Path Smoothing Using Clothoids for Differential Drive Mobile Robots ⋆

Mišel Brezak * **Ivan Petrović** **

* *University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia (e-mail: misel.brezak@fer.hr).*
** *University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia (e-mail: ivan.petrovic@fer.hr).*

Abstract:
This paper presents a new online path-smoothing algorithm for smoothing a path that consists of straight line segments and is primarily intended for differential drive mobile robots. Algorithm allows non-zero initial path curvature, which is important for replanning. Further, algorithm uses clothoid curves as primitives for path smoothing, which have inherent property that their curvature changes proportionally with distance traveled along the curve.

Keywords: Mobile robots, Path planning, Path smoothing, Clothoid, Navigation

1. INTRODUCTION

Common path-planning methods usually generate obstacle-free path, but with no or very little concern about path feasibility or optimality. It is usually necessary to apply some kind of transform algorithm to locally modify such a path. There are various motivations to do this, e.g. the planned path may not be feasible for the particular robot because the path planner does not consider kinematic or other constraints of the robot, or the initial path may contain sharp turns that the robot cannot follow.

It is assumed that global obstacle-free path is already planned by some method that produces piecewise linear path. Also, if obstacles have to be accounted in smoothing process, for each linear segment amount of clearance to nearest obstacle should be known. Application of differential drive mobile robot is assumed, which has no curvature limits, so that it is sufficient to transform a path using an appropriate smoothing algorithm that filters out the curvature steps. We refer to such algorithms as *path-smoothing* algorithms.

Various path-smoothing algorithms are proposed in the literature. E.g. Kanayama and Hartman (1997) use cubic splines, Delingette et al. (1991) use intrinsic splines, while Segovia et al. (1991)) propose Bezier's curves. It is however difficult to control curvature of these curves. This problem is avoided with clothoids, whose advantage is linearly changing curvature which makes them very attractive in path-smoothing applications.

So Kanayama and Miyake (1985) join initial and final configurations, which are supposed to have zero curvature, by sequences of clothoid pairs and straight lines. Shin and Singh (1990) produce a smooth path by connecting each pair of neighboring postures with three clothoids. In addition to clothoids Fleury et al. (1995) also introduced curves called anticlothoids, which enable changes of motion direction along the trajectory. The clothoids and anticlothoids are used because they represent the time-optimal trajectories of a two driving wheels mobile robot. Scheuer and Fraichard (1997) use a set of paths, called bi-elementary paths, which are smooth and have upper-bounded curvature. These paths are composed of two equal piecewise clothoids.

However, as clothoids are transcendental curves defined in terms of Fresnel integrals, they cannot be solved analytically which makes them difficult to use in online planning. Montés et al. (2008) solve this by approximating clothoid offline with rational Bezier curve and later use this approximation online.

This paper proposes a new path-smoothing algorithm using clothoids with the following properties:

i) G^2 continuity of the smoothed path;
ii) smoothing with non-zero initial curvature;
iii) low computational complexity.

The first property ensures continuous path curvature, allowing the robot to traverse the smoothed path without stopping. The second property is important if the path is to be replanned while robot is moving. Namely, if current robot's motion has non-zero curvature, which is often true in praxis, the smoothing method must be capable of taking this initial condition into account to obtain a feasible path. While most of the existing methods fulfill the requirement i), neither method fulfills both requirement ii) and requirement iii). In other words, the existing methods are whether not specifically designed to work online, or do not allow specifying of non-zero initial curvature.

This paper is organized as follows. In Section 2 some properties of clothoid and evaluation of its coordinates are discussed. In Sections 3 and 4 path-smoothing algorithms are described. In Section 5 some results are presented.

⋆ This research was supported by the Ministry of Science, Education and Sports of the Republic of Croatia (grant No. 036-0363078-3018).

2. CLOTHOID EVALUATION

Initially we give important clothoid equations. The curvature $\kappa(s)$ of the clothoid is

$$\kappa(s) = \kappa_0 + cs, \tag{1}$$

where κ_0 is initial path curvature, c is parameter called sharpness and s is parameter that denotes traveled distance along the clothoid. Instead of clothoid sharpness, *scaling factor* C is sometimes used, where $C^2 = 1/c$. The tangent angle $\theta(s)$ of the clothoid is

$$\theta(s) = \theta_0 + \kappa_0 s + \frac{1}{2}cs^2, \tag{2}$$

where θ_0 is initial heading. Parametric expressions for clothoid coordinates are

$$x(s) = \mathrm{Clx}(s) = x_0 + \int_0^s \cos(\theta_0 + \kappa_0\xi + \frac{1}{2}c\xi^2)d\xi, \tag{3a}$$

$$y(s) = \mathrm{Cly}(s) = y_0 + \int_0^s \sin(\theta_0 + \kappa_0\xi + \frac{1}{2}c\xi^2)d\xi. \tag{3b}$$

In majority of literature it is reported that evaluation of (3) is computationally expensive and therefore clothoids is hard to use online. However, in this work this is solved by storing points of a single clothoid in the lookup table. Then points of any clothoid can be efficiently computed based on the stored clothoid by rescaling, rotating and translating (Brezak (2010)). In this way the clothoid evaluation is typically executed in time order of microsecond, so that very fast smoothing is achieved that can be used in online planning. Moreover, fast clothoid evaluation also enables execution of algorithms such as clothoid-line intersection test. This also enables fast collision checks of clothoids, which are required by some algorithms in continuation.

3. STATIC SMOOTHING

The simplest case of smoothing occurs when initial and final curvatures are zero. This case is called static smoothing. As initial and final path segments are straight lines, the problem is how to compute parameters of two clothoids that ensure smooth transition between two lines.

An example is shown in Fig. 1, where the turn is given with points S, T and G in the (x, y) plane, which denote the start position, the location of the sharp turn, and the goal position, respectively. Without loss of generalization, it is assumed that T is at the origin, S is on the negative x-axis, while G is in $y > 0$ half-plane. Any general case can be transformed to the described one. Algorithm outputs are parameters of the two clothoids that build a final smoothed path. The procedure is called SmoothSharpTurn and its pseudocode is listed in Alg. 1.

To enable collision avoidance, the deviation of the smoothed path from the original path can be controlled by means of parameter e_{max} that denotes the maximum allowed distance between the clothoid and point T, denoted as e in Fig. 1.

The smoothed path contains a pair of two symmetrical clothoids, the first from point C_1 to C_2, and the second from C_2 to C_3. Because of symmetry, the amount of

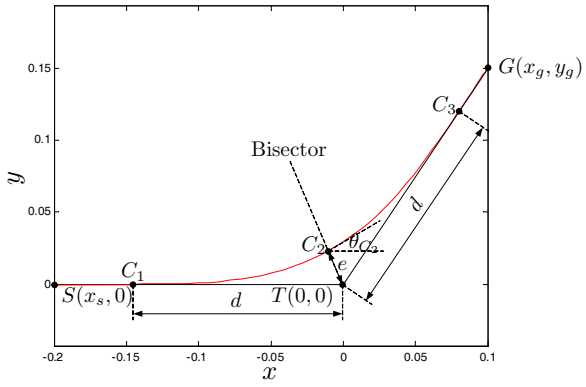


Fig. 1. Smoothing a sharp turn with clothoid pair.

Algorithm 1. SmoothSharpTurn

Input: $\mathrm{x}_s, x_g, y_g, e_{max}, d_{max}$
Output: $x_{0C_1}, y_{0C_1}, \theta_{0C_1}, \kappa_{0C_1}, c_{C_1}, s_{C_1}$: first clothoid
$x_{0C_2}, y_{0C_2}, \theta_{0C_2}, \kappa_{0C_2}, c_{C_2}, s_{C_2}$: second clothoid

$d_{max} = \min(|\mathrm{ST}|, |\mathrm{TG}|, d_{max})$;
$\theta_{C_2} = \arctan(y_g/x_g)/2$;
$s_{BC} = \sqrt{2 \cdot \theta_{C_2}}$;
$(x_{BCs}, y_{BCs}, x_{BCt}, y_{BCt}, d_{BC}, e_{BC}) =$
  EvaluateBasicClothoid(s_{BC}, θ_{C_2});
$ScaleRatio = \max(d_{BC}/d_{max}, e_{BC}/e_{max})$;
$s = s_{BC}/ScaleRatio$;
$x_{C_2} = x_{BCt}/ScaleRatio$; $\quad y_{C_2} = y_{BCt}/ScaleRatio$;
$d = d_{BC}/ScaleRatio$;
$e = e_{BC}/ScaleRatio$;
$x_{0C_1} = -d$; $y_{0C_1} = 0$;
$\theta_{0C_1} = 0$; $\kappa_{0C_1} = 0$;
$c_{C_1} = ScaleRatio^2$;
$s_{C_1} = s$;
$x_{0C_2} = x_{C_2}$; $y_{0C_2} = y_{C_2}$;
$\theta_{0C_2} = \theta_{C_2}$; $\kappa_{0C_2} = s_{C_1} c_{C_1}$;
$c_{C_2} = -ScaleRatio^2$;
$s_{C_2} = s$;

orientation change is equal for both clothoids. Using this fact the tangent angle at the transition point θ_{C_2} is determined, which enables to efficiently solve the problem by first evaluating the clothoid with scaling $C = 1$ stored in lookup table, in function EvaluateBasicClothoid. The obtained path is then scaled back to the original problem.

Besides these two clothoids, the final path can also consist of two line segments, SC_1 and C_3G. Finally, parameters of both clothoids are computed. The second clothoid is obtained by reflecting the first one with respect to the bisector. Tangent angle and curvature continuity between the two clothoids (i.e. in the point C_2) follow by symmetry.

4. DYNAMIC SMOOTHING

In dynamic environments we often encounter situations when current plan becomes invalid due to changes in the environment etc. When replanning is performed, initial linear and angular velocities cannot be assumed to be zero because the robot was already moving while executing the previous plan. The smoothing in this case is called *dynamic smoothing*, and is considerably harder than static

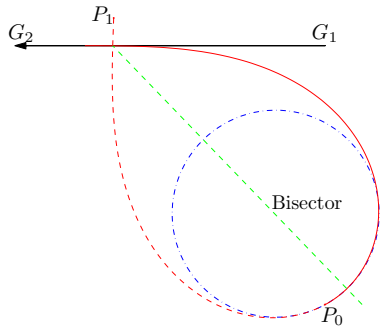


Fig. 2. Connecting circle and line using two clothoids.

smoothing. If initial path curvature is non-zero the problem translates to task of connecting circle and line in order to obtain a G^2 continuous path. In this work, at the goal we always want zero-curvature path, so that we do not study circle-circle connection. In algorithms that follow we solve the problem using various combinations of clothoids, circular arcs and straight lines.

4.1 Connecting Circle and Line by Two Clothoids

An example of dynamic smoothing using two symmetric clothoids is shown in Fig. 2. The corresponding algorithm is called ConnectCircleLine2C (Alg. 2). The input to the algorithm is an initial state $P_0(x_{P_0}, y_{P_0})$, tangent direction θ_{P_0} and curvature $\kappa_{P_0} \neq 0$. The goal line is line G_1G_2, where motion direction is from G_1 to G_2. Output of the algorithm are parameters of two clothoids that are solution to the problem.

In function CheckNecessaryConditions the algorithm performs fast heuristics in order to check whether it is worth to try to solve the problem. The solution may exist even if the necessary conditions are not fulfilled, but those are solutions that typically result in extremely long paths or extremely sharp turns. The first necessary condition is that the goal point is on the same side of the robot where the center of the circle of curvature is located. The second condition is that start point P_0 is to the left of vector G_1G_2 if the initial curvature is positive, and right otherwise.

To compute parameters of both clothoids, it is sufficient to find required scaling C of the first (entering) clothoid—it is then straightforward to determine remaining parameters. A single equation of the form $f(C) = 0$ has to be solved in order to obtain a solution. Function f is nonlinear so that numerical root-finding method is utilized. Those methods require that a root is bracketed in some interval (C_1, C_2) so that $f(C_1)$ and $f(C_2)$ have opposite signs. This is done in function BracketSolution of the algorithm.

The lower bracket C_1 is determined so that for a while we assume that initial curvature κ_{P_0} is zero. Then the problem is reduced to smoothing the sharp turn from Sec. 3. If we additionally constrain that entering clothoid must strictly begin in point P_0, then correspondent Alg. 1 gives us the lower bracket C_1. Here we use the fact that if the initial curvature is zero, the required sharpness of the clothoid is always higher, i.e. scaling factor is lower, compared to the case of non-zero curvature.

Algorithm 2. ConnectCircleLine2C

Input: P_0 : initial state with parameters (x, y, θ, κ)
G_1, G_2 : begin and end points of goal line
Output: CL_1, CL_2 : parameters of two clothoids
Success flag (**true** or **false**)

if fail CheckNecessaryConditions(P_0, G_1, G_2)
return (**false**);
if fail $(C_1, C_2) =$ BracketSolution(P_0, G_1, G_2)
return (**false**);
if fail $(C) =$ FindRoot$(P_0, G_1, G_2, C_1, C_2)$
return (**false**);
if fail CheckAdmissibility(P_0, G_1, G_2, C)
return (**false**);
$(CL_1, CL_2) =$ ComputeClothoidParms(P_0, G_1, G_2, C)
return (**true**);

Upper bracket C_2 is obtained by limiting total orientation change of the turn to less than 2π. Let's suppose a positive-direction turn. Total orientation change is then $\Delta\theta = \theta_{G_1G_2} - \theta_{P_0}$, where $\theta_{G_1G_2}$ is direction of vector G_1G_2. As this orientation change is divided to both entering and exiting clothoid, the maximum absolute orientation change of any of these clothoids is less than π. Let point P_1 in Fig. 2 denote a virtual begin point of the entering clothoid where its curvature is zero. The maximum orientation change of the portion of entering clothoid from P_1 to P_0 is

$$\Delta\theta_{P_1P_0max} = \min(2\pi - \Delta\theta, \pi). \tag{4}$$

Using (1), we obtain the upper bracket as

$$C_2 = \frac{\sqrt{2\Delta\theta_{P_1P_0max}}}{\kappa_{P_0}}. \tag{5}$$

It remains to define an appropriate function $f(C)$ whose root gives a solution of our problem. As a preliminary, let's define a bisector line as an axis of symmetry between vector opposite to tangent vector in P_1 and vector G_1G_2 (see Fig. 2). Now we want to find a value C for which the entering clothoid intersects the bisector so that in the point of intersection clothoid and bisector are perpendicular to each other. If such C exists, then the problem has a solution and the function $f(C)$ has a root in this value. The function $f(C)$ that has such a property is given in the Alg. 3 and illustrated in Fig. 3. Here a point on the clothoid is searched where the tangent is normal to the bisector line, which is denoted with point L in Fig. 3. Then the value of the function is defined as the signed distance between point L and bisector (denoted as f in Fig. 3). The problem with this definition is that distance f can become infinite, but this is handled as a special case in Alg. 3

Function FindRoot in Alg. 2 finds solution to the equation $f(C) = 0$ using bisection method. The solution of $f(C)$ is always unique because the entering clothoid is constrained to pass exactly through the initial position P_0. If the function FindRoot reports success, the solution C to the equation $f(C) = 0$ is found. However, the solution may still not be admissible, so that it is additionally checked in function CheckAdmissibility in Alg. 2 (ConnectCircleLine2C). Here the conditions are checked such as that point L in Fig. 3 must lie after point P_0 on the path. Additionally, the produced path may not be collision-free so that it is tested using collision

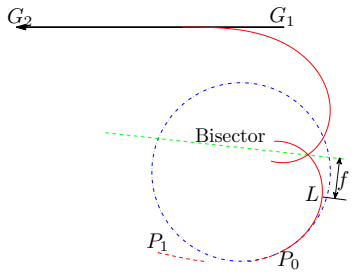


Fig. 3. An iteration of Alg. 2.

Algorithm 3. CRITERIONFUNCTION

Input: C, P_0, G_1, G_2
Output: f

$P_1 = \text{COMPUTEP1}(C, P_0)$;
$P_1TgOpp = \text{LINEDIRPOINT}(P_1.\theta + \pi, P_1.x, P_1.y)$;
$G_1G_2 = \text{LINEFROMPOINTS}(G_1.x, G_1.y, G_2.x, G_2.y)$;
if (ISPARALLELANDOPPOS(P_1TgOpp, G_1G_2))
  $distG_1G_2toP_1 = \text{DISTTOPT}(G_1G_2, P_1.x, P_1.y)$;
  if (ISZERO($distG_1G_2toP_1$)
    $f = 0$;
  else if ($distG_1G_2toP_1 * P_0.k < 0$)
    $f = \infty$;
  else
    $f = -\infty$;
else
  $Bisect = \text{SYMMETRICLINE}(P_1TangentOpp, G_1G_2)$;
  $Cloth.x_0 = P_1.x$; $Cloth.y_0 = P_1.y$;
  $Cloth.th_0 = P_1.th$;
  $Cloth.k_0 = 0$;
  $Cloth.C = C$;
  $s = \text{CLOTHPTWITHANGLE}(Cloth, Bisect.\theta - \pi/2)$;
  $(x_{lat}, y_{lat}) = \text{CLOTHOIDPT}(Cloth, s)$;
  $distToBisect = \text{DISTTOPT}(Bisect, x_{lat}, y_{lat})$;
  $f = -\operatorname{sgn}(C) * distToBisect$;

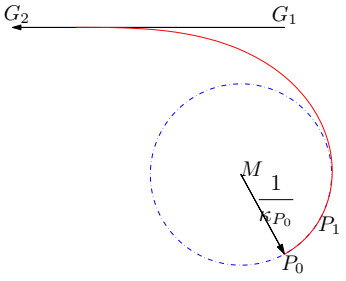


Fig. 4. Connecting circle and line using a single clothoid.

detection module. If the solution C is admissible, it is straightforward to compute remaining parameters of both required clothoids. This is done in function COMPUTECLOTHOIDPARMS in Alg. 2.

4.2 Connecting Circle and Line by a Single Clothoid

In some circumstances it is possible to connect circle and line by a single clothoid. An example is shown in Fig. 4. The point P_0 is initial position, so that the circle

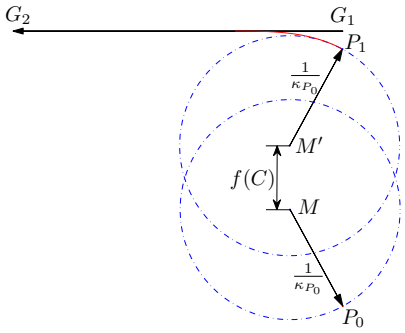


Fig. 5. An iteration of algorithm CONNECTCIRCLELINE1C.

is determined with coordinates and tangent direction in point P_0, while its radius r is obtained as inverse of the initial curvature, i.e. $r = 1/|\kappa_{P_0}|$. Line G_1G_2 is the goal line. In Fig. 4, the clothoid begins in point P_1 on the circle and ends on the line G_1G_2. An additional circular arc is used in order to obtain G^2 continuity, which is inserted between points P_0 and P_1.

Again, a nonlinear equation $f(C) = 0$ has to be solved, whose root C represents, if one exists, the scaling of the required clothoid. In this case function $f(C)$ is defined as

$$f(C) = d(G_1G_2, M) - d(G_1G_2, M')$$

where $d(G_1G_2, M)$ and $d(G_1G_2, M')$ are signed Euclidean distances between vector G_1G_2 and points M and M', respectively (see Fig. 5). Points M and M' are centers of path curvature in points P_0 and P_1, respectively. The coordinates of point M are

$$x_M = x_{P_0} - \frac{1}{\kappa_{P_0}} \sin\theta_{P_0}, \quad y_M = y_{P_0} + \frac{1}{\kappa_{P_0}} \cos\theta_{P_0}. \quad (6)$$

The point M' is center of curvature of the clothoid in the point P_1 where it has the curvature equal to κ_{P_0}. The parameter s of the clothoid in point P_1 is therefore

$$s_{P_1} = |\kappa_{P_0}| C^2. \quad (7)$$

The coordinates of point P_1 are now obtained as

$$x_{P_1} = \text{Clx}(s_{P_1}), \quad y_{P_1} = \text{Cly}(s_{P_1}), \quad (8)$$

where the coordinates of clothoid $(\text{Clx}(s), \text{Cly}(s))$ are defined with (3), while its parameters are $(x_0, y_0, \theta_0, \kappa_0) = (x_{G_1}, y_{G_1}, \theta_{G_2G_1}, 0)$. The path tangent direction θ_{P_1} in point P_1 is opposite to clothoid tangent direction in P_1, so that

$$\theta_{P_1} = \theta_{G_2G_1} + \operatorname{sgn}(C) \frac{s_{P_1}^2}{2C^2} + \pi. \quad (9)$$

Finally, the coordinates of point M' are obtained as

$$x_{M'} = x_{P_1} - \frac{1}{\kappa_{P_0}} \sin\theta_{P_1}, \quad y_{M'} = y_{P_1} + \frac{1}{\kappa_{P_0}} \cos\theta_{P_1}. \quad (10)$$

The algorithm that solves the problem of connecting circle and line by a single clothoid is called CONNECTCIRCLELINE1C, and its structure is very similar to Alg. 2 (CONNECTCIRCLELINE2C), except that the implementation details of the functions are different, so that pseudocode is not given here. Implementation of the function BRACKETSOLUTION is also similar. Function CHECKADMISSIBILITY here checks if the point P_1 is before point P_0

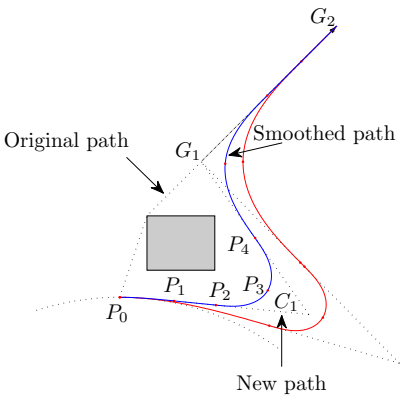


Fig. 6. Example of two paths constructed by ConnectCircleLine3C algorithm.

on the path. Also, it is tested if the path is collision-free. If the path is not admissible, it is rejected.

4.3 Connecting Circle and Line by Three Clothoids

Algorithms ConnectCircleLine2C and ConnectCircleLine1C give a quick solution to the problem of connecting circle and line path segments in most cases. However there are configurations for which those algorithms may not have a solution (e.g. if robot is currently steering right, but the goal is to the left), solution may not be admissible (e.g. because of obstacles) or may not be feasible (e.g. because of actuator limits). This is illustrated in Fig. 6, where the initial configuration is in point P_0. Original path connects initial configuration to the node G_1 of the path. As current configuration in point P_0 has high curvature to the right, i.e. opposite of the goal line G_1G_2, both algorithms ConnectCircleLine1C and ConnectCircleLine2C fail to produce an acceptable path.

In such situations three clothoids can be used to obtain more flexibility. This may be obtained by first forcing the path to go straight. This transition from circle to line can be obtained using a single clothoid called *transition clothoid*. An example is displayed in Fig. 6, where the transition clothoid goes from initial point P_0 to point P_1.

After the path is made straight, the new path is obtained by extending the straight line segment to point C_1 and dynamically inserting C_1 as a new node into the path. In Fig. 6 this results with the new path (P_1-C_1-G_1). The new path segment consists of lines so that it is easily smoothed using algorithm SmoothSharpTurn, resulting with line P_1P_2, and two clothoids from P_2 to P_3 and from P_3 to P_4.

Altogether three clothoids were used to connect initial configuration in the point P_0 with the path point G_1, so that this is equivalent to connecting circle and line by three clothoids. This method gives more flexibility, because in this way any circle can be connected to the path. However, the problem has an infinite set of possible solutions. Some heuristics and trial and error methods are utilized to solve the problem in real time. The smoothing algorithm alone is simpler in this case comparing to algorithms ConnectCircleLine2C and ConnectCircleLine1C because there is no need to solve any nonlinear equation.

Algorithm 4. ConnectCircleLine3C

Input: CurrentState, Path
Output: SuccessFlag, SmoothedPath

c_{max} = GetMaxSharpness($CurrentState$);
for 1 **to** $MaxIterations$
  C = PickScaling(c_{max});
  $ClothAndLine$ = StraightPath($CurrentState, C$);
  if fail CheckAdmissibility($ClothAndLine$)
   continue ;
  $NewPath$ = Update($Path, ClothAndLine$);
  $SmoothedPath$ = SmoothPath($NewPath$);
  if fail CheckFeasibility($SmoothedPath$);
   continue ;
  else
   return (**true**);
return (**false**);

However, the whole problem is more complex and overall computation time is typically higher compared to previous algorithms, mainly due to the fact that a new node has to be inserted into the path in every iteration. The correspondent algorithm is called ConnectCircleLine3C (Alg. 4).

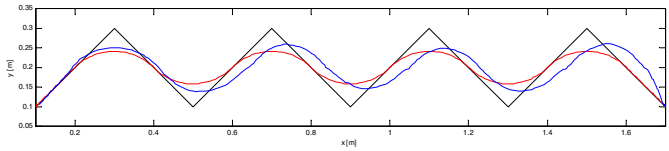
The algorithm first determines maximum allowed clothoid sharpness that can be used for transition to the straight line. This requires knowledge about robot dynamics and actuator limits and is done in function GetMaxSharpness. In each iteration of the **for** loop in the algorithm, a new scaling factor from allowed range is picked based on heuristics. The allowed range of scalings is determined based on maximum sharpness and maximum allowed length of the transition clothoid.

For the current scaling the transition clothoid is computed in function StraigthPath. This function also returns a line where the clothoid lands. The obtained clothoid and line are checked for admissibility, i.e. collision check is performed. If there is no collision, the endpoint of the line is inserted as a new path node, and the path is replanned. This path is then smoothed in function SmoothPath which internally calls SmoothSharpTurn algorithm.

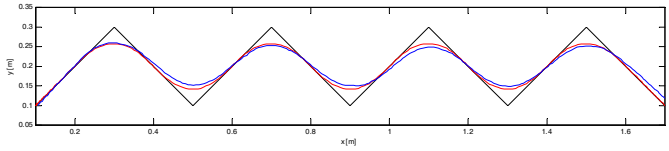
Optionally, the smoothed path can be checked for feasibility, which cannot be done by the path-smoothing module, but is directed to separate module that accounts for robot dynamics. Here it is tested if the path can actually be tracked by the robot regarding its actuator limits and other dynamic constraints. If the path is feasible, the algorithm terminates and reports success. Otherwise, the next iteration is tried, until the maximum number of iterations is reached. If this happens, the failure is reported. Note that no optimality criterion is examined in the described algorithm, but this could easily be added so that the shortest, or even the fastest path can be searched for.

An example of two different paths produced by algorithm ConnectCircleLine3C are given in Fig. 6. It can be seen that algorithm successfully finds an admissible path. For illustration, another path produced by the same algorithm is shown with lower transition clothoid sharpness.

In some situations changes in the environment happen to be so drastic that none of the described replanning



(a)



(b)

Fig. 7. Path tracking experiment. Legend: black – original path; red – smoothed path; blue – actual robot path. *(a)* Path smoothed with circular arcs. *(b)* Path smoothed with clothoids.

algorithms can find acceptable solution—this is most likely to be caused by fast moving obstacles or change of goal configuration. Then the last resort is to brake as fast as possible and possibly to pick a steering direction that is clearest of obstacles—this is called *emergency stop* maneuver. Once the robot has stopped, path smoothing is always easier because the feasible trajectory along the smoothed path can always be found if one exists. This is enabled by the fact that no excessive space is needed for robot to continue its prior motion and robot can simply reorientate itself in path direction (here we do not impose any constraints on maximum path curvature) and then normally execute the rest of the path.

5. EXPERIMENTAL RESULTS

Path smoothing by proposed algorithms is already illustrated in Figures 1, 2, 4, and 6.

To verify correctness of the clothoid steering model, some experiments on the real mobile robot were also performed. For this we used a small differential-drive mobile robot, which is RF-controlled and its position and heading are tracked by the camera mounted above the robot using computer vision algorithm (Brezak (2010)). To illustrate necessity of introducing G^2 continuous paths, the results were also compared to the case where only circular arcs are used for smoothing, which is only G^1 continuous.

A zig-zag shaped path was used, which is very demanded for the robot as it requires fast changes of steering direction. The trajectories were planned so that the robot tracks smoothed paths at a constant velocity of 0.6 m/s, except at begin and end segments, where uniform acceleration and deceleration were applied. For trajectory tracking a nonlinear trajectory-tracking controller (Brezak et al. (2009)) was used.

The case where the path is smoothed using circular arcs is shown in Fig. 7 (a). It is visible that the robot has a hard time tracking the trajectory, as the tracking error is rather high. On the contrary, tracking of path smoothed with clothoids resulted with significantly lower tracking error (Fig. 7 (b)).

Furthermore, with clothoid-smoothed path maximum trajectory-tracking velocity that could be achieved was up to 25 % higher compared to the case with circular arcs. Here the maximum trajectory-tracking velocity denotes the velocity at which the robot is still able to track the trajectory, i.e. the tracking error remains stable.

6. CONCLUSION

In this paper a path-smoothing algorithm capable of smoothing piecewise linear paths is presented. The produced path is G^2 continuous with linearly changing curvature which is achieved by using clothoid curves as smoothing primitives. Clothoids were used due to their very attractive properties, where the major one is linear relation between the curvature and the arc length. Moreover, efficient algorithms capable of smoothing a path with non-zero initial curvature are developed, which is essential e.g. when path replanning for moving robot is required. To the best author's knowledge, this is the first complete solution that enables application of clothoids for online path replanning. This feature is useful for motion planning in changing environments.

REFERENCES

Brezak, M. (2010). *Localization, Motion Planning and Control of Mobile Robots in Intelligent Spaces*. Ph.D. thesis, University of Zagreb, Zagreb, Croatia.

Brezak, M., Petrović, I., and Perić, N. (2009). Experimental comparison of trajectory tracking algorithms for nonholonomic mobile robots. In *Proc. of Annual Conference of the IEEE Industrial Electronics Society*.

Delingette, H., Herbert, M., and Ikeuchi, K. (1991). Trajectory generation with curvature constraint based on energy minimization. In *Proceedings of Int. Conference on Intelligent Robots and Systems*. Osaka, Japan.

Fleury, S., Soueres, P., Laumond, J.P., and Chatila, R. (1995). Primitives for smoothing mobile robot trajectories. *IEEE Trans. Robot. Autom.*, 11(3), 441–448.

Kanayama, Y. and Miyake, N. (1985). Trajectory generation for mobile robots. In *Proceedings of the International Symposium on Robotics Research*, 16–23.

Kanayama, Y.J. and Hartman, B.I. (1997). Smooth local planning for autonomous vehicles. *The International Journal of Robotics Research*, 16(3), 263–284.

Montés, N., Herraez, A., Armesto, L., Tornero, J., Herraez, A., Armesto, L., and Tornero, J. (2008). Real-time clothoid approximation by rational bezier curves. In *Proc. of International Conference on Robotics and Automation*.

Scheuer, A. and Fraichard, T. (1997). Collision-free and continuous-curvature path planning for car-like robots. In *Proceedings of International Conference on Robotics and Automation (ICRA '97)*. Albuquerque, U.S.A.

Segovia, A., Rombaut, M., Preciado, A., and Meizel, D. (1991). Comparative study of the different methods of path generation for a mobile robot in a free environment. In *International Conference on Advanced Robotics*.

Shin, D.H. and Singh, S. (1990). Path generation for robot vehicles using composite clothoid segments. Technical Report CMU-RI-TR-90-31, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA.