Online Generation of Kinodynamic Trajectories for Non-Circular Omnidirectional Robots

Christoph Sprunk* Boris Lau* Patrick Pfaff‡ Wolfram Burgard*

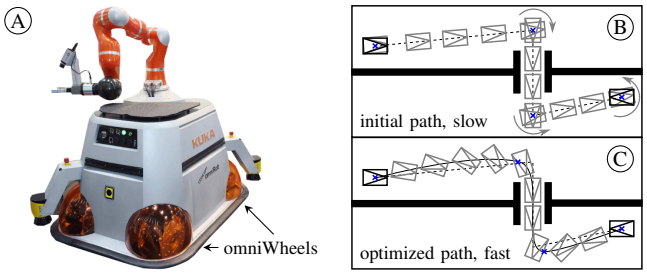
***Abstract*— This paper presents a novel approach to kinodynamic trajectory generation for non-circular omnidirectional platforms that can be combined with existing path planners. We use quintic Bézier splines to specify position and orientation of the holonomic robot for every point in time. To fully exploit the capabilities of the holonomic robot we propose a novel path representation. It allows for continuous variation of path shapes in the spectrum between straight-line paths with turns on the spot and smooth paths with independent rotations and translations. Using this representation our method optimizes trajectories according to a user-defined cost function, considering the constraints of the platform. This way, it generates fast and efficient trajectories in an anytime fashion. The experiments carried out on an industrial robot show that our approach generates highly efficient and smooth motion trajectories that can be tracked with high precision and predictability. Furthermore, the system operates in real-world environments containing unmapped obstacles and narrow passages.**

I. Introduction

Over the past years, holonomic vehicles with non-steered omnidirectional wheels developed into commercially available products. With their ability to freely translate and rotate in all directions independently and simultaneously they can perform complex maneuvers in narrow spaces as shown in Fig. 1 and Fig. 2. Due to their slip-free motion, precise movements are possible even with heavy payloads. Therefore, this type of platform is attractive for applications like mobile manipulators, fork lifts, or transportation vehicles.

To achieve autonomous motion of such platforms, a planning system is required to generate trajectories that are free of collisions, feasible regarding constraints, and effective or ideally optimal with respect to a given cost function. Early approaches for omnidirectional platforms assume circular robots and completely neglect the orientation [1], [2], which reduces the size of the problem space compared to non-holonomic planning. Although non-holonomic constraints add a level of complexity to motion planning, they also reduce the size of the search spaces. Thus, the problem space for a non-circular holonomic platform is substantially larger due to the added dimension (see the table in Fig. 1), and due to the absence of a confining non-holonomic constraint.

We propose a new optimization-based approach to generate smooth and efficient trajectories for non-circular omnidirectional robots. In contrast to previous work it operates online and in the full configuration space of the holonomic platform. It furthermore deals with both, mapped and unexpected obstacles. To achieve this, it does not perform global search in the spatio-temporal state space, but instead builds on waypoints given by a global path planner. From these waypoints it generates an initial path consisting of pure translations alternating with turns on the spot. Considering admissible velocities, the trajectory is iteratively optimized to minimize a cost function that might assess the travel time, energy consumption, or other user preferences.



Platform	State space	Control space
1) Orientation-free holonomic	4D: $x, y, \dot{x}, \dot{y}$	2D: $\ddot{x}, \ddot{y}$
2) Differential drive / Ackermann	5D: $x, y, \theta, v, \omega/\phi$	2D: $\dot{v}, \dot{\omega}/\dot{\phi}$
3) Non-circular holonomic (ours)	**6D:** $x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}$	**3D:** $\ddot{x}, \ddot{y}, \ddot{\theta}$

Fig. 1. Holonomic platform with four "omniWheels" (A), and trajectories before (B) and after optimization (C) with our method. The table compares state and control spaces of different platform types, containing pose (x, y, θ), translational/rotational velocities (v, ω), steering angle (ϕ), or derivatives.

In this paper we describe the individual properties of our approach, the implementation on an omnidirectional platform, and the evaluation in real-world experiments that show its versatile behavior. We furthermore present simulations to evaluate the performance of our method in combination with several waypoint planning algorithms.

II. Related Work

Previous work on holonomic robots with omnidirectional wheels mostly covers fundamental control topics like position and velocity control [3], trajectory tracking [4], and optimal trajectories in the absence of obstacles [5], [6]. A number of search-based approaches to kinodynamic motion planning have been developed for differential and Ackermann drives. Often AD* or RRTs are used with predefined action sets consisting of feasible trajectory pieces [7], [8]. To gain online feasibility, one usually has to make compromises like coarse discretization of the action space. For omnidirectional platforms we have to consider an even larger state space and also a considerably larger action set. LaValle and Kuffner use RRTs to find paths in such high-dimensional but static spaces in an offline manner [9].

*Department of Computer Science, University of Freiburg, Germany, {sprunkc,lau,burgard}@informatik.uni-freiburg.de

‡KUKA Laboratories GmbH, Augsburg, Germany, patrick.pfaff@kuka.com

This work has partly been supported by the European Commission under contract numbers FP7-248258-First-MM and FP7-248873-RADHAR.

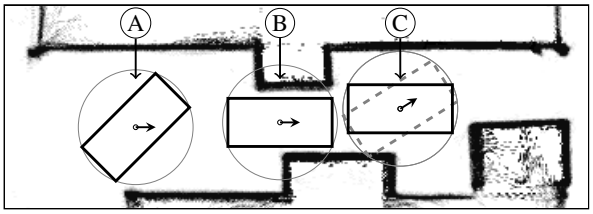


Fig. 2. Omnidirectional platforms are versatile in narrow spaces. They can rotate freely in open spaces (A), move with constant orientation in narrow passages (B), and execute maneuvers (C) where non-holonomic trajectories would lead to collisions (dashed). For readability, the figure contains the circumcircle of the rectangular robot.

To reduce the complexity of the motion planning problem, one can decouple the generation of the trajectory shape from determining velocities along the path [10]. Many approaches plan paths through traversable space without considering velocities at all. To execute these paths, motor velocities are typically determined by controllers [11] or reactive systems [12] which smooth the trajectories and thereby deliberately deviate from the planned path during execution.

Alternatively, several approaches exist that employ parametric path representations and inherently plan smooth paths. By modifying control points, these paths can be deformed to account for obstacles [13], [14], [15]. Unlike our approach, all of these systems do not plan velocities in advance. This way, accurate motion prediction is rendered impossible, hard kinodynamic constraints become difficult to enforce, and a dynamically feasible solution cannot be guaranteed.

Kinodynamic trajectories in contrast specify the position, orientation, and corresponding velocities of the robot for every point in time. If platform constraints are satisfied during generation, these trajectories can be tracked with high precision and predictability using an error-feedback controller. Fraichard and Delsart deform such trajectories to restore collision-freeness after they have been invalidated by dynamic obstacles [16]. In our previous work we proposed a system for non-holonomic motion planning that starts from an admissible initial path [17]. It uses quintic Bézier splines to model trajectories that can be appended or locally modified without any discontinuities. This way, it accounts for unexpected obstacles, unlike a similar cubic B-spline approach that only operates in static environments [18].

This paper addresses spline-based trajectories for holonomic vehicles that can rotate and translate independently and simultaneously. Exploiting these capabilities is usually required to achieve efficient and smooth trajectories as in Fig. 1 (C). However, pure translation on straight paths is needed to traverse narrow corridors and to generate collision-free initial paths between waypoints, as shown in Fig. 1 (B). A key contribution of this paper is a novel representation that enables path optimization in the full spectrum between these two extremes. Thus, we can exploit the full potential of omnidirectional platforms. Furthermore, we formulate kinodynamic constraints for holonomic platforms and propose methods for efficient collision checking. In contrast to previous work, we plan kinodynamic trajectories for holonomic platforms that can be executed with high precision and account for constraints and unexpected obstacles.

III. Holonomic Path Representation

As mentioned before, our system employs a global planner that provides a traversable path to the goal. This path is represented by $N+1$ waypoints W_i^0, $i \in \{0, \dots, N\}$, connected by N straight line segments. The planner assures the traversability of the path when moving with constant orientation θ_i^0 between W_i^0 and W_{i+1}^0 for all segments, and turning on the spot at the waypoints.

Given this path we create a continuous trajectory $Q(u(t))$ that specifies the pose (position and orientation) of the robot for any point in time t, and by its derivative the corresponding velocities as well. It consists of the path $Q(u)$ which determines the progression of robot poses $\langle x, y, \theta \rangle$ in global coordinates as a function of a native parameter $u \in [0, N]$, i.e., without regarding velocities. The path is associated with a mapping $u(t)$ that represents an admissible velocity profile for the path, as described in Sect. IV.

This section recalls our previous work on 2D path representations where the direction of travel determined the robot's orientation. Furthermore, it proposes a novel representation for independent rotation that specifically addresses the challenges of omnidirectional platforms. Together, this provides a set of meaningful parameters that allow for continuous and efficient optimization of holonomic paths.

A. 2D path shape over ground

The orientation-free 2D path of the robot is modeled using quintic Bézier splines segments that connect pairs of consecutive waypoints W_i and W_{i+1}. This allows for smooth, curvature continuous paths with minimal interdependence of spline segments that are locally adaptable.

To closely approximate the provided straight-line path with the spline segments, we set $W_i := W_i^0$ and use small derivatives at these points to achieve turns on the spot. The positional components of the waypoints and the magnitudes of the corresponding first derivatives are part of the high-level parameters altered by the optimization, e.g., to widen curves and adapt the distance to obstacles, see [17].

B. From turns on the spot to holonomic motion

At first glance, 2D spline paths could trivially be extended with an additional dimension to independently represent orientation. This way, however, the robot would rotate inbetween instead of at the waypoints, which appears unnatural to humans and furthermore prohibits translation through narrow passages with constant orientation as in Fig. 1 (right).

Our representation tackles this problem by introducing points r_i^{s} and r_i^{e} on the segments enclosing each waypoint W_i. At these points, the spline segments are subdivided allowing constant orientations θ_{i-1} and θ_i on the "middle" part of the adjacent segments (between r_{i-1}^{e}, r_i^{s} and r_i^{e}, r_{i+1}^{s}, respectively) and smooth rotation from θ_{i-1} to θ_i around the waypoint W_i. Thus, rotation starts at r_i^{s} and ends at r_i^{e}.

To approximate the straight line path as initial path for the optimization, r_i^{s} and r_i^{e} are placed directly on W_i which yields a turn on the spot as in Fig. 3 (left). The optimization can slide r_i^{s} and r_i^{e} away from W_i (green arrows) which

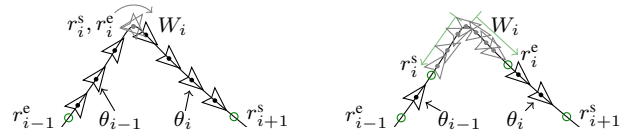


Fig. 3. Settings of rotation control points r_i^{s} and r_i^{e} (green) at waypoint W_i for a turn on the spot (left) vs. rotation during translation (right). The darts indicate robot motion with constant (black) and changing (gray) rotation.

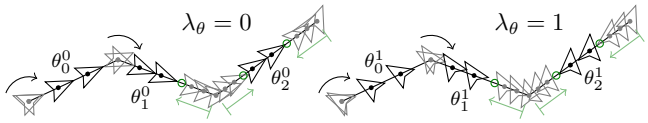


Fig. 4. Paths with different rotation behaviors. *Left:* initial paths use $\lambda_\theta = 0$ to orient the robot as specified by the waypoint planner. *Right:* minimized change of rotation with $\lambda_\theta = 1$. The optimization can adjust λ_θ to interpolate between these two extremes.

causes smooth rotation during translation (right) and thus reduces travel time. In the extreme case, r_i^{s} and r_i^{e} of waypoint W_i coincide with r_{i-1}^{e} and r_{i+1}^{s} of the neighboring waypoints W_{i-1} and W_{i+1}, which causes uninterrupted rotation.

C. From conservative orientation to minimized rotation

The control points r_i^{s} and r_i^{e} presented above determine *where* on the path the robot rotates. However, the initial values for θ_i given by the waypoint planner (θ_i^0) can be conservative. Adjusting the θ_i to reduce the total amount of rotation as shown in Fig. 4 can further reduce travel time or energy consumption (cf. Sect. VI-C).

Usually, the orientation of the robot at start and goal is given. At intermediate waypoints it can also be fixed due to adjacent path segments requiring a particular orientation. We therefore split the path at these waypoints and optimize the rotational behavior of the resulting subpaths independently.

For such a subpath with waypoints W_i, $i \in \{0, \dots, M\}$ we compute the orientations θ_i^1 that realize a minimal change of orientation. They interpolate the given start and end orientation θ_{start} and θ_{end} of the subpath as shown in Fig. 4 (right), and are determined according to

$$\theta_i^1 = \theta_{\text{start}} + s_i \cdot (\theta_{\text{end}} - \theta_{\text{start}}) \ . \tag{1}$$

Here, $s_i \in [0, 1]$ stands for the fraction of path length up to the middle of the ith segment, and is computed as

$$s_i = \frac{\sum_{k=1}^{i} \|W_k - W_{k-1}\| + \frac{1}{2}\|W_{i+1} - W_i\|}{\sum_{k=1}^{M} \|W_k - W_{k-1}\|} \ . \tag{2}$$

For every subpath we introduce a parameter $\lambda_\theta \in [0, 1]$ that blends the rotational behavior between the initial (θ_i^0) and minimized (θ_i^1) values according to

$$\theta_i = (1 - \lambda_\theta) \cdot \theta_i^0 + \lambda_\theta \cdot \theta_i^1 \ . \tag{3}$$

For $\lambda_\theta = 0$, the initial θ_i^0 orientations are used in the corresponding subpath as shown in Fig. 4 (left). Choosing $\lambda_\theta = 1$ uses the θ_i^1 and thus achieves a minimal change of rotations as depicted in Fig. 4 (right) which however might not be free of collisions anymore. The optimization starts with all $\lambda_\theta = 0$ and continuously changes these parameters to obtain more efficient paths that are still collision-free.

IV. Velocity Profiles

The speed at which a path $Q(u)$ is traversed is determined by the mapping $u(t)$, such that $Q\left(u(t)\right)$ determines the pose and velocities of the robot over time t. For every path $Q(u)$ our method seeks to compute the mapping $u(t)$ that corresponds to the fastest velocity profile that respects given dynamic constraints. The algorithms are based on the approach presented in our previous work [17] and extended for holonomic robots. A trajectory $Q(u(t))$ defines velocities by its first derivative with respect to time t,

$$\dot{Q}(u(t)) = Q'(u(t)) \cdot \dot{u}\left(t\right), \tag{4}$$

where the dot and the prime denote derivatives with respect to t and u, respectively. We specify finely discretized support points along the path, e.g., whenever the robot would travel more than 0.02 m or rotate more than 0.02 rad after the preceding support point. For these supports we compute the maximum $\dot{u}(t)$ that satisfies a set of constraints. The translational and rotational velocities of the platform can be limited to $v_{\max}, \omega_{\max}$ for safety reasons, e.g., in the vicinity of obstacles. The resulting constraint for $\dot{u}(t)$ is given by

$$\dot{u}(t) \leq \frac{v_{\max}}{\|Q'_{xy}(u(t))\|}, \quad \dot{u}(t) \leq \frac{\omega_{\max}}{|Q'_{\theta}(u(t))|} \, . \tag{5}$$

To prevent skidding and protect sensitive payload, a maximal centripetal acceleration a_c can be enforced by

$$\dot{u}(t) \leq \sqrt{\frac{a_{\mathrm{c}}}{\|Q'_{xy}(u(t))\| \cdot |Q'_{\theta}(u(t))|}} \, . \tag{6}$$

Between the supports we assume constantly accelerated motion. To ensure safe transportation of sensitive payload, we also limit the range of allowed acceleration and deceleration in these intervals. Given $\dot{u}(t)$ we can now compute $u(t)$.

The specific algorithm to calculate velocity profiles is not integral to our method and could also be implemented using different approaches, e.g., the time scaling algorithm [10].

V. Trajectory Optimization

The optimization starts with a close approximation of the straight-line path provided by the global path planner. By subdividing long segments, the approximation error can be kept below the map resolution due to the convex hull property of Bézier splines. Typically, the optimization can substantially reduce the time of travel by widening curves, performing simultaneous translation and rotation, and adjusting the distance to obstacles that impose a speed limit.

Direct optimization of spline segments might theoretically be possible, but it would be very complex due to the high number of parameters and their interdependence. Therefore, our optimization relies on the meaningful higher-level parameters provided by the path representation, see Sect. III.

The optimization repeatedly iterates through the list of tunable parameters as long as planning time is left, and separately optimizes each parameter using a derivative-free gradient descent [17]. Each step involves modifying a parameter, checking the resulting trajectory for collisions and computing the associated cost. The optimization terminates

collides_circ(circle C, distmap D)
$(c_x, c_y) \leftarrow$ center(C)
return $(D(c_x, c_y) \leq$ radius(C))

collides_rect(rectangle R, distmap D)
$(c_x, c_y) \leftarrow$ center(R)
if $D(c_x, c_y) \leq r_i$ **then**
return *true*
else if $D(c_x, c_y) > r_o$ **then**
return *false*
compute rectangles R_1, R_2
return (collides_rect(R_1, D)
$\vee$ collides_rect(R_2, D))

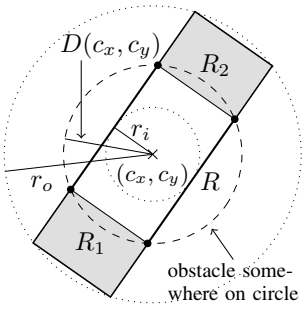


Fig. 5. Efficient collision checks for circular and rectangular elements. The drawing shows a rectangular element R with inner and outer diameter r_i and r_o. Only if the obstacle distance $D(c_x, c_y)$ at the center of R is between r_i and r_o, the result depends on recursive collision checks of R's subparts R_1 and R_2. Exceeding recursion can be terminated assuming collision.

if an iteration through all parameters did not result in a considerable cost reduction.

Checking the trajectories for collisions dominates the computational cost during optimization. For efficient checks, our algorithm approximates the shape of the robot and its payload by a set of circles and rectangles. Using a dynamically updated distance map $D(x, y)$ of the environment [19], these elements can be efficiently checked for collisions with the two methods outlined in Fig. 5.

Since the optimization starts from a valid trajectory and rejects colliding variants, it can provide valid trajectories with decreasing costs at any time. As another consequence, the initial path predefines the topological route between obstacles. However, this can be overcome by running the algorithm in parallel with different path inputs, e.g., with the n-best topologically different paths from start to goal.

VI. Experiments

We conducted experiments using the KUKA omniRob shown in Fig. 1. It has a drive like the Uranus robot [20], a rectangular shape of 1.15×0.86 m, and a mass of 250 kg. Two SICK laser range finders are used for self-localization and detection of obstacles. The motion planning system was executed on an Intel® Core™2 Duo 2.2 GHz. In our experiments we allowed velocities up to 1.2 m/s and used travel time as cost where not stated otherwise.

The trajectories $Q(u(t))$ were executed using an error feedback controller that in every time step t computes a velocity vector $V = \langle v_x, v_y, v_\theta \rangle^T$. This vector is determined using the trajectory $Q(u(t))$ and its derivative $\dot{Q}(u(t))$,

$$V = \dot{Q}(u(t + t_{\text{del}})) + \kappa \odot (Q(u(t)) - X(t)) \,. \qquad (7)$$

The first term on the right-hand side is the feed-forward signal that accounts for a command execution delay t_{del}. The second term realizes the error-feedback with the pose estimate X provided by the robot's odometry. Finally, $\odot$ denotes the component-wise product of vectors. The parameters have been set to $\kappa = \langle 2, 2, 0.2 \rangle^T$, and $t_{\text{del}} = 0.1$ s.

The motor speeds for the four actuated wheels are computed from V and X via the motion equations for the platform, in our case for the Uranus [20, Eq. (6.2.13)].

In dynamic environments, it is not reasonable to plan velocities for the entire path as it might be obstructed by

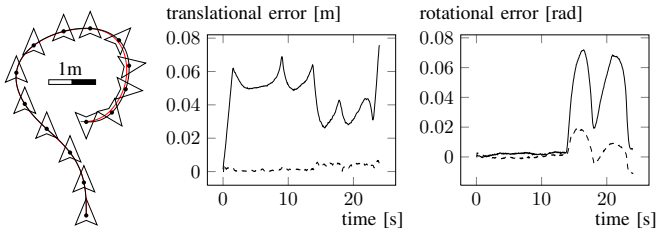


Fig. 6. *Left:* Execution of a reference trajectory drawn in black, driven path in red. *Middle/right:* the plots show the resulting deviations without (solid) and with error feedback and delay compensation (dashed).

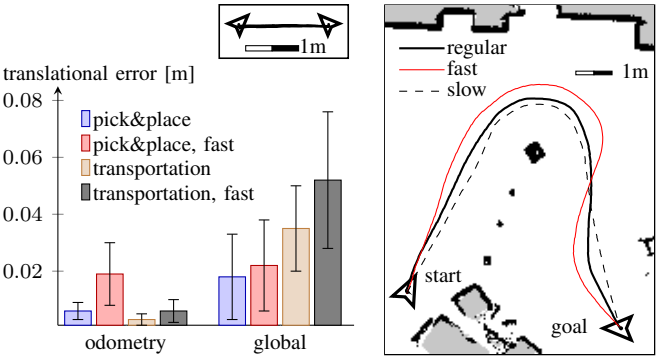


Fig. 7. The bar plot shows the translational tracking error (mean, standard deviation) for the experimental tasks: the pick&place task (top) is shown with an overlay of all driven trajectories and the transportation task (right) is shown with example trajectories for different velocity constraint settings.

unexpected obstacles. Frequent replanning is also required to cope with localization errors and odometry drift. In our experiments we found a good tradeoff by using the next four waypoints and replanning every $\Delta_t = 1.6$ s, which is the time for path planning, optimization, and communication delays.

Besides one initial planning interval of Δ_t all computation is done online, i.e., while the robot is moving. Our experiments evaluate tracking accuracy, behavior in difficult situations, and reaction to different constraint settings and waypoint planners. Due to availability of admissible initial paths and the anytime characteristic of the optimization, our system was always able to reach the specified goal pose.

A. Open-loop and closed-loop trajectory tracking

We evaluate the suitability of our trajectory representation on the basis of tracking errors with and without error feedback in the controller. The reference trajectory, shown in Fig. 6 (left), specifies a hook-shaped curve with roughly 25 s of travel time and exploits the holonomic capabilities of our platform. Fig. 6 also presents plots of the translational (middle) and rotational (right) deviation of the actual robot pose from the planned pose for every point in time, as measured by the odometry of the robot. Even when driving with feedforward commands only, the translational errors were below 0.08 m and the rotational errors below 0.08 rad. With feedback control the errors were below 0.01 m and 0.02 rad. These low errors show that our approach generates trajectories that can accurately be followed by the robot.

B. Performance in real-world scenarios

This experiment evaluates the applicability of our approach in four industrial real-world scenarios. In a medium-

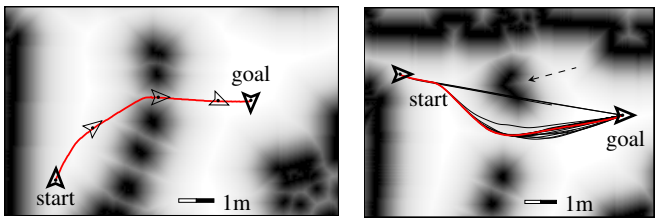


Fig. 8. *Left:* trajectory of the holonomic robot when driving through a narrow passage. *Right:* avoiding an unmapped and unexpected obstacle (arrow) by replanning trajectories. The trajectories are shown on distance maps of the environment, where darker spots are closer to obstacles.

distance transportation task the robot has to reach the goal with a specified orientation, as shown in Fig. 7 (right). The task was executed ten times yielding an average travel time of 35.4 s. To challenge our system and to test its adaptivity, the task was executed another five times with raised velocity limits. Although the resulting trajectories were longer, this reduced the travel time to 28.4 s. In a single run with slow admissible velocities it took the robot 64.7 s to execute the trajectory. The low standard deviation of 0.2 s in all settings shows the consistent behavior of the system. The effects of different constraint settings on the path shape can be seen in Fig. 7 (right). The faster the robot can drive, the more the system elongates trajectories to make wider curves and to stay away from obstacles, as small distances and sharp curves impose limits on the speed.

The second task is a typical pick&place application that requires the robot to repeatedly travel between two poses with a waiting time of 0.2 s between each trip. The task was executed for two minutes yielding 19 very similar trajectories as shown in Fig. 7 (top). In a run with higher admissible velocities and accelerations, the robot achieved 30 iterations in the same time interval.

The translational tracking errors for both tasks and velocity settings are shown in Fig. 7 (left). The errors are the deviation of the robot from its planned pose, averaged over time. When we measure the deviation based on the odometry of the robot, the errors are below 0.02 m. The errors according to the global self-localization are below the map resolution of 0.05 m, even for the high velocity settings.

To demonstrate the capability of our system to handle challenging situations, we initialized the optimization with a straight line path through a narrow passage with 0.07 m clearance on both sides of the robot, shown in Fig. 8 (left). This is considerably narrow, given the map resolution of 0.05 m. As shown in the figure, the robot aligned itself with the passage to maximize obstacle distance.

The last scenario involved a cardboard box thrown right in front of the moving robot as unexpected obstacle, see Fig. 8 (right). The straight trajectories planned at first would collide with the box. After sensing the obstacle, the system smoothly circumvents it (red) by generating new trajectories that join the original ones without discontinuities.

The results of our experiments demonstrate that our approach generates smooth and precise motion, effectively adapts to varying parameters like velocity limits, and handles challenging situations that occur in real applications.

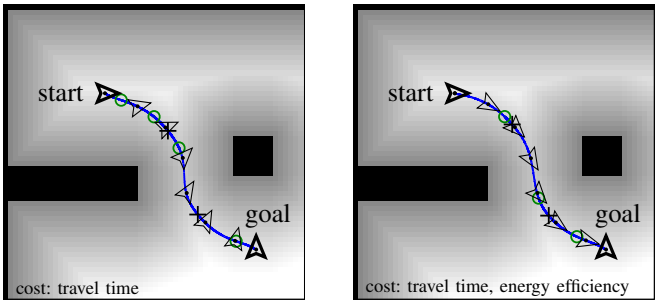


Fig. 9. *Left:* optimized trajectory using time of travel as cost. *Right:* using an additional penalty for non-forward motion, which is motivated by a higher energy efficiency. The main difference is the rotational behavior.

C. User-defined cost function

We conducted experiments in simulation to test our system with different cost functions. Depending on the application, other aspects might be appropriate besides the time of travel. For example, it can be desirable to bias the system to generate movements anticipated by humans, e.g., moving forward rather than sideways. Depending on wheel design this can also be more energy efficient due to lower friction.

We implement this by adding a penalty F for non-forward motion to our cost function, $cost = t_{\text{travel}} + \alpha \cdot F \cdot t_{\text{travel}}$, where α controls the influence of F. The penalty is computed from the angular difference γ between the robot's orientation and the direction of travel, integrated over the trajectory length l,

$$F = \frac{1}{l} \int_0^l 1 - |\cos \gamma(s)| \, \mathrm{d}s \, . \tag{8}$$

We optimized the same initial trajectory with two different settings for α. Although the two-dimensional shape of the traveled paths look similar, the trajectories differ substantially in their rotational behavior, see Fig. 9. For $\alpha = 0$ the robot needed 14.46 s and continuously rotated during translation to reach the specified goal orientation. With $\alpha = 1$ the resulting trajectory took 16.28 s to traverse, but mostly used forward motion until shortly before reaching the goal.

D. Influence of the choice of waypoint planner

Our system uses waypoints from a global path planner to generate an initial trajectory as starting point for the optimization. Naturally, the choice of planning algorithm can have an influence on the final result.

2D path planners ignore orientation and therefore require the robot's circumcircle to be obstacle-free along the path, see Fig. 2 (A). Waypoint planners that account for the orientation of the robot during forward motion are able to find paths through passages as shown in Fig. 2 (B). Planners operating in the full holonomic configuration space (C-space) can also find paths through passages that require lateral or diagonal movements as in Fig. 2 (C). Our approach can generate admissible spatio-temporal trajectories for all these situations, given a suitable waypoint planner.

To assess the robustness of our system against the choice of waypoint planner, we compare the optimization results for the CARMEN 2D value iteration planner, a C-space planner, and a hybrid approach that uses Voronoi graphs where

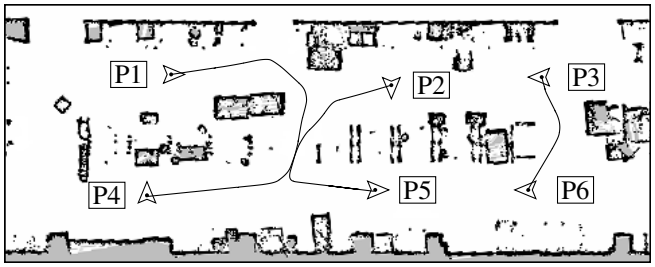


Fig. 10. Map with start/goal poses used in our evaluation of different path planners in combination with our trajectory generation method. The figure also contains selected example trajectories between the end poses.

TABLE I

NUMERICAL RESULTS OF GLOBAL PATH PLANNER COMPARISON.

Planner	t_{plan}	t_{opt}	$t_{\text{travel-I}}$	$t_{\text{travel-O}}$
2D value iteration	0.022 s	0.55 s	82.37 s	38.11 s
Voronoi	0.009 s	0.46 s	36.28 s	28.46 s
C-space	4.864 s	0.43 s	31.49 s	27.19 s

possible and resorts to C-space planning when necessary. We selected six start/goal poses on a map of a factory floor as shown in Fig. 10 and let a simulated robot perform travel tasks for all of the 30 start-goal combinations.

Tab. I shows the average times needed for planning (t_{plan}), optimization (t_{opt}), and execution of the initial and optimized trajectories ($t_{\text{travel-I}}$ and $t_{\text{travel-O}}$, respectively). As expected, the C-space planner requires substantially more time to find a path compared to the others, which prevents its use in most practical applications. On the other hand, it generates waypoints that fully exploit the holonomic capabilities of the robot which leads to trajectories with the shortest initial and optimized travel times. The hybrid planner uses Voronoi paths which results in substantially different waypoints. Nevertheless, the optimization results are comparable, but are achieved in a much shorter planning time. Additionally, this planner has the same completeness properties as the C-space planner. In contrast, the 2D value iteration plans with the circumcircle of the robot. Thus it has to take detours in some runs to avoid the narrow passage between P1 and P2 which increases execution times. Additionally, its waypoints have to maintain a larger distance to obstacles in general, which also causes longer initial paths. However, our optimization can compensate this drawback to a large degree by adjusting the trajectory shape, which causes a substantial decrease in execution time as shown in Tab. I.

VII. CONCLUSION

In this paper, we presented a novel approach to kinodynamic trajectory generation for holonomic platforms. Our approach starts with an initial, collision-free trajectory which can directly be computed from waypoints calculated by a global path planner. Our novel path representation can model the spectrum between both, pure translations alternating with turns on the spot and continuous rotation during translation on smoothly curved trajectories. Relying on this representation, our method iteratively optimizes the given trajectory according to a user-defined cost function and under the constraint that no collisions occur. The output of our anytime algorithm is a smooth trajectory that complies with the dynamic constraints of the robot and can therefore be tracked by the robot with high precision.

Our approach has been implemented and evaluated on an industrial holonomic platform. The experimental results show that it yields highly versatile motion behaviors even in complex situations with narrow passages or unexpected obstacles. During all our experiments the positioning error relative to the planned trajectory was below the map resolution of 0.05 m. Further experiments demonstrated the robustness of our system against the choice of global path planner, and its flexibility when adapting to different cost functions.

REFERENCES

[1] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Intl. Conf. on Robotics and Automation*, vol. 1, Detroit, USA, 1999.

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

[3] R. Rojas and A. G. Förster, "Holonomic control of a robot with an omnidirectional drive," *Künstliche Intelligenz*, vol. 20, no. 2, 2006.

[4] Y. Liu, J. J. Zhu, R. L. Williams II, and J. Wu, "Omni-directional mobile robot controller based on trajectory linearization," *Robotics and Autonomous Systems*, vol. 56, no. 5, 2008.

[5] D. J. Balkcom, P. A. Kavathekar, and M. T. Mason, "Time-optimal trajectories for an omni-directional vehicle," *The Intl. Journal of Robotics Research*, vol. 25, no. 10, pp. 985–999, 2006.

[6] O. Purwin and R. D'Andrea, "Trajectory generation and control for four wheeled omnidirectional vehicles," *Robotics and Autonomous Systems*, vol. 54, pp. 13–22, 2006.

[7] M. Likhachev and D. Ferguson, "Planning long dynamically-feasible maneuvers for autonomous vehicles," in *Robotics: Science and Systems Conference*, Zurich, 2008.

[8] K. Maček, G. Vasquez, T. Fraichard, and R. Siegwart, "Towards safe vehicle navigation in dynamic urban scenarios," *Automatika*, 11 2009.

[9] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *Intl. Journal of Robotics Research*, vol. 20, no. 5, 2001.

[10] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 2005.

[11] Y. Yang and O. Brock, "Elastic roadmaps–motion generation for autonomous mobile manipulation," *Autonomous Robots*, vol. 28, no. 1, pp. 113–130, 2010.

[12] J. Minguez and L. Montano, "Nearness diagram navigation (ND): Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, 2004.

[13] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *The Intl. Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[14] J. Connors and G. Elkaim, "Manipulating B-Spline based paths for obstacle avoidance in autonomous ground vehicles," in *National Meeting of the Institute of Navigation*, San Diego, USA, 2007.

[15] F. Lamiraux, D. Bonnafous, and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 967–977, 2004.

[16] T. Fraichard and V. Delsart, "Navigating dynamic environments with trajectory deformation," *Journal of Computing and Information Technology*, vol. 17, 2009.

[17] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), St. Louis, MO, USA*, 2009.

[18] Z. Shiller and Y. Gwo, "Dynamic motion planning of autonomous vehicles," *IEEE Trans. on Robotics and Automation*, vol. 7, 1991.

[19] B. Lau, C. Sprunk, and W. Burgard, "Improved updating of Euclidean distance maps and Voronoi diagrams," in *IEEE Intl. Conf. on Intelligent Robots and Systems (IROS), Taipei, Taiwan*, 2010.

[20] P. Muir, "Modeling and control of wheeled mobile robots," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 1988.