Local trajectory generation using B-splines

In [11], the authors proposed a method to transform OCPs of the form (1) into a small-scale nonlinear optimization problem suitable for online optimization. There are two key aspects of the method: (i) a B-spline parameterization is adopted for the motion trajectory $q(\cdot)$; and (ii) the properties of B-splines are exploited to replace constraints over the entire time horizon by small finite, yet conservative, sets of constraints.

Splines are piecewise polynomial functions, which can describe complex trajectories, using few variables. A spline is given by a linear combination of B-spline basis functions [14]. Therefore, the motion trajectory is parameterized as:

$$q(\tau) = \sum_{i=1}^{n} c_i^q \cdot B_i^q(\tau).$$

Here $B_i^q(\tau)$ are B-spline basis functions, c_i^q are the spline coefficients and n follows from the spline degree and the number of knots. When using this spline parameterization in (1), the variables of the problem become the coefficients c_i^q and motion time T. Since the motion trajectory $q(\tau)$ is a spline, the velocities $\dot{q}(\tau)$ and accelerations $\ddot{q}(\tau)$ are splines as well. Their coefficients $c_i^{\dot{q}}$ and $c_i^{\ddot{q}}$ are readily verified to depend linearly on c_i^q, e.g.:

$$\dot{q}(\tau) = \sum_{i=1}^{n-1} c_i^{\dot{q}}(c_i^q) \cdot B_i^{\dot{q}}(\tau).$$

The main reason for adopting the B-spline parameterization is the corresponding convex hull property, which states that a spline is always contained in the convex hull of its B-spline coefficients [14]. This way, spline constraints can be relaxed to constraints on the B-spline coefficients. For instance, the semi-infinite velocity constraints

$$\dot{q}_{\min} \leq \dot{q}(\tau) \leq \dot{q}_{\max}, \ \forall \ \tau \in [0, 1] \tag{2}$$

are guaranteed to hold if

$$\dot{q}_{\min} \leq c_i^{\dot{q}} \leq \dot{q}_{\max}, \ i = 1 \ldots n. \tag{3}$$

Replacing semi-infinite sets of constraints of the form (2), by the finite, yet conservative sets (3) is called a B-spline relaxation. The major advantage is that B-spline relaxations avoid time gridding of the constraints, while they guarantee constraint satisfaction at all times. The disadvantage is that B-spline relaxations also introduce some conservatism. This conservatism can be reduced by choosing a higher dimensional basis, at the cost of introducing extra constraints [11]. The collision avoidance constraints in (1) are implemented using the separating hyperplane theorem [15] and use a spline parameterization of the normal vector and offset of the hyperplane. Since these constraints also need to hold over the complete motion time, they are relaxed as well. For a more detailed explanation, see [12].

Using a spline parameterization for the OCP, in combination with B-spline relaxations of semi-infinite constraints, leads to a tractable nonlinear optimization problem, which can be solved fast. To further reduce the solution time and allow a swift reaction to changes in the environment, an initial guess is provided. By the initialization of a new frame, the part of the global path that lies inside the frame is used to guess a trajectory. While the vehicle is moving through the frame, the solution from the previous time step is used as an initial guess to solve the current OCP.

IV. Results

This section validates the presented motion planning method using numerical simulations (Section IV-A), together with an experimental validation (Section IV-B). In all validations, the spline-based motion planning problem is formulated using OMG-tools: a user-friendly open-source toolbox. The Github page of this toolbox contains animations of all simulations, and a video of the experimental validation [13]. OMG-tools is written in Python and uses CasADi [16] as a symbolic framework to formulate the motion planning problem, transform it into a tractable optimization problem and pass it to the solver. The default solver for all problems in this paper is Ipopt [17]. All simulations were done on a notebook with Intel Core i5-4300M CPU @2.60GHz x 4 processor and 8GB of memory.

A. Simulations

Figure 1 shows two types of frames for an example in which a holonomic vehicle (maximum velocity $1.2\,\frac{\text{m}}{\text{s}}$) is crossing an environment containing four stationary and one moving circular obstacle (moving at $1\,\frac{\text{m}}{\text{s}}$). For a square frame of fixed size (Figure 1a), the average solving time (t_{avg}) for the example is 52 ms, the maximum solving time (t_{max}) is 324 ms. When selecting a frame which includes as many waypoints as possible without containing any stationary obstacles (Figure 1b), easy local motion planning problems are obtained, resulting in a lower t_{avg} (34 ms) and t_{max} (264 ms). The downside is that the average time to compute a new frame (7 ms) is slightly higher, compared to when using a frame of fixed size (3 ms). In addition, there are only three frames of fixed size, while four frames of variable size are required. Note that for both frame types there is a bend in the beginning of the trajectory. This apparent detour is necessary to avoid the moving obstacle.

Figure 2 shows the simulated motion of a holonomic vehicle (maximum velocity $1.2\,\frac{\text{m}}{\text{s}}$) moving through a vast, structured environment with two circular obstacles which move vertically and one which moves horizontally (indicated by the arrows), at a velocity of $1.5\,\frac{\text{m}}{\text{s}}$. All moving obstacles invert their movement direction when hitting a wall. The figure indicates all considered local frames without stationary obstacles, indicated by the dashed rectangles and plotted in different colors. The dotted red line shows the rough global path that is originally computed by the global planner. The solid line is a connection of the spline trajectories over all frames, computed by the local planner. For each frame the initial vehicle position is plotted (circle), together with its trajectory through the frame and its goal position within the frame (cross), all in the corresponding frame color. Figure 2 clearly shows that the local spline trajectories traverse the warehouse in a more optimal way than the grid path, since they are more flexible. Furthermore, it is clear that the goal position differs from the end position of the grid path. This is because the grid path can only connect grid points, while the spline trajectory can reach any feasible position. Note that the region near the goal position seems to contain a detour, this is due to the obstacle which moves horizontally, and bounces back off the wall at the moment when the vehicle comes close. The local planner takes the new movement direction of the obstacle into account and finds an updated trajectory around it, keeping the motion collision-free. While less clearly visible, a similar explanation holds for the detour near center of Figure 2. For the complete example, t_{avg} is 34 ms, t_{max} is 260 ms and the average time to compute one of the six local frames is 24 ms.

To demonstrate the potential of the presented method for nonholonomic vehicles, Figure 3 shows how a differential drive moves through the environment of Figure 1. Again, a detour is required in the beginning of the trajectory to avoid collision with the moving circular obstacle. For this example, t_{avg} is 120 ms and t_{max} is 419 ms. The approach to solve this problem is very similar to one explained in Section III, only

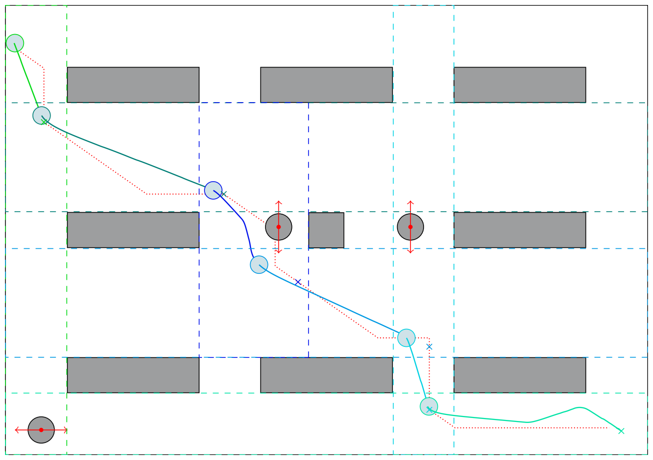
Fig. 2: Holonomic vehicle moving through a vast environment

the local trajectory generator using B-splines requires changes. For a detailed description on how to apply Section III-C to this type of vehicle, the reader is referred to [18].

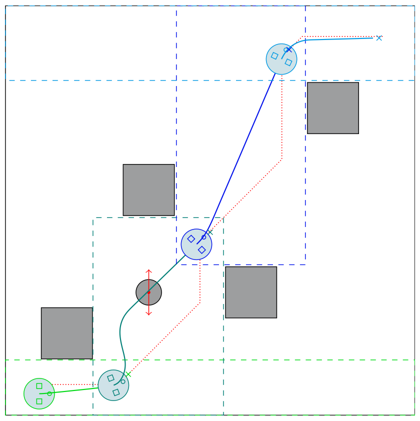
Fig. 3: Differential drive moving through an obstructed environment

B. Experimental validation

For the experimental validation, the proposed combination of a global and local planner is applied on a test set-up in which a *KUKA youBot* moves through a vast environment. The position of this holonomic vehicle is calculated using an on-board Ultra Wideband (UWB) beacon and four UWB beacons on known locations at the border of the environment. The stationary obstacles are manually placed in the environment, at a known location. A second holonomic vehicle serves as a moving obstacle and is manually steered using a joystick. The position of this vehicle is determined by an on-board UWB beacon as well. In the optimization problem, both the vehicle and the moving obstacle are represented as circles, since they are nearly square, and their orientation is not taken into account. During the experiment both the *youBot* and obstacle velocities were limited to $0.4\,\frac{\text{m}}{\text{s}}$. All computations are performed on an external desktop computer with Intel Core Xeon E5-1603 v3 CPU @ 2.8GHz x 4 processor and 16GB of memory.

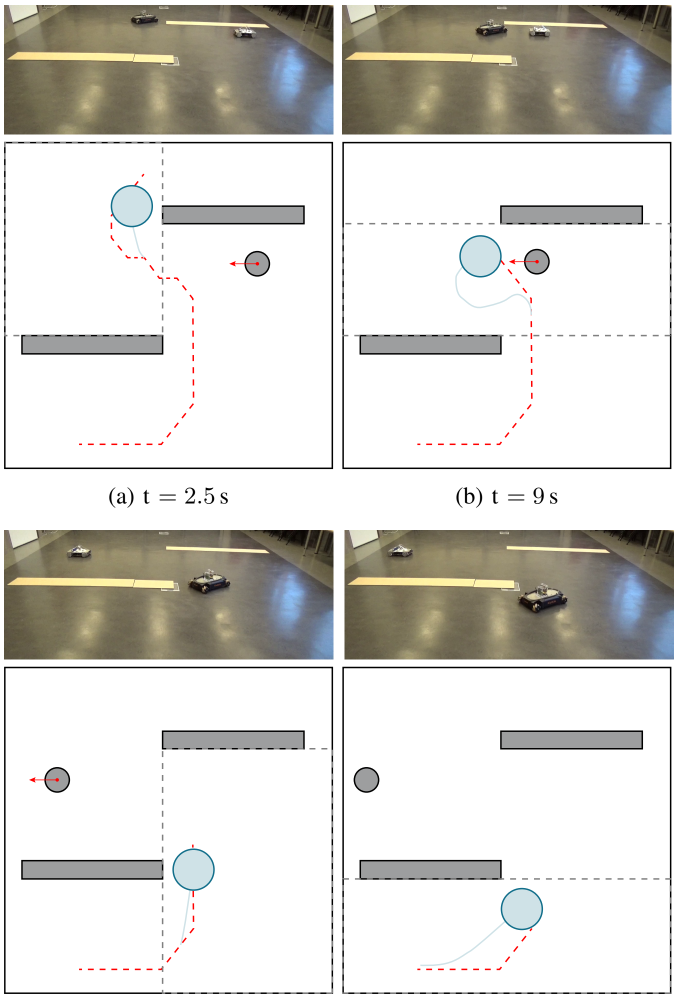
(a) t = 2.5 s (b) t = 9 s (c) t = 16 s (d) t = 17.5 s

Fig. 4: *KUKA youBot* moving through an environment with a moving obstacle

The local trajectory planner returns both position and velocity trajectories for the *youBot*. The vehicle has an on-board PI position controller running at 25 Hz, using the position trajectories in world coordinates as a reference, and the velocity trajectories as feedforward signals. While the vehicle orientation is not included in the optimization, it is still controlled to the desired value, using a separate PI controller.

Figure 4 shows several snapshots of a *youBot* moving through the environment. The top parts of the figure show the scene, while the bottom parts show the corresponding environment, including the trajectories that are computed in OMG-tools. Figure 4b shows how the receding horizon implementation of the local trajectory generation allows taking the movement of the circular obstacle into account, such that the vehicle can avoid it. The solving times for this example are comparable to the ones for the examples from Figure 1b and Figure 2.

V. Conclusion

To realize free motion of autonomous vehicles through vast and uncertain environments, this paper presents a new approach to compute time-optimal motion trajectories. By combining a global path planner with a local trajectory generator, the complex total environment is divided over several smaller environments, reducing the complexity of the resulting optimal control problems. These problems are formulated efficiently by using a spline parameterization of the trajectory, allowing online solution of the problem in a receding horizon fashion. As a consequence, the method can swiftly react to changes in the environment. Numerical simulations, together with an experimental validation, show that this approach allows the efficient computation of collision-free trajectories at every time instant. The presented approach is implemented in OMG-tools, a user-friendly open-source Python toolbox.

References

[1] F. Debrouwere, “Optimal Robot Path Following: Fast Solution Methods for Practical Non-convex Applications,” Ph.D. dissertation, KU Leuven, 2015.

[2] P. Hart, N. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[3] S. Koenig and M. Likhachev, “D* Lite,” *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pp. 476–483, 2002.

[4] S. M. Lavalle and J. J. Kuffner, “Randomized Kinodynamic Planning,” *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

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

[6] S. Ge and Y. Cui, “Dynamic motion planning for mobile robots using potential field method,” *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002.

[7] L.-C. Lai, C.-J. Wu, and Y.-L. Shiue, “A potential field method for robot motion planning in unknown environments,” *Journal of the Chinese Institute of Engineers*, vol. 30, no. 3, pp. 369–377, 2007.

[8] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, “Finding locally optimal, collision-free trajectories with sequential convex optimization,” *Robotics: Science and Systems*, vol. 9, no. 1, pp. 1–10, 2013.

[9] C. Park, J. Pan, and D. Manocha, “ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments,” *Proceedings of the International Conference on Automated Planning and Scheduling*, pp. 207–215, 2012.

[10] E. Camacho and C. Bordons Alba, *Model Predictive Control*. London: Springer-Verlag, 2007.

[11] W. Van Loock, G. Pipeleers, and J. Swevers, “B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction,” *Mechanical Sciences*, vol. 6, no. 2, pp. 163–171, 2015.

[12] T. Mercy, W. Van Loock, and G. Pipeleers, “Real-time motion planning in the presence of moving obstacles,” in *European Control Conference (ECC)*, 2016, pp. 1586–1591.

[13] R. Van Parys and T. Mercy, “OMG-tools,” https://github.com/meco-group/omg-tools, 2016.

[14] C. de Boor, *A Practical Guide to Splines*. New York: Springer-Verlag, 1978.

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[16] J. Andersson, “A General-Purpose Software Framework for Dynamic Optimization,” Ph.D. dissertation, KU Leuven, 2013.

[17] A. Wächter and L. T. Biegler, “On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming,” *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[18] T. Mercy, R. Van Parys, and G. Pipeleers, “Spline-Based Motion Planning for Autonomous Guided Vehicles in a Dynamic Environment,” *IEEE Transactions On Control Systems Technology*, 2017.