Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)

Clothoidal Local Path Template for Intention Estimation by Assistive Mobile Robots

Kevin Denis, Johan Philips, Herman Bruyninckx, Eric Demeester
KU Leuven, Department of Mechanical Engineering
Wetenschapspark 27, 3590 Diepenbeek, Belgium
eric.demeester@kuleuven.be

Abstract

This work proposes and evaluates various improvements to a circular local path template (LPT) that we have designed in the past to estimate driver intents and to provide navigation assistance to wheelchair drivers. This LPT may also be useful in other mobile robotics applications such as for intuitive teleoperated control, for fast collision checking in path planning or for obstacle avoidance algorithms. The LPT consists of a large but fixed set of paths in the mobile robot's local neighborhood. Based on an efficient look-up table, the LPT paths' lengths are adjusted such that they are collision-free. However, experiments have shown that in dense environments insufficient circular paths are found, which impedes correct intention estimation and thus navigation assistance. In this work, the use of clothoidal paths rather than circular paths is evaluated. This substantially improves the capability to find complex paths in dense areas. Furthermore, we adapt the LPT to deal with dynamic obstacles of random shape using motion prediction estimates of these objects.

Introduction

The ability to move around to any desired location is critical to all human development, activity and interaction. With the increase in average age in almost all nowadays societies, the loss of mobility caused by reduced physical capabilities frequently leads to a loss in social contact and therefore quality of life. The use of robotic technologies can give back a level of mobility and sense of autonomy to the elderly or disabled. For example, powered wheelchairs enable people with motion impairments to regain movement control and thus to re-engage in more frequent human interactions. However, this renewed mobility comes with its own constraints. Wheelchairs are relatively large compared to their indoor environment. In order to navigate the chair without colliding, a significant degree of dexterity is needed. Executing fine manoeuvres is even more demanding due to the non-holonomic characteristics of most wheelchairs. It is therefore often a tiresome and frustrating task for elderly and disabled people to control their wheelchair properly.

Similar to several other research groups, we have developed navigation assistance algorithms in the past to lower the users' workload for maneuvering their wheelchair, see e.g. (Demeester et al. 2014; Demeester and Hüntemann 2016). This is performed by estimating the driver's navigation intention and by providing manoeuvring assistance based on the estimated intention. In order to motivate the rationale behind and the requirements for the local path template (LPT), this section briefly explains our intention estimation approach using Figure 1. The remainder of this paper then solely focuses on the local path template, i.e. on fast computation of many collision-free paths in a robot's local neighborhood. This LPT may also be useful in other mobile robotics applications such as for intuitive teleoperated control or for autonomous mobile robots that need fast collision checking for path planning or obstacle avoidance.

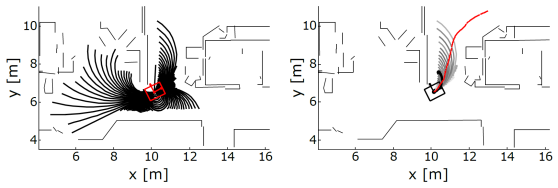

Figure 1: (left) Fast computation of a set of local, circular, collision-free trajectories for a robotic wheelchair in an apartment; the red rectangle represents the wheelchair. We consider these paths as hypotheses $\boldsymbol{i}_k$ regarding the driver's navigation intention. (right) Plan recognition results at that location: each circular path gets assigned a probability (the darker a path, the higher its probability), the red path is the user's actual, but unknown, navigation plan. Locally, the estimated intent corresponds to the user's actual intent.

Estimation of the driver's navigation plan is performed as follows. We model the driver's intention $\boldsymbol{i}_k$ at time instant k as a path, a succession of desired robot states from the current state $\boldsymbol{x}_{current}$ to a goal state $\boldsymbol{x}_{goal}$: $\boldsymbol{i}_k = \{\boldsymbol{x}_{current}, \ldots, \boldsymbol{x}_{goal}\}$. The robot's state is defined as $\boldsymbol{x} = [x\ y\ \theta]^T$ with (x, y) the robot's Cartesian position and θ its orientation. A probability $p(\boldsymbol{i}_k)$ is computed for each local path $\boldsymbol{i}_k$ as shown in Equation 1. Bayes' theorem is adopted to compute the posterior probability p_{post} on $\boldsymbol{i}_k$,

based on the given steering signals ($\boldsymbol{u}_k$) and a history ($\boldsymbol{H}$) of driver signals, robot actions, robot poses and sensor readings. p_{prior} is the prior probability distribution on the set of paths and p_{user} is a user model, which models how the driver transforms a particular intention $\boldsymbol{i}_k$ into a certain steering signal $\boldsymbol{u}_k$. η is a scale factor normalizing the probability distribution. For a more detailed description, we refer to (Hüntemann et al. 2008; 2013).

$$p_{post}(\boldsymbol{i}_k \mid \boldsymbol{u}_k, \boldsymbol{H}_{0:k}) = p_{user}(\boldsymbol{u}_k \mid \boldsymbol{i}_k, \boldsymbol{H}_{0:k}) \cdot p_{prior}(\boldsymbol{i}_k \mid \boldsymbol{H}_{0:k}) \cdot \eta. \quad (1)$$

In the past, we adopted circular local paths as an approximation of potential driver navigation plans $\boldsymbol{i}_k$, see Figure 1. This seemed a reasonable choice, as the wheelchair's instantaneous motion corresponds to a circular path, given its nonholonomic constraints. Since we adopted a haptic joystick to provide navigation assistance feedback to the user at a high frequency (Vander Poorten et al. 2012), we needed a very fast and accurate collision checking algorithm to verify potential collisions between all paths $\boldsymbol{i}_k$ and detected obstacles. This was obtained through an obstacle-based look-up table (Demeester et al. 2012) that uses a large but fixed set of paths expressed in the robot's local coordinate frame; we call this fixed set of paths a "local path template". However, experiments showed that in dense environments, e.g. when entering narrow doors or elevator entrances, too few or even no circular paths at all were found through these narrow passageways as shown in Figure 2 (left). As a result, no appropriate intent estimation and thus navigation assistance could be provided under these circumstances.

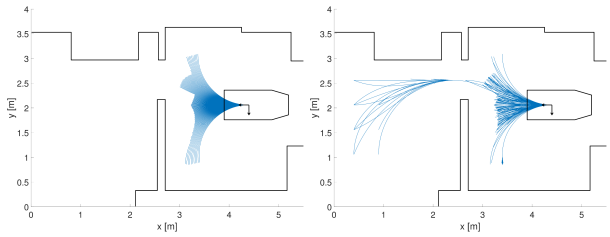

Figure 2: Circular (left) versus clothoidal (right) curves to find paths through narrow doorways, showing that the clothoidal paths have a higher chance to find paths in dense environments.

This work's goal is to improve the richness of the local paths such that substantially more local paths through narrow passageways can be found. More specifically, we explore whether clothoidal geometries can be adopted for this, see Figure 2 (right). The remainder of the paper is organized as follows. First, the clothoidal local path template is described. Then, an extension of this local path template is given that is able to deal with dynamic obstacles. Next, the clothoidal local path template is evaluated and its improved planning capability in dense environments is demonstrated. The paper concludes with some tracks for future work.

Clothoidal Local Path Template

This section describes the formation of a clothoidal local path template (LPT). The LPT proposed by Demeester et al. (2012) consists of a fixed set (or *template*) of paths that are compatible with the robot's kinematics, that start from the robot's current pose (hence *local*), and that are expressed in the robot's local coordinate frame. In (Demeester et al. 2012), kinematically feasible paths are obtained with the forward generation method, i.e. by integrating achievable linear and angular velocity (v, ω) pairs over a certain amount of time Δt. Another option to generate paths is to adopt the inverse generation method as proposed by Pivtoraiko and Kelly (2012). They introduce a novel search space, the State Lattice, consisting of a set of discretised states interconnected with kinematically feasible paths, called Motion Primitives (MPs).

Our clothoidal LPT is created by using a Local State Lattice (LSL) (a local version of the State Lattice of Pivtoraiko and Kelly) and an obstacle-based lookup table for collision checking. The method consists of an offline phase (steps 1 to 4) and an online phase (step 5):

1. Generation of a multi-size, multi-resolution grid (fine nearby, coarse farther away) of discretised poses.
2. Each discretised pose in a chosen Region Of Interest (ROI) around the origin is connected with a clothoidal curve. If this path is kinematically feasible, it is added to the set of MPs.
3. To ensure a certain degree of flexibility, step 2 is repeated at certain grid cells, called Expansion Positions (EPs). By doing this, certain paths in the LSL will consist of a sequence of clothoidal curves.
4. A look-up table is built for this fixed set of paths, assuming a static robot geometry. This table stores through which grid cells the robot passes when it follows the paths.
5. Using the look-up table and perceptual information of the robot's surroundings, the lengths of the paths in the clothoidal LPT are adjusted in order to be collision-free.

Steps 1 to 4 are computed once for a given LPT and robot geometry. As these elements do not change while driving around, these steps are not recomputed online. The LPT's paths are expressed in the local robot coordinate frame. Hence, the look-up table does not need to be built for all possible robot locations, nor when the robot changes location. Only in step 5 information regarding the robot's environment is adopted; therefore, this step needs to be executed online. The following paragraphs describe each of the steps.

Step 1 - Generation of a multi-size, multi-resolution grid The LSL constitutes a fixed set of feasible trajectories originating from the robot's current pose to nearby end poses. The first step in the creation of the LSL is to sample the reachable space surrounding the robot by using a multi-size grid (MSG). A clothoidal curve will then be connected to each of these discrete poses, if possible. The MSG is composed of three different sizes (fine, medium, coarse), as shown in Figure 3 (top). The inverse generation method

is adopted to create a set of MPs. First, the environment is sampled and a boundary value problem is solved to connect these discrete states with feasible paths using a defined curve geometry. The inverse approach is preferred in this implementation as it makes the state discretization the driver of the design of the LSL. Parameters can then be fine-tuned depending on the application. The main reason to adopt a MSG is to limit the number of paths to end points far away from the origin. For intention estimation, it may be inefficient to look too far into the future, since the environment and the intention of the driver may change. An end pose located in the coarse grid therefore represents a large group of poses compared to an end pose in the fine grid. A ROI selects the Candidate End Poses (CEPs) of the grid cells in the immediate surrounding of the origin. This step is illustrated in Figure 3 (top). Depending on the chosen geometry of the MP and the constraints of the robot, a CEP will be reachable or not. If it is reachable, the path leading to the CEP will be added to the set of MPs. Table 1 contains values of the used parameters.

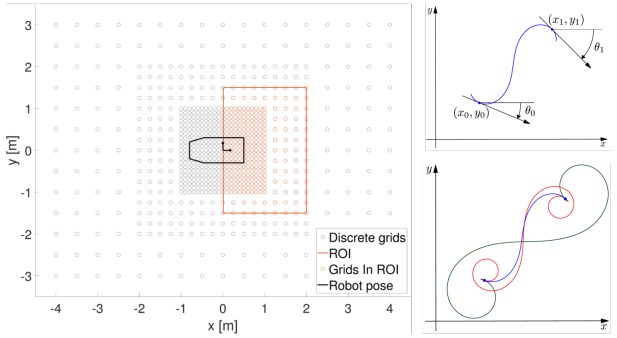

Figure 3: (left) Multi-size grid (MSG) and region of intrest (ROI) shown at the origin in the robot's local coordinate frame. Three grid sizes are used to sample the environment. Some grid cells in the ROI become candidate expansion points (CEPs), which will be connected to the origin by a geometrical curve. (right, top) Notation for the G1 Hermite interpolation scheme, and (right, bottom) yielding multiple solutions using clothoids (Bertolazzi and Frego 2015).

Step 2 - Connection of grid cells in the ROI with a geometrical curve Once the CEPs around the origin are determined, a geometrical connection has to be established between the origin and these CEPs. In this work, clothoids have been adopted, but other choices are possible. Clothoids (also called Euler or Cornu spirals) are curves whose curvature changes linearly with their arc length, see Equation 2. In Equation 2 to 5, s denotes the path coordinate (length along the curve measured from the initial position), x and y correspond to the robot's position along x and y axis respectively, θ is the robot's orientation, κ denotes the path curvature and apostrophes indicate derivatives. Clothoids are frequently used for the design of railway tracks (Cope 1993) and highway design (Baass 1982) to connect a tangent to a circular curve, resulting in a continuous curvature profile. Joining a tangent and a circular curve directly would result in a discontinuity, thus leading to an instantaneous change in the centripetal acceleration, causing discomfort to passengers. Clothoids have also been applied in path planning for

Table 1: LSL parameters and corresponding values.

parameter	value	description
dx_1, dy_1	0.10 m	Discretization of the fine grid
$\{x, y\}_{1,max}$	1.00 m	Width, height of the fine grid
dx_2, dy_2	0.25 m	Discretization of the medium grid
$\{x, y\}_{2,max}$	2.00 m	Width, height of the medium grid
dx_3, dy_3	0.50 m	Discretization of the coarse grid
$x_{3,max}$	4.00 m	Width of the coarse grid
$y_{3,max}$	3.00 m	Height of the coarse grid
$d\theta$	$\pi/8$ rad	Angular discretization
x_{ROI}	2.00 m	x-distance (only +) of the ROI
y_{ROI}	1.50 m	y-distance (+/-) of the ROI
κ_{max}	1 m^{-1}	Maximum allowed curvature
dx_{EP}	0.50 m	l_1 distance between EPs
res_{OG}	2 cm	Resolution of the OG
res_{path}	1 cm	Resolution of the path

mobile robots (Fleury et al. 1995; Brezak and Petrović 2011; Kelly and Nagy 2003; Scheuer and Fraichard 1997). An iterative process has to take place in order to connect one pose $\boldsymbol{p}_0$ to another $\boldsymbol{p}_1$ using a clothoid, because there is no unique G1 fitting solution, see Figure 3 (bottom, right). Extensive research has been done to find stable numerical solutions to calculate the Hermite G1 interpolation with a single clothoid curve, which can be formulated as a system of three nonlinear equations (yielding multiple solutions), see Equations 3 to 5. Walton and Meek (2009) designed their algorithm to handle three different situations, straight lines, circles and clothoids to then solve only one single nonlinear equation. However, when the solution of a clothoid approaches the shape of a circle ($\kappa' \approx 0$) or a straight line ($\kappa = \kappa' \approx 0$) the root of the nonlinear equation becomes ill-conditioned, resulting in numerical errors. Bertolazzi and Frego (2015) solve this problem by recasting the problem into a well-conditioned zero of a unique nonlinear equation. Moreover, their algorithm does not treat straight lines, circles and clothoids differently and thus achieves robust results at the transition zones. Their solution has been used without modification to generate the clothoids, see Figure 4 (a).

$$\kappa(s) = \kappa' s + \kappa_0, \kappa', \text{change of curvature, constant} \quad (2)$$

$$\theta'(s) = \kappa(s), \text{where } \theta(0) = \theta_0, \theta(l) = \theta_1 \quad (3)$$

$$x'(s) = \cos\theta(s), \text{where } x(0) = x_0, x(l) = x_1 \quad (4)$$

$$y'(s) = \sin\theta(s), \text{where } y(0) = y_0, y(l) = y_1 \quad (5)$$

By applying G1 interpolation, the smooth transition of curvatures is not guaranteed; this would require (computationally more expensive) G2 Hermite interpolation. As G1 continuity is acceptable for this application, G2 is not applied. If G2 continuity is preferred, e.g. for the generation of local paths for car-like vehicles, the use of piecewise clothoid

curves is necessary, as developed in (McCrae and Singh 2009).

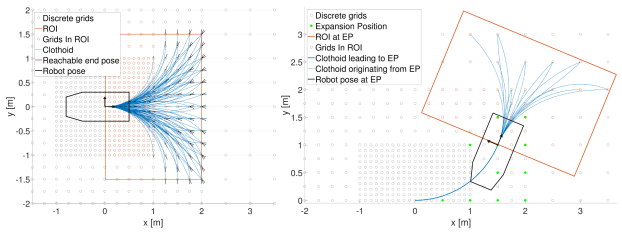

Figure 4: (left) Set of MPs based on clothoids, connecting the origin with feasible states within the ROI. (right) Example of the EP and ROI at [1.5, 1, 45°]. CEPs are now linked with the pose that the robot would have at that EP and not with the origin.

Step 3 - Expansion points Finally, in order to obtain a larger variety of paths, the procedure of connecting CEPs to the set of MPs is repeated at certain discrete positions, which we call Expansion Positions (EPs). EPs are defined as reachable end poses directly connected to the origin, with a Manhattan Distance (l_1-norm) equal to a multiple of dx_{EP}. It should be noted that those discrete poses in the ROI (at that EP) will be connected with the pose at that EP, and not at the origin. There will be significantly less CEPs if the EP is further away from the origin, due to the use of the MSG, thereby reducing the number of paths leading far away from the origin. Figure 4 (right) illustrates this procedure. Repeating this procedure for every EP results in a LSL, a set of paths starting from the origin and connecting end poses in the surrounding of the robot with feasible clothoidal trajectories. This is illustrated in Figure 5. Algorithm 1 provides an overview of the complete procedure. The output of this algorithm is the LSL data structure as presented in Table 2, containing all required information to reconstruct the set of paths.

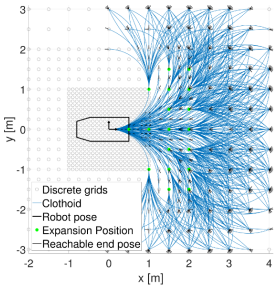

Figure 5: A LSL based on clothoids is obtained after repeating the procedure shown in Figure 4 (b) at every EP. This connects every feasible end pose close to the robot whilst moving forward with one or two clothoids. Backward movements and on-the-spot-turning are also implemented but are not illustrated in the present figure to maintain readability.

Step 4 - Offline construction of lookup table for collision checking Once the set of feasible clothoidal trajectories are defined, a lookup table is constructed offline to quickly evaluate online which paths are collision-free and to adjust their length if needed. The first step in the creation of the lookup table is to calculate the occupancy grid (OG) of each path. This is the space that the wheelchair will occupy when moving along a particular trajectory. The LSL structure defined in Table 2 contains all the information needed to reconstruct the calculated paths. Matrix $\boldsymbol{XY\Theta}$ contains all the discrete poses the mobile robot will adopt while following the path between the start pose $\boldsymbol{p}_0$ and the end pose $\boldsymbol{p}_1$. For each pose along the path, the path length (s) and path curvature (κ) at that position are in the table as well. The resolution for every curve is fixed at $s(k) - s(k-1) = res_{path} < res_{OG}/2$ to ensure that all cells of the OG are examined.

Table 2: REPRESENTATION OF THE LSL DATA STRUCTURE FOR A SINGLE PATH. This structure includes all necessary information to represent each path in the LSL and comprises precomputed data ($\boldsymbol{XY\Theta}, \boldsymbol{s}, \boldsymbol{\kappa}$). These vectors contain the position and orientation, path length and curvature along each path, discretized in such a way that $s_{k+1} - s_k < res_{path}$. The $blockIdx$ entry (used in the online phase) represents the position index at which a certain path is blocked, meaning that if the position index is increased by one, the path will result in a collision with the environment. Prior to taking obstacles into account, this index is put to $rowlength(\boldsymbol{XY\Theta}) + 1$, meaning that the path should not be shortened. Finally, each path has a unique ID.

$\boldsymbol{p_0}$	$\boldsymbol{p_1}$	L_{tot}	κ_0	κ'
1×3	1×3	0.516	0.78	-0.05
$\boldsymbol{XY\Theta}$	$\boldsymbol{s}$	$\boldsymbol{\kappa}$	blockIdx	ID
52×3	52×1	52×1	53	14

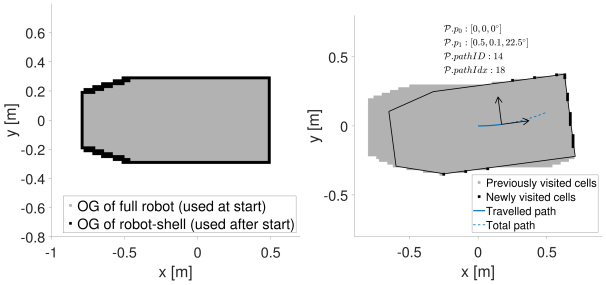

Figure 6: (a) Occupancy grid (OG) of the wheelchair's footprint. (b) The path OG is created by moving the robot along the path. Each time a grid cell is visited for the first time, it is stored along with the path-ID and position index along the path. The distance between 2 path indices is at most equal to $res_{path} < res_{OG}/2$ to ensure that no cells in the OG are skipped along the path.

The OG of the robot at the origin is shown in Figure 6. As the mobile robot moves along the path, grid cells that were not visited during the previous steps are stored, along with

the current position index and path ID. This procedure is shown in Figure 6 (b). The reason for storing this position index is to be able to efficiently adjust the path length in the presence of obstacles in the online phase (step 5).

Step 5 - Online computation of collision-free paths The lookup table is constructed based on the cells the robot occupies when following the clothoidal paths of the LPT. Each entry in this table corresponds to a grid cell that, if that cell is occupied, results in a collision with one or more paths from the LSL. The pathID and position index of each trajectory going through this cell are stored for that grid cell. In the online phase, while navigating, for all occupied cells in the grid map representing the robot's environment, their corresponding cells in the lookup table are retrieved. Next, each path length is updated by the provided information of that cell (pathID and position index). Note that this only takes place if this reduces the path length. This procedure is displayed in Algorithm 2. An example of the path length adjustment is illustrated in Figure 7. A single entry of the lookup table data structure can be found in Table 3.

Use of the LPT with Dynamic Obstacles

This section describes an extension of the clothoidal LPT to deal with dynamic obstacles of which the motion is known. Such a situation is shown in Figure 8. This work assumes we have access to an algorithm that estimates both the path of dynamic obstacles and their velocity profile along this path, see e.g. (Bennewitz et al. 2005; Bascetta et al. 2011). To

Algorithm 1 LSL ALGORITHM computing the structure $\mathcal{LSL}$ shown in table 2. Each new entry (row) in this structure represents a feasible path $\mathcal{P}$. Paths are added incrementally to the structure in line 9 and 19. The dot-operator (.) is used to access individual fields of each path in the $\mathcal{LSL}$.

Input: $userSettings$ to create matrix $\boldsymbol{P_{grid}}$ containing all discrete poses $\boldsymbol{p_{grid,k}}$.
Output: LSL structure ($\mathcal{LSL}$) with a set of feasible paths $\mathcal{P}$.

```
% calculate feasible paths at origin
1: p_0 ← [0, 0, 0]
2: P_grid ← CREATEMULTISIZEGRID(userSettings)
3: for all p_grid,i ∈ P_grid do
4:     if ISINROI(p_grid,i, p_0) then
5:         p_1 ← p_grid,i
6:         [x, y, θ, s, κ] ← GETCLOTHOIDDATA(p_0, p_1)
7:         if ‖κ‖_∞ ≤ κ_max then
8:             P ← [p_0, p_1, x, y, θ, s, κ]
9:             LSL ← ADDPATHTOLSLSTRUCT(LSL, P)
% calculate feasible paths at Expansion Positions
10: for all P ∈ LSL do
11:     if ISEXPANSIONPOSITION(P.p_1) then
12:         p_0 ← P.p_1        % end of path P is start of next paths
13:         for all p_grid,i ∈ P_grid do
14:             if ISINROI(p_grid,i, p_0) then
15:                 p_1 ← p_grid,i
16:                 [x, y, θ, s, κ] ← GETCLOTHDATA(p_0, p_1)
17:                 if ‖κ‖_∞ ≤ κ_max then
18:                     P ← [p_0, p_1, x, y, θ, s, κ]
19:                     LSL ← ADDPATHTOLSL(LSL, P)
20: LSL ← CLEANUPLSL(LSL)  % Remove non-unique paths
```

Algorithm 2 OBSTACLE-BASED LOOKUP TABLE (LUT) (online phase)

```
1: for all P ∈ LSL do                    % initialize path P as free:
2:     P.blockIdx ← ROWLENGTH(P.XYΘ) + 1
3: for all C ∈ GETCELLSOCCUPIEDBYLSL(LSL) do
4:     if GRIDMAPENVIROMENT(C) == occupied then
5:         for all P ∈ LUTPATHSINCELL(C) do
6:             if P.blockIdx > LUTBLOCKIDX(C, P) then
7:                 P.blockIdx ← LUTBLOCKIDX(C, P)
```

Table 3: REPRESENTATION OF A SINGLE CELL OF THE OBSTACLE-BASED LOOKUP TABLE DATA STRUCTURE. The entries x and y represent the location of the occupied grid cell by the robot (up to a resolution $res_{grid} = 2cm$) and contain all the paths (represented by $pathID$) going through this particular cell along with the position index along the path $\boldsymbol{XY\Theta}$ at which the robot occupies this cell for the first time. In particular, this table contains the occupied cell shown in Figure 7 at the top right corner.

x	y	$pathID$	position index
3.90 m	2.96 m	[763, 764, 779]	[202, 192, 226]

avoid collisions with dynamic obstacles, an optimal speed profile $v(t)$ along the LPT paths is calculated by computing a constraint optimization problem (COP). This method assumes a given motion model for each obstacle and an OG representing their shape. We do not impose any restriction on the obstacle's path or its shape (convex or concave), the number of obstacles, the robot's shape, or the environment's shape; this makes this approach flexible and generic. The

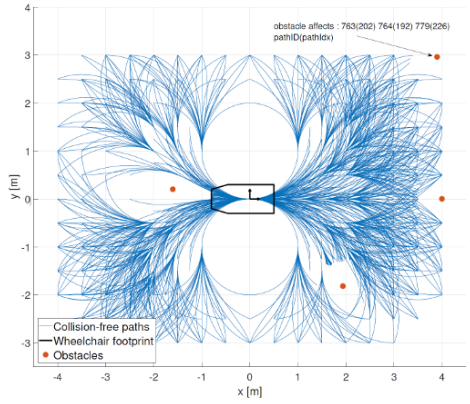

Figure 7: Example of fast online path length adjustment for 4 obstacle points. Each obstacle is exactly one grid cell (inflated in the figure for readability). For the top right obstacle, the affected paths stored in the look-up table are shown.

next section first explains the creation of a distance-time collision space (s, t space) where s is the distance along a fixed path of the LPT. This collision space will be used for a COP, using a dynamic model $\boldsymbol{f}$ of the wheelchair along with time-varying separating hyperplanes to provide collision-free mo-

tion by finding an optimal speed profile to reach the end of a given path.

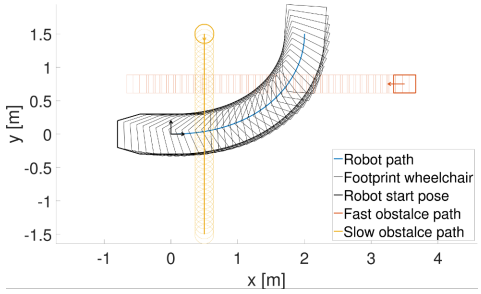

Figure 8: Dynamic obstacles potentially causing a collision with the mobile robot.

Distance-time Collision Space

The s, t space can be seen as a grid, within which all collisions between the robot positioned at a certain distance s along the fixed path and the time-varying position of the moving obstacle are calculated. This is computed with the following procedure:

1. The full OG of the robot along the fixed path is used, to determine the first and last impact time of the moving obstacle. This will narrow the search space along the t-axis (Figure 9 (a)).

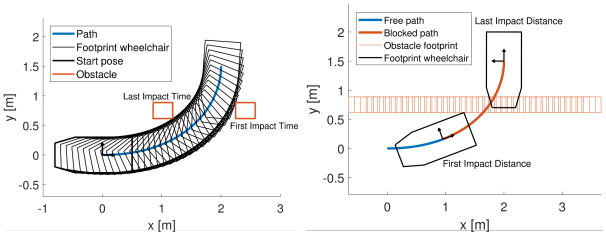

Figure 9: First and last impact time and distance are calculated to reduce the computational time of the individual collision states in the s, t space.

2. The full OG of the moving obstacle is determined by integrating its velocity. The first and last position on the fixed robot-path colliding with the moving obstacle is calculated. This will narrow the search space along the s-axis (Figure 9 (b)).
3. In the restricted space determined by $[t_{first}, s_{first}]$ - $[t_{last}, s_{last}]$, all discrete s, t collision states are calculated, determined by the path resolution (s) and the time resolution (t). An example at a fixed time $t = 1.25\,s$ for the obstacle is shown in Figure 10 (a). Repeating this for every time instance results in a s, t space grid, indicating which s, t pair results in a collision, see Figure 10 (b).
4. Further optimization can be performed on the obtained collision states. As time-varying separating hyperplanes will be used in the COP, it will be more efficient to only keep a convex shape of the obtained collision states. This convex shape can be further simplified when applying the following constraints: (i) assuming that the mobile robot cannot move backwards and (ii) time always moves forward. This results in a nearly rectangular shape determined by $[t_{first}, s_{first}]$ - $[t_{last}, s_{last}]$. The resulting shape of the obstacle in s, t space is not always rectangular, as can be seen in the (s, t - space) in Figure 11.

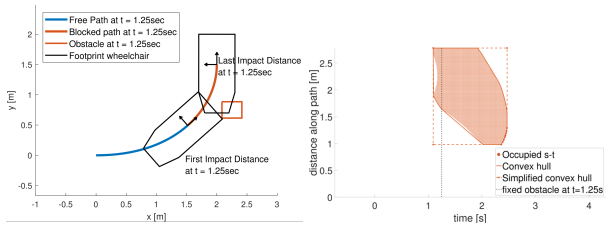

Figure 10: (a) The distance-time collision space is obtained by calculating the distance along the robot path resulting in a collision at a certain time with the moving obstacle. The dashed vertical line at t = 1.25s in the right figure corresponds to the situation shown on the left, where distances yielding a collision along the robot-path are marked in red for the position of the obstacle at t = 1.25s. These distances correspond to occupied cells at t = 1.25s in the s, t distance-time collision space at the right.

Optimal Speed Profile Calculation

A COP can be formulated to find an optimal speed profile given a distance-time collision space compliant with kinematic and dynamic constraints of the mobile robot and an objective, which is to arrive within the minimum amount of time at the end of the path. This is shown in Equation 6. This results in finding a path in the s, t - space, from position [0, 0] to $[s(end), t(end)]$ without colliding with the simplified convex hull of the collision states. This is shown in Figure 11.

$$\underset{(\boldsymbol{x}_{1:N}, u_{1:N}, \boldsymbol{a}_{1:N}, b_{1:N}, T)}{\text{minimize}} \quad T \tag{6}$$

subject to

$$\begin{aligned}
&\boldsymbol{x}_{k+1} = \boldsymbol{I}(\boldsymbol{f}(\boldsymbol{x}_k, u_k, h)), k = 1 \dots N, \\
&\boldsymbol{a}_k^T \boldsymbol{v}_i - b_k \geqslant 0, i = 1 : N_{vertices}, \\
&\boldsymbol{a}_k^T \boldsymbol{x}_k - b_k \leqslant -r_{safe}, \\
&\boldsymbol{x}_1 = [0, v_{start}], \\
&\boldsymbol{x}_{N,1} = \boldsymbol{s}(end), \\
&0 \leqslant \boldsymbol{x}_{k,2} \leqslant v_{max}, \\
&u_{min} \leqslant u_k \leqslant u_{max}, \\
&T \geqslant 0, \\
&\|\boldsymbol{a}_k\| \leqslant 1
\end{aligned}$$

The COP is formulated based on the multiple-shooting approach, resulting in the discretization of the state along the path ($\boldsymbol{x}(t) = [s(t), v(t)]$) and force input $u(t)$ over a finite grid of N samples. The model representing the wheelchair

dynamics ($\boldsymbol{f}$) is then integrated with an integrator function ($\boldsymbol{I}$). For this solution, a simple mass-damper model was used. The time (T) needed to arrive at the end of the path ($s(end)$) is kept as a decision variable, therefore an extra variable is defined, the variable time step $h = T/N$. Although not explicitly formulated in the COP, there is one time-varying hyperplane for every moving obstacle, each requiring two parameters $\boldsymbol{a}(t)$ and $b(t)$. $\boldsymbol{v}_i$ are the fixed vertices (positions) defining the simplified convex hull of the (s,t)-space. r_{safe} is a safety factor, enforcing a minimum distance in the s,t-plane between obtained path and the simplified convex hull. Start conditions are enforced on both start distance and speed, but only an end distance is set as a constraint for the end state. The followed approach is described in detail in (Mercy, Loock, and Pipeleers 2016).

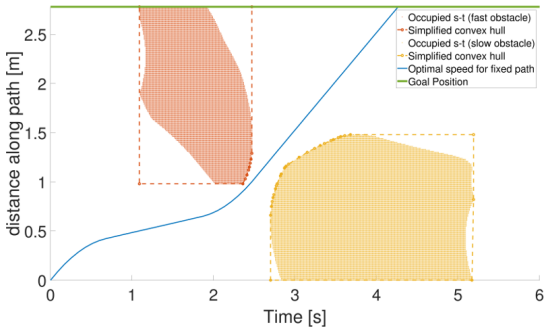

Figure 11: Motion planning among dynamic obstacles is achieved by finding a path in the s,t-plane without intersecting the occupied cells. This results in an optimal collision-free speed profile that also adheres to the robot's dynamic constraints.

Evaluation

This section evaluates the performance of the clothoidal LPT by comparing it with its predecessor, the circular LPT. Two different path planning scenarios will be presented (door and elevator), requiring the planner to plan a trajectory through a narrow opening. These scenarios were chosen because these are often encountered by wheelchair drivers, and we experienced difficulties with these scenarios in the past when using circular paths. In order to objectively compare both LPTs, identical curvature constraints on the path are applied to the circular LPT and the clothoidal LPT. For the circular LPT, paths are generated starting from 500 discrete input velocity pairs (v,ω) but paths yielding a curvature $\kappa > \kappa_{max}$ are discarded. This results in a circular LPT composed of 250 trajectories (both forward and backward), shown in Figure 12. The clothoidal LPT used for this evaluation is the same as shown in Figure 5 and is composed of 1500 trajectories (forward and backward).

Path Planning Performance

The first benchmark consists of driving forward through a doorway; for the second benchmark, the wheelchair has to drive backwards into an elevator.

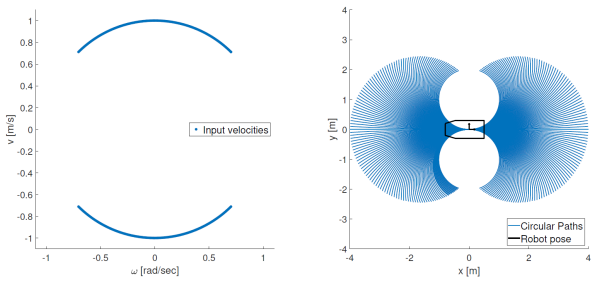

Figure 12: Input velocities (left) integrated over a period of $t = 4\,s$ to obtain 250 circular paths complying with the kinematic constraints (right).

Forward driving through a doorway In this situation, the mobile robot has to drive through a doorway to exit a room. Figure 13 (top) shows a successful, collision-free path going through the doorway using both LPTs.

A uniform set of start poses is generated to assess the planning performance of both LPTs. This is shown in Figure 13 (bottom, left), where the test region (red polygon) contains the set of uniformly spaced start poses. If a path of the LPT originating from a start pose reaches the goal region (green polygon), that pose is defined as successful. Successful start poses are divided in three cases: (1) only the circular LPT achieved to plan a path reaching the goal area, (2) both LPTs achieved to plan a path, (3) only the clothoidal LPT achieved to plan a path.

The final outcome when following this procedure is shown in Figure 13 along with a histogram of the occurrence of the three different cases. The unique successful start poses based on circular, common and clothoidal LPTs are shown respectively in red, black and green. The width of the doorway from the corridor to the robot laboratory is 80 cm while the width of the wheelchair is 60 cm.

The following conclusions can be drawn:

- The majority of the paths from start poses at the lower end of the figure are achieved by the circular LPT. This is because those paths only require a circular arc to enter the doorway. Since the circular LPT is composed of a uniformly spread set of circular trajectories, this LPT is favoured, compared to the clothoidal LPT.
- From the moment the required trajectory is more complex (for example, Figure 13 (top, right) the clothoidal LPT is the only LPT able to plan a path.
- As per the histogram provided in Figure 13, the majority (87%) of the poses with a successful path are generated by the clothoidal LPT, which demonstrates that for this first benchmark, the lack of uniformly spread circular trajectories is not so crucial, as only 13% of the total amount of successful start poses are uniquely found by using the circular LPT.

Backwards driving in an elevator In this situation, the mobile robot must drive backwards from a corridor into an elevator. The width of the elevator is 90 cm while the width

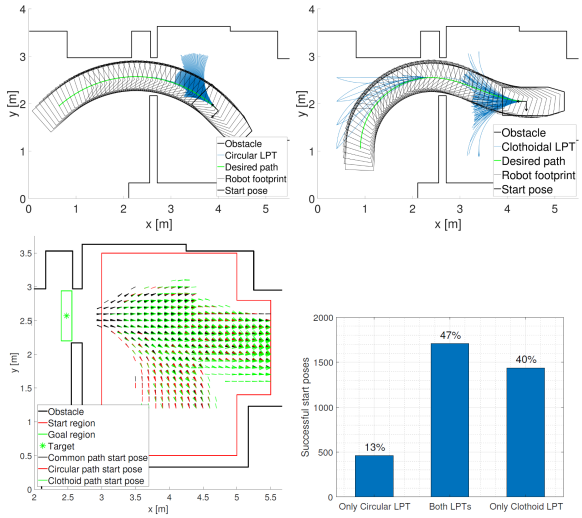

Figure 13: (top) Visual inspection of a collision-free path going through the doorway. (bottom, left) Successful start poses for each LPT finding a path through the doorway. (bottom, right) Histogram showing the outcome of the different cases of driving through the doorway. There are in total 3604 successful start poses, of which 1708 are common to both LPTs, whereas 460 are only of the circular LPT and 1436 of the clothoidal LPT.

of the wheelchair is 60 cm. Figure 14 (top) shows a successful, collision-free path going in reverse through the doorway of the elevator whilst using both LPTs. The same procedure as provided in Figure 13 is applied for this benchmark. It can be concluded that from the moment the start pose of the wheelchair is located farther away in the corridor, only the clothoidal LPT manages to find a trajectory to the elevator. This is confirmed by the histogram shown in Figure 14 (bottom, right). Nearly all start poses are successful when using the clothoidal LPT (98%).

Time Performance

The clothoidal LPT has six times more paths than the circular LPT. In order to verify that this does not come at a too high computation cost, the mobile robot is positioned at several places in the environment as shown in Figure 14 (bottom, left). The presented benchmarks were run on a computer with an Intel® Core^{TM}i5-4460 quad core 3.20GHz CPU with 16GB of memory. Benchmark scripts were written in MATLAB. Figure 15 (left) shows a histogram of the execution time needed to adapt each path from the LPTs. The median execution time of the circular LPT is 29 ms compared to 114 ms for the clothoidal LPT; the clothoidal LPT is therefore 3.9 times slower. Figure 15 (right) shows the execution time of the path length adjustment over the number of occupied cells.

Conclusions and Future Work

This paper proposed various improvements to a circular local path template (LPT) that we have designed in the past

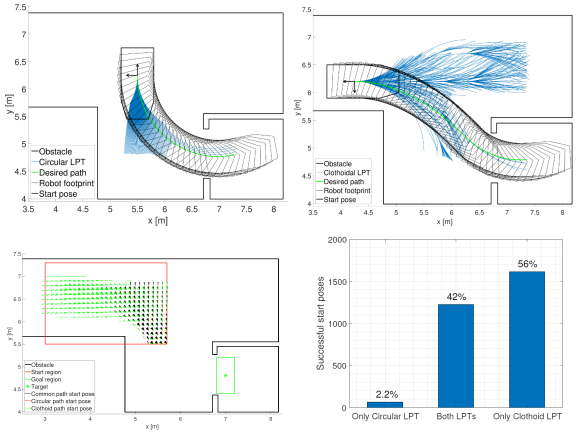

Figure 14: (top) Visual inspection of a collision-free path for backwards driving through an elevator. (bottom, left) Successful start poses for each LPT planning a path backwards into an elevator. (b) Histogram illustrating the outcome of the different cases for planning a path backwards into an elevator. There are in total 2904 successful start poses, from which 1224 are common to both LPTs. Whereas 64 are only from the circular LPT and 1616 from the clothoidal LPT.

to estimate driver intents and to provide navigation assistance in wheelchair applications. Using two performance criteria, path planning capability and time performance, the clothoidal LPT was compared with the existing circular LPT. The clothoidal LPT allows the use and tuning of more complex paths, resulting in an improved path planning performance; however, it has a higher computation time. In sparse environments, the circular LPT will prove more efficient. One could choose the LPT type depending on the context the robot is in, or implement a switching between both LPTs if one LPT fails in finding a path. In our intent estimation framework, we plan to only use the clothoidal LPT as the impact on computation time is minor. Future work will

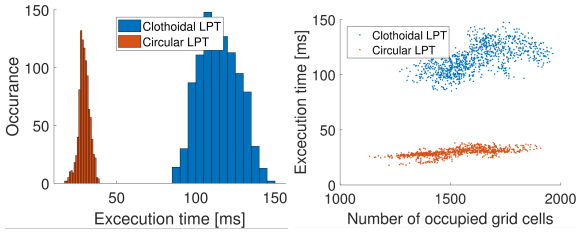

Figure 15: Execution time needed to adapt the path length of the circular and clothoidal LPT for a simulated environment. (left) Histogram of the resulting execution time. (right) Computation time as a function of the number of occupied cells.

include an implementation in C/C++ and an optimization of the position of the EPs to reduce the number of clothoidal paths. Furthermore, we will investigate the use of this LPT for fast collision checking in path planning algorithms.

References

Baass, K. G. 1982. Use Of Of Clothoid Templates In Highway Design. *Transportation Association of Canada*.

Bascetta, L.; Ferretti, G.; Rocco, P.; Ardö, H.; Demeester, E.; Bruyninckx, H.; and Di Lello, E. 2011. Towards safe human-robot interaction in robotic cells: an approach based on visual tracking and intention estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2971–2978.

Bennewitz, M.; Burgard, W.; Cielniak, G.; and Thrun, S. 2005. Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research* 24(1):31–48.

Bertolazzi, E., and Frego, M. 2015. G1 fitting with clothoids. *Mathematical Methods in the Applied Sciences* 38(5):881–897.

Brezak, M., and Petrović, I. 2011. Path Smoothing Using Clothoids for Differential Drive Mobile Robots. *IFAC Proceedings Volumes* 44(1):1133–1138.

Cope, G. H. 1993. *British Railway Track: Design, Construction and Maintenance*. Permanent Way Institution Loughborough,, UK.

Demeester, E., and Hüntemann, A. 2016. Detecting user intention changes using the Kullback-Leibler distance. In *AAAI Fall Symposium Series*, 318–324.

Demeester, E.; Vander Poorten, E.; Philips, J.; and Huntemann, A. 2012. Design and evaluation of a lookup-table based collision-checking approach for fixed sets of mobile robot paths. In *International Symposium on Robotics*, 1045–1050.

Demeester, E.; Hüntemann, A.; Vander Poorten, E.; and De Schutter, J. 2014. ML, MAP and greedy POMDP shared control: Qualitative comparison of wheelchair navigation assistance for switch interfaces. *Zhongguo Jixie Gongcheng Xuehui (Journal of Chinese Society of Mechanical Engineers)* 35(4):333–342.

Fleury, S.; Soueres, P.; Laumond, J. P.; and Chatila, R. 1995. Primitives for smoothing mobile robot trajectories. *IEEE Transactions on Robotics and Automation* 11(3):441–448.

Hüntemann, A.; Demeester, E.; Nuttin, M.; and Van Brussel, H. 2008. Online user modeling with gaussian processes for bayesian plan recognition during power-wheelchair steering. In *Proceedings of the IEEE/RSJ international conference on Intelligent Robots and Systems (IROS)*, 285–292.

Hüntemann, A.; Demeester, E.; Vander Poorten, E.; Van Brussel, H.; and De Schutter, J. 2013. Probabilistic approach to recognize local navigation plans by fusing past driving information with a personalized user model. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 4376–4383.

Kelly, A., and Nagy, B. 2003. Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control. *The International Journal of Robotics Research* 22(7-8):583–601.

McCrae, J., and Singh, K. 2009. Sketching piecewise clothoid curves. *Computers & Graphics* 33(4):452–461.

Mercy, T.; Loock, W. V.; and Pipeleers, G. 2016. Real-time motion planning in the presence of moving obstacles. In *2016 European Control Conference (ECC)*, 1586–1591.

Pivtoraiko, M., and Kelly, A. 2012. Generating State Lattice Motion Primitives for Differentially Constrained Motion Planning. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 101–108.

Scheuer, A., and Fraichard, T. 1997. Continuous-curvature path planning for car-like vehicles. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 997–1003.

Vander Poorten, E.; Demeester, E.; Reekmans, E.; Philips, J.; Hüntemann, A.; and Schutter, J. D. 2012. Powered wheelchair navigation assistance through kinematically correct environmental haptic feedback. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, 3706–3712.

Walton, D. J., and Meek, D. S. 2009. G1 interpolation with a single Cornu spiral segment. *Journal of Computational and Applied Mathematics* 223(1):86–96.