2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Trajectory Deformation Through Generalized Time Scaling

Arun Kumar Singh[1], Reza Ghabcheloo [1]

Abstract— **In this paper, we propose a new concept called Generalized Time Scaling (GTS) which as the name suggests is a generalization of the time scaling concept extensively used for the kino-dynamic motion planning of robots. Given a trajectory, a time scaling transformation modifies the motion profile of the trajectory while preserving the geometric path associated with it. In contrast, GTS transformation modifies both the motion profile as well as the geometric path of the given trajectory. We show that this feature of GTS can be leveraged to derive a new formulation for trajectory deformation which in turn has several key advantages over existing frameworks. Firstly, it directly works at the trajectory description level and does not require multiple reintegration of state space models. Secondly, it can can directly deform straight line and circular trajectories into complex shapes. Finally, the proposed formulation takes the form of an optimization problem which is interesting in itself and has potential beyond the proposed work. In particular, we use Augmented Lagrangian (AL) and Alternating Minimization (AM) to reformulate a complex non-convex problem into a sequence of distributed convex optimization problems. As an application, we use the trajectory deformation for the terminal heading correction, goal point correction, collision avoidance of non-holonomic mobile robots as well as for smoothing of trajectories consisting of piece-wise linear and circular segments.**

I. Introduction

Trajectory deformation is the problem of modifying an existing trajectory to satisfy some new set of constraints like changed terminal heading, goal point or avoiding obstacles. The motivation of trajectory deformation is clear: it allows to preserve as much as possible of the original trajectory which in turn would have come from some motion planner while optimizing some user defined metrics. Moreover, as we show later, trajectory deformation can implicitly minimize the deviation from the original trajectory both in terms of spatial and temporal aspects.

The problem of trajectory deformation has been extensively studied in the past couple of decades. The seminal concept of elastic bands was proposed in [1] which deforms a given trajectory based on artificial interaction forces similar to the concept of potential fields. In contrast, framework proposed in [2] follows the approach of iteratively perturbing the originally planned control input profile. Recent frameworks like [3], [4] have advocated the use of isometry groups for trajectory deformation. For example, [3] proposes the use of lie group symmetries while [4] is based on the concept of affine transformations.

In this paper, we present a novel formulation to deform both the temporal and spatial aspects of a given trajectory with the objective of improving its smoothness, to make it conform to some new terminal heading, goal point or collision avoidance constraints. At the heart of our approach is the following observation. Consider two time dependent functions $x_1(t), x_2(t), t \in [t_0 \ t_f]$ such that $X(t) = (x_1(t), x_2(t))$ represents the 2D trajectory of the robot. Then, both the smoothness and path of the trajectory can be modified by manipulating the derivatives of $x_1(t), x_2(t)$.

The first part of the observation is straightforward as trajectory smoothness can be modeled as continuity in the derivatives. For example, a continuous $\dot{x}_1(t), \dot{x}_2(t)$ will lead to a smoother trajectory as compared to that resulting when the derivatives are discontinuous. The second part of the observation stems from the fact that the path of the trajectory depends on the ratio between $\dot{x}_1(t)$ and $\dot{x}_2(t)$. For example, if $\dot{x}_1(t) = \dot{x}_2(t), \forall t$, then we obtain a trajectory with a straight line path.

The above discussion leads us to the concept of time scaling which is a convenient approach for modifying the derivatives of time dependent functions. [5], [6], [7]. However, conventional time scaling transformation would scale both $\dot{x}_1(t)$ and $\dot{x}_2(t)$ by the same scaling function. Consequently, its utility in improving trajectory smoothness is very limited. Further, for the same reason, the path associated with the trajectory remains unchanged under conventional time scaling transformations. In this paper, we present Generalized Time Scaling (GTS), which as the name suggests is a generalization of the conventional time scaling concept wherein derivative of each trajectory component is scaled by a different scaling function. Some additional criteria also needs to be satisfied by the scaling functions used in the GTS transformation to ensure a feasible trajectory deformation. We reserve that discussion till later on in the paper.
The proposed trajectory deformation based on GTS provides the following advantages and contributions over the current state of the art.

- Firstly, unlike [2], the proposed formulation does not require re-integration of state space models which for non-holonomic robots could be expensive.
- Secondly, the effectiveness of approaches like [3] and [4] based on the use of isometry groups are very limited on simple trajectories like straight lines and circular arcs. For example, the affine transformation approach of [4] can deform a straight line path into just a different straight line path (albeit rotated and scaled). Similarly, circular arcs can only be deformed into segments of a rotated and sheared ellipses. In contrast, the proposed trajectory deformation can transform straight line and circular segments into complex geometries. This key

The research was partly supported by TUT postdoc fellowship to the first author

[1] AUT, Tampere University of Technology, Finland

feature significantly simplifies the process of trajectory deformation for goal point correction or collision avoidance. Moreover, it also opens opens up a brand new application for trajectory deformation namely trajectory smoothing. We show how the smoothness of a trajectory can be improved while incurring only minimal deformation of the path associated with the original trajectory.

- The proposed trajectory deformation takes the form of a non-convex optimization problem. We present a solution approach based on Augmented Lagrangian (AL) and Alternating Minimization (AM) which is interesting in itself and could be useful for many robotic applications beyond the presented work.

The rest of the paper is organized as follows. Section II provides mathematical preliminaries useful for the exposition of the paper. Section III introduces the GTS. Section IV-VI details various optimizations and their solution process for trajectory deformation.

II. Preliminaries

A. Trajectory Feasibility for Non Holonomic Robots

Following [4], we characterize feasibility in terms of continuity and differentiability of a trajectory. Let, $\mathcal{C}^n$ denote the space of functions with n continuous derivatives. Then, non-holonomic robots require the trajectory be in $\mathcal{C}^2$. This in turn ensures that the curvature of the trajectory is continuous. Further, it would also ensure smooth and differentiable velocity and a continuous acceleration profile.

B. Augmented Lagrangian (AL)

Consider, the following optimization with convex cost and affine equality constraints in terms of variable $x \in \Re^n$

$$\min f(x) \text{ such that } Ax = B \tag{1}$$

The AL technique solves this problem by incorporating the affine equality constraints as a penalty in the cost function in the following manner [8].

$$f(x) + \overbrace{\lambda^T(Ax - B) + \rho\|Ax - B\|_2^2}^{augmented\ \ lagrangian}. \tag{2}$$

Where, $\lambda \in \Re^n$ is called the Lagrange multipliers and ρ is a positive constant which we will henceforth call as proximal weights (since they are associated with the proximal term). The solution of (2) can be computed through the following iterative process.

$$\begin{aligned} {}^{k+1}x = \arg\min f(x) + {}^{k}\lambda^T(Ax - B) + {}^{k}\rho\|Ax - B\|_2^2 \\ {}^{k+1}\lambda = {}^{k}\lambda + {}^{k}\rho(A^{k+1}x - B) \\ {}^{k+1}\rho = \Delta^{k}\rho \end{aligned}$$

As can be seen, at each iteration $(k + 1)$, we sequentially update the values of x and λ. The value of ρ is just usually increased by a factor of $\Delta > 1$ at each iteration. Intuitively, at each iteration, the AL technique looks for a solution which minimizes the cost as well as the residual of the affine equality constraints.

C. Alternating Minimization

Consider the following optimization problem in terms of variable $x,\, y \in \Re^n$

$$\min g(x, y). \tag{4}$$

Where $g(x, y)$ is a non-convex function with respect to x, y. However, if we fix x, it is convex in y and vice versa. AM technique provides a convenient approach for solving optimization problem with such structure and consists of following iterative steps [9].

$${}^{k+1}x = \arg\min g(x, {}^{k}y),\, {}^{k+1}y = \arg\min g({}^{k+1}x, y) \tag{5}$$

III. Generalized Time Scaling

Without loss of generalization, we consider a 2D trajectory $(x_1(t), x_2(t)), t \in [t_0\ \ t_f]$, though extensions to 3D is trivial. Generalized time scaling (GTS) corresponds to transformation of the independent variable t to some other arbitrary variable τ_1, τ_2 based on the following relation

$$\frac{dt}{d\tau_1} = \dot{s}_{x_1}(t), \frac{dt}{d\tau_2} = \dot{s}_{x_2}(t). \tag{6}$$

.

The functions, $\dot{s}_{x_1}(t), \dot{s}_{x_2}(t)$ are called the scaling function and decides the transformation between the variables t and τ_1, τ_2. Further, this transformation needs to be monotonically increasing and thus, we have the following necessary conditions on scaling functions.

$$\dot{s}_{x_1}(t), \dot{s}_{x_2}(t) > 0 \tag{7}$$

Using (6), the time scaled trajectory $(x_1(\tau_1), x_2(\tau_2))$ can be obtained in the following manner.

$$x_1(\tau_1 = t_0 + g_{x_1}(t)) = x_1(t), x_2(\tau_2 = t_0 + g_{x_2}(t)) = x_2(t). \tag{8}$$

$$g_{x_1}(t) = \int_{t=t_0}^{t=t} \frac{1}{\dot{s}_{x_1}(t)} dt, g_{x_2}(t) = \int_{t=t_0}^{t=t} \frac{1}{\dot{s}_{x_2}(t)} dt. \tag{9}$$

To understand (8) further, imagine $x_1(t)$ to be a 1D path which is traversed according to time t. After GTS transformation, some point $x_1(\tau_i)$ on the time scaled trajectory would be the same as some point $x_1(t_i)$ on the original trajectory. The exact correspondence is given by (8). The derivatives of the time scaled trajectory is given by the following equations.

$$\dot{x}_1(\tau_1 = t_0 + g_{x_1}(t)) = \dot{x}_1(t)\dot{s}_{x_1}(t) \tag{10a}$$

$$\dot{x}_2(\tau_2 = t_0 + g_{x_2}(t)) = \dot{x}_2(t)\dot{s}_{x_2}(t) \tag{10b}$$

Now, note the following sequence of remarks regarding GTS transformations.

- *Remark 1:* The ratio between the derivatives $\dot{x}_1(\tau_1)$ and $\dot{x}_2(\tau_2)$ can be manipulated by choosing appropriate scaling functions, $\dot{s}_{x_1}(t), \dot{s}_{x_2}(t)$. If $\dot{s}_{x_1}(t) = \dot{s}_{x_2}(t), \forall t$, then GTS reduces to the conventional time scaling under which the path associated with a trajectory remains unchanged.
- *Remark 2:* The paths associated with $(x_1(\tau_1), x_2(\tau_2))$ and $(x_1(t), x_2(t))$ should have the same start and end point. As can be seen from (8), the first condition is

trivially satisfied. For the satisfaction of second requirement, we must have

$$\int_{t_0}^{t_f} \frac{1}{\dot{s}_{x_1}(t)} dt = \int_{t_0}^{t_f} \frac{1}{\dot{s}_{x_2}(t)} dt \tag{11}$$

Intuitively, the above equality just means that 1D paths associated with $x_1(\tau_1)$ and $x_2(\tau_2)$ are traversed in same amount of time.

- *Remark 3:* It is straightforward to deduce that the deformation of the path associated with a trajectory through GTS transformation is purely a consequence of changing the pairing of points from the image of $x_1(t)$ and $x_2(t)$. Moreover, this change of ordered pairing is affected by manipulating the ratio between $\dot{x}_1(\tau_2)$ and $\dot{x}_2(\tau_2)$ through the scaling functions.

A. Choice of Scaling Functions

Most existing works on time scaling transformations assume some parametric functional representation for the scaling functions. For example, [5], [10] represents scaling functions as a combination of piece-wise linear functions. We have observed that the linear form has its own pros and cons in the context of the proposed work. One one hand it leads to a convex formulation [1] for the various optimizations that we discuss in the next sections for trajectory deformation. However, on the other hand, the linear form can only ensure a $\mathcal{C}^1$ trajectory. As discussed in Section II, non-holonomic robots require the trajectory to be in $\mathcal{C}^2$. Although, higher level continuity can be accomplished by assuming polynomial representations for the scaling functions [11], we have shown in our earlier work that such representations are not appropriate. Specifically, it is difficult to ensure the positive definiteness requirement (refer inequality (7)) in the polynomial scaling functions [7]. Thus, we follow [7] and adopt the following representation wherein scaling functions are represented as exponential of polynomial functions.

$$\dot{s}_{x_1}(t) = \exp(P_{x_1}(t)), \dot{s}_{x_2}(t) = \exp(P_{x_2}(t)). \tag{12}$$

$$P_{x_1}(t) = \sum_{i=0}^{i=n} P_i(t) c_{x_1}^i, P_{x_2}(t) = \sum_{i=1}^{n} P_i(t) c_{x_2}^i \tag{13}$$

Here, $P_i(t)$ are the polynomial basis functions while $c_{x_1}^i$ and $c_{x_2}^i$ are the coefficients associated with the basis. To ensure a $\mathcal{C}^2$ trajectory, the polynomials needs to be atleast C^1. In general the polynomials in C^n class ensures that the trajectory belong to C^{n+1}.

IV. Terminal Heading Correction for Differential Drive Robots.

In this section, we apply GTS based trajectory deformation to modify the terminal heading of an already existing trajectory, $(x_1(t), x_2(t))$ of a non-holonomic differential drive robot. To this end, we note that the heading at any time t can be represented as $atan2(\dot{x}_2(t), \dot{x}_1(t))$. Thus, modifying the heading at the terminal point just requires a GTS transformation such that trajectory derivatives at the terminal point assume some appropriate values. Following the derivations presented in the Appendix, the constraint that velocities and accelerations assume some specific values at the initial and terminal points can be all written in affine form with respect to polynomial coefficients, $c_{x_1}^i$, $c_{x_2}^i$. Consequently, we formulate the requisite trajectory deformation as the following optimization problem.

$$\arg\min_{C_{x_1}, C_{x_2}} J_{smooth} = \sum_{i}^{i=n} \ddot{P}_{x_1}(t_i)^2 + \ddot{P}_{x_2}(t_i)^2 \tag{14a}$$

$$A_{x_1} C_{x_1} = B_{x_1} \tag{14b}$$

$$A_{x_2} C_{x_2} = B_{x_2} \tag{14c}$$

$$\sum_{i=0}^{i=m} \frac{1}{\exp(P_{x_1}(t_i))} = \sum_{i=0}^{i=m} \frac{1}{\exp(P_{x_2}(t_i))} \tag{14d}$$

$$C_{x_1} = \begin{bmatrix} c_{x_1}^0 \\ c_{x_1}^1 \\ . \\ . \\ c_{x_1}^n \end{bmatrix}, C_{x_2} = \begin{bmatrix} c_{x_2}^0 \\ c_{x_2}^1 \\ . \\ . \\ c_{x_2}^n \end{bmatrix}$$

Where, $A_{x_1}, A_{x_2}, B_{x_1}, B_{x_2}$ are constant matrices. The cost function, J_{smooth} is designed to minimize the norm of the double derivative of the polynomials. Such a cost function induces smoothness in the polynomials which translates to smoothness in the scaling functions and consequently the deformed trajectory. The affine constraints (14b)-(14c) model the boundary constraints on velocities and accelerations. The non-convex equality (14d) is derived from (11) by approximating the integrals through summation over the grid obtained by discretization of the time interval $[t_0 \;\; t_f]$ in m subintervals. It can be seen that the only difficulty in solving optimization (14a)-(14d) stems from the non-convex equality constraint (14d). Thus, we next present a novel reformulation based on AL and AM technique.

A. Reformulation

For each time instant, t_i, we introduce a pair of slack variables, $z_{x_1}^i$, $z_{x_2}^i$. Further, we stacked them in a vector form in the following manner.

$$Z_{x_1} = \begin{bmatrix} z_{x_1}^0 \\ z_{x_1}^1 \\ . \\ . \\ z_{x_1}^n \end{bmatrix}, Z_{x_2} = \begin{bmatrix} z_{x_2}^0 \\ z_{x_2}^1 \\ . \\ . \\ z_{x_2}^n \end{bmatrix}$$

We now use them to define the following functions.

$$R_{x_1}^Z = \frac{1}{\exp(P_{x_1}(t_i)} - z_{x_1}^i, \forall i = 0, 1, 2..n \tag{15a}$$

$$R_{x_2}^Z = \frac{1}{\exp(P_{x_2}(t_i)} - z_{x_2}^i, \forall i = 0, 1, 2..n \tag{15b}$$

$$R_{x_1}^P = P_{x_1}(t_i) - \log \frac{1}{z_{x_1}^i}, \forall i = 0, 1, 2..n \tag{15c}$$

$$R_{x_2}^P = P_{x_2}(t_i) - \log \frac{1}{z_{x_2}^i}, \forall i = 0, 1, 2..n \tag{15d}$$

[1]Due to lack of space, we do not present the convex reformulation with piece-wise linear scaling function

$$R_{eq} = \sum_{i=0}^{n} z_{x_1}^i - \sum_{i=0}^{n} z_{x_2}^i \quad (15e)$$

Here, $R_{x_1}^Z$, $R_{x_2}^Z$, $R_{x_1}^P$, $R_{x_2}^P$ are vector valued functions while while R_{eq} is a scalar. It can be seen that as $\|R_{x_1}^Z\|_1$, $\|R_{x_2}^Z\|_1$ and $\|R_{eq}\|_1$ simultaneously goes to zero, it leads to satisfaction of equality constraint (14d). The same effect can also be achieved by ensuring $\|R_{x_1}^P\|_1$, $\|R_{x_2}^P\|_1$ and $\|R_{eq}\|_1$ simultaneously goes to zero. With this insight as the basis we propose alogrithm 1 wherein AL and AM techniques are combined to construct and solve two hierarchical optimizations which assist each other to minimize the residual of (15a)-(15e) leading to a locally optimal solution of (14a)-(14d)

B. Algorithm 1

The first layer of optimization (line 3) minimizes the smoothness cost while satisfying the affine velocity and acceleration constraints (14b)-(14c). At the same time, it also ensures that $\|R_{x_1}^Z\|_1$, $\|R_{x_2}^Z\|_1$ is minimized while the slack variables are held fixed at ${}^kZ_{x_1}$, ${}^kZ_{x_2}$. As shown, this is achieved by defining an augmented Lagrangian penalty for $R_{x_1}^P$, $R_{x_2}^P$. The solution obtained at the first layer is used in the second layer optimization to minimize the augmented Lagrangian penalty over $R_{x_1}^Z$, $R_{x_2}^Z$ while the polynomial coefficients are held fixed at ${}^{k+1}C_{x_1}$, ${}^{k+1}C_{x_2}$. The Lagrangian multipliers are updated based on residuals of $\|R_{x_1}^P\|_1$, $\|R_{x_2}^P\|_1$, $\|R_{x_1}^Z\|_1$, $\|R_{x_2}^Z\|_1$ for the obtained solutions (lines 5-9), while the proximal weights are increased by a factor Δ. The algorithm continues till the residual of constraint (14d) and decrease in the smoothness cost between subsequent iterations is greater than a particular user defined threshold, ϵ (typically 10^{-3}).

It is straightforward to note that for the fixed polynomial coefficients, $R_{x_1}^Z$, $R_{x_2}^Z$ are affine with respect to the slack variables. Similarly, for the fixed slack variables, $R_{x_1}^P$, $R_{x_2}^P$ are affine with respect to polynomial coefficients. Thus, optimization at both the layers in algorithms 1 represent convex quadratic programming problems which can be solved efficiently. Moreover, the optimization at the first layer can even be decomposed into two decoupled problems for separately obtaining the two sets of polynomial coefficients, C_{x_1}, C_{x_2}.

C. Examples

As sample outputs of algorithm 1, consider Fig. 1(a) which shows the deformation of a circular trajectory to satisfy four new terminal heading requirements. Fig. 1(b) shows the residual of constraint (14d) at each iteration of algorithm 1. It can be seen that the residual gradually reduces to zero, thus, validating the efficacy of algorithm 1. The velocity and acceleration plots shown in Fig. 4(d) validates that the deformed trajectory is indeed $\mathcal{C}^2$.

We implemented algorithm 1 on a laptop with 12GB RAM, $i7$ processor with 2.5Ghz clock speed. The prototyping as done in Python with CVXOPT [12] as the optimization library. The mean computational time was observed to be around 40 ms without exploiting the distributed structure for obtaining C_{x_1}, C_{x_2} separately. The polynomials $P_{x_1}(t)$ and $P_{x_2}(t)$ were constructed from cubic spline basis with 50 knot points. So, the optimization consisted of 100 variables.

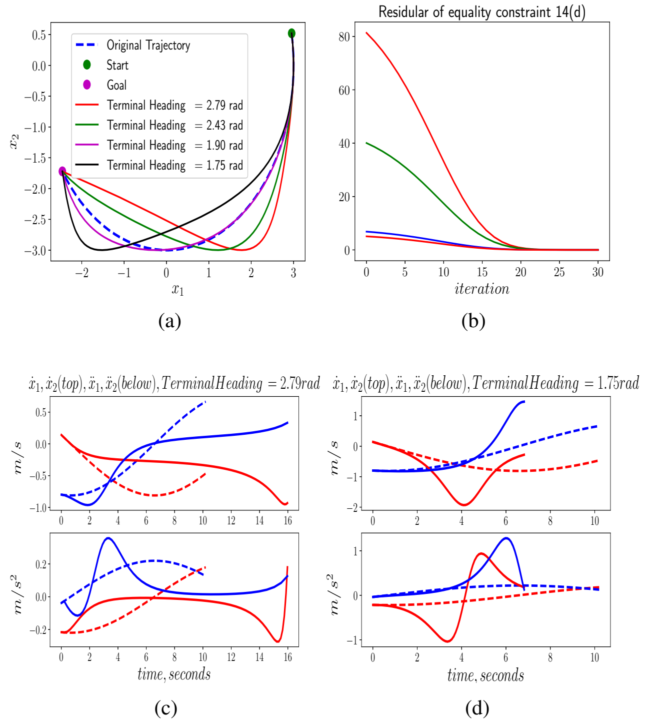


Fig. 1. (a) shows a circular trajectory being deformed to meet four new terminal heading requirements. (b) plots the residual of constraint (14d) with each iteration of algorithm 1. (c)-(d) shows the velocity and acceleration plots of the original(dotted) and two of the deformed (solid line) trajectories. The plots in red show the x_1 component while that in blue show the x_2 component.

V. Improving Trajectory Smoothness

In this section, we consider a set of trajectories $(x_1^j(t), x_2^j(t)), \forall j = 1, 2..m$ which only have a $\mathcal{C}^0$ continuity. The task is to deform each trajectory segment to ensure that the combined trajectory belongs to $\mathcal{C}^2$. To this end, we introduce scaling functions, $\dot{s}_{x_1}^j(t) = \exp(P_{x_1}^j(t))$, $\dot{s}_{x_2}^j(t) = \exp(P_{x_2}^j(t))$ for the j^{th}trajectory segments and formulate the following optimization problem.

$$\arg\min_{C_{x_1}, C_{x_2}} J_{smoothness} = \sum_{i}^{i=n} \sum_{j=1}^{m} \ddot{P}_{x_1}^j(t_i)^2 + \ddot{P}_{x_2}^j(t_i)^2 \quad (16a)$$

$$A_{x_1}^j C_{x_1}^j = A_{x_1}^{j+1} C_{x_1}^{j+1} \quad (16b)$$

$$A_{x_1}^j C_{x_1}^j = A_{x_1}^{j+1} C_{x_1}^{j+1} \quad (16c)$$

$$A_{x_1}^1 C_{x_1}^1 = B_{x_1}^1 \quad (16d)$$

$$A_{x_2}^1 C_{x_2}^1 = B_{x_2}^1 \quad (16e)$$

$$A_{x_1}^m C_{x_1}^m = B_{x_m}^m \quad (16f)$$

$$A_{x_2}^m C_{x_2}^m = B_{x_2}^m \quad (16g)$$

$$\sum_{i=0}^{i=n} \frac{1}{\exp(P_{x_1}^j(t_i))} = \sum \frac{1}{\exp(P_{x_2}^j(t_i))} \quad (16h)$$

Algorithm 1 Algorithm for Terminal Heading Correction

```
1: **Initialization**: Initial guess for slack ${}^kZ_{x_1}$ and ${}^kZ_{x_2}$, values of Lagrange multipliers $k\lambda^Z_{x_1}$, ${}^k\lambda^Z_{x_2}$, ${}^k\lambda^P_{x_1}$, ${}^k\lambda^P_{x_2}$ and constants ${}^k\rho^P_{x_1}$, ${}^k\rho^P_{x_2}$, ${}^k\rho^Z_{x_1}$, ${}^k\rho^P_{x_2}$, iteration counter, $k = 0$
2: **while** $\| \sum \frac{1}{\exp(P_{x_1}(t))} - \sum \frac{1}{\exp(P_{x_2}(t))} \|_1 \geq \epsilon$ and $|{}^{k+1}J_{smoothness} - {}^kJ_{smoothness}| \geq \epsilon$ **do**
3:     ${}^{k+1}C_{x_1}, {}^{k+1}C_{x_1} = \arg\min_{C_{x_1}, C_{x_2}} J_{smooth} \overbrace{+({}^k\lambda^P_{x_1})^T R^P_{x_1}({}^kZ_{x_1}) + {}^k\rho^P_{x_1}\|R_{x_1}({}^kZ^P_{x_1})\|^2 + ({}^k\lambda^P_{x_2})^T (R^P_{x_2}({}^kZ_{x_2})) + {}^k\rho^P_{x_2}\|R^P_{x_2}({}^kZ_{x_2})\|^2}^{Augmented\ \ Lagrnagian}$
           $A_{x_1}C_{x_1} = B_{x_1}$
           $A_{x_2}C_{x_2} = B_{x_2}$
4:     ${}^{k+1}Z_{x_1}, {}^{k+1}Z_{x_2} = \arg\min_{Z_{x_1}, Z_{x_2}} \overbrace{({}^k\lambda^Z_{x_1})^T R^Z_{x_1}({}^{k+1}C_{x_1}) + {}^k\rho^z_{x_1}\|R^Z_{x_1}({}^{k+1}C_{x_1})\|^2 + ({}^k\lambda^Z_{x_2})^T R^Z_{x_2}({}^{k+1}C_{x_2}) + {}^k\rho^Z_{x_2}\|R^Z_{x_2}({}^{k+1}C_{x_2})\|^2}^{Augmented\ \ Lagrnagian}$
           $R_{eq} = 0$
5:     ${}^{k+1}\lambda^P_{x_1} \leftarrow {}^k\lambda^P_{x_1} + {}^k\rho^P_{x_1} R^P_{x_1}({}^{k+1}C_{x_1}, {}^{k+1}C_{x_2}, {}^{k+1}Z_{x_1}, {}^{k+1}Z_{x_2})$
6:     ${}^{k+1}\lambda^P_{x_2} \leftarrow {}^k\lambda^P_{x_2} + {}^k\rho^P_{x_2} R^P_{x_2}({}^{k+1}C_{x_1}, {}^{k+1}C_{x_2}, {}^{k+1}Z_{x_1}, {}^{k+1}Z_{x_2})$
7:     ${}^{k+1}\lambda^Z_{x_1} \leftarrow {}^k\lambda^Z_{x_1} + {}^k\rho^Z_{x_1} R^Z_{x_1}({}^{k+1}C_{x_1}, {}^{k+1}C_{x_2}, {}^{k+1}Z_{x_1}, {}^{k+1}Z_{x_2})$
8:     ${}^{k+1}\lambda^Z_{x_2} \leftarrow {}^k\lambda^Z_{x_2} + {}^k\rho^Z_{x_2} R^Z_{x_2}({}^{k+1}C_{x_1}, {}^{k+1}C_{x_2}, {}^{k+1}Z_{x_1}, {}^{k+1}Z_{x_2})$
9:     $({}^{k+1}\rho^P_{x_1}, {}^{k+1}\rho^P_{x_2}, {}^{k+1}\rho^Z_{x_1}, {}^{k+1}\rho^Z_{x_2}) \leftarrow \Delta({}^k\rho^P_{x_1}, {}^k\rho^P_{x_2}, {}^k\rho^Z_{x_1}, {}^k\rho^Z_{x_2})$
10:    $k \leftarrow k + 1$
11: **end while**
```

$$C^j_{x_1} = \begin{bmatrix} c^{0j}_{x_1} \\ c^{1j}_{x_1} \\ . \\ . \\ c^{nj}_{x_1} \end{bmatrix}, C^j_{x_2} = \begin{bmatrix} c^{0j}_{x_2} \\ c^{1j}_{x_2} \\ . \\ . \\ c^{nj}_{x_2} \end{bmatrix}$$

Where, $A^j_{x_1}, A^j_{x_2}$, $B^j_{x_1}, B^j_{x_2}$ are the constant matrices and can be derived using a procedure similar to that shown in the Appendix. It can be seen that optimization (16a)-(16h) is just a generalization of (14a)-(14d). In the latter, we have already discussed how to choose the scaling functions to ensure specific velocity and acceleration values at the initial and terminal points. The same methodology is applied to ensure that the subsequent trajectory segments have velocity and acceleration continuity (16b)-(16c). Similarly, constraints (16d)-(16g) are constructed to ensure that the first and last trajectory segment has continuity at the velocity and acceleration level at the initial and terminal points respectively with the boundary values.

A. Reformulation

Here, we build upon the approach discussed in the previous section and start by introducing slack variables $z^{ij}_{x_1}$, $z^{ij}_{x_2}$ at time instant t_i for the j^{th} trajectory segment. We then stack them in the following vector form.

$$Z^j_{x_1} = \begin{bmatrix} z^{0j}_{x_1} \\ z^{1j}_{x_1} \\ . \\ . \\ z^{nj}_{x_1} \end{bmatrix}, Z^j_{x_2} = \begin{bmatrix} z^{0j}_{x_2} \\ z^{1j}_{x_2} \\ . \\ . \\ z^{nj}_{x_2} \end{bmatrix}$$

We subsequently used them to construct the following functions.

$$R^{Z^j}_{x_1} = \frac{1}{\exp(P^j_{x_1}(t_i))} - z^{ij}_{x_1}, \forall i = 0, 1, 2..n \quad (17a)$$

$$R^{Z^j}_{x_2} = \frac{1}{\exp(P^j_{x_2}(t_i))} - z^{ij}_{x_2}, \forall i = 0, 1, 2..n \quad (17b)$$

$$R^{P^j}_{x_1} = P^j_{x_1}(t_i) - \log\frac{1}{z^{ij}_{x_1}}, \forall i = 0, 1, 2..n \quad (17c)$$

$$R^{P^j}_{x_2} = P^j_{x_2}(t_i) - \log\frac{1}{z^{ij}_{x_2}}, \forall i = 0, 1, 2..n \quad (17d)$$

$$R^j_{eq} = \sum_{i=0}^{n} z^{ij}_{x_1} - \sum_{i=0}^{n} z^{ij}_{x_2} \quad (17e)$$

We now propose algorithm 2 which ensures that the residual of left hand side of (17a)-(17e) goes to zero while simultaneously minimizing the smoothness cost and satisfying the affine equality constraints (16b)-(16g).

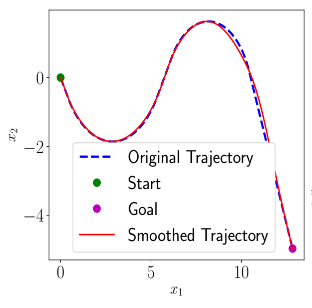
B. Algorithm 2

The algorithm 2 is a generalization of 1. The first layer optimization (line 3) now computes the polynomial basis for all the trajectory segments. The AL penalty is formed by $R^{P^j}_{x_1}$, $R^{P^j}_{x_2}$, $R^{Z^j}_{x_1}$ and $R^{Z^j}_{x_2}$ $\forall j$. The second layer optimization (line 4) is separately run for each trajectory segment. The Lagrange multipliers are updated based on the residual just like algorithm 2.

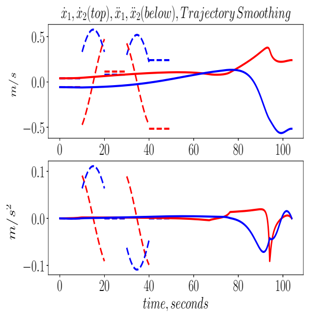
The distributiveness in algorithm 2 is much more prominent than algorithm 1. Just like the latter, the first layer optimization in the former can be split and solved in parallel to obtain the polynomial coefficients, $C^j_{x_1}$, $C^j_{x_2}$. However, the key distinction comes at line 4. As can be seen, the second layer optimization is decoupled for each trajectory segments and can be solved in parallel. This can lead to significant computational gain while considering large number of trajectory segments.

C. Example

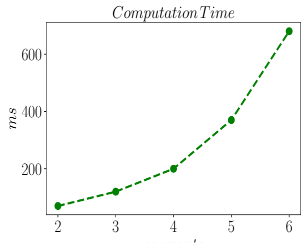
A sample output of algorithm 2 is shown in Fig. 2(a) where five trajectory segments consisting of piece-wise straight lines (S) and circles (C) are given. As well known, such CS trajectories belong to $\mathcal{C}^0$. Thus, the corresponding velocity and acceleration plots are discontinuous (dotted red and blue lines in Fig. 2(b)). In contrast, the velocity and acceleration plots of the smoothed trajectory are respectively once differentiable and continuous. An important point to note from Fig. 2(a) is how a very minimal deviation from the original path is required to smoothen out the velocity and acceleration profile. Fig. 2(c) presents the computation time of the trajectory smoothing as a function of the trajectory segments. Note that here we do not exploit the distributive nature of algorithm 2 and thus, there is a good potential for further reduction in computation time. The computation hardware and prototyping environment is same as that mentioned in the previous section.



(a)



(b)



(c)

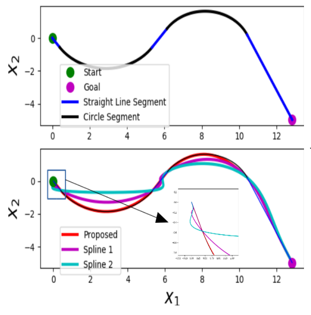
Fig. 2. (a) shows a Circle-Straight line (CS) trajectory and the smoothed $\mathcal{C}^2$ trajectory. (b) shows the velocity and acceleration plots for the CS (dotted) and smoothed trajectory (solid line). The plots shown in red shows the x_1 component while that in blue shows the x_2 component. Note that the velocity and acceleration plots of CS trajectory are discontinuous. (c) shows the computation time of trajectory smoothing as a function of number of trajectory segments.

D. Comparisons with Spline Based Trajectory Smoothing

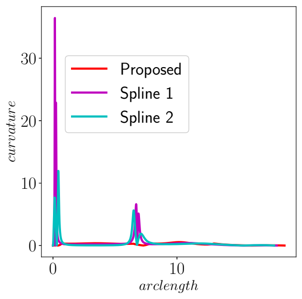
An alternate and popular approach for trajectory smoothing is based on the use of cubic splines. Here we compare the spline based trajectory smoothing with that obtained via proposed formulation based on trajectory deformation.

To this end, we associate each segment of the CS trajectory shown in Fig. 3(a) with a cubic spline polynomial. Subsequently, we formulated the trajectory smoothing as an optimization problem which trades-off spline smoothness (squared norm of jerk) with its closeness to the given CS trajectory. Our implementation is similar to that proposed in [13] except that we do not include an bounds on maximum velocity and accelerations. Moreover, the spline optimization also have additional constraints which ensures that the smoothed spline trajectory passes exactly through the intermediate points where the straight lines and circular segments are joined. This was done for fair comparison as the proposed formulation also satisfy such constraints by design. Moreover, such constraints are also important from practical application standpoint as robot trajectories are often required to pass through some key way-points.

A sample result is shown in Fig. 3(b). The comparison between the trajectories is done based on their curvature magnitudes. As can be seen, for the given CS trajectory, the proposed formulation results in smoothing with far smaller curvature magnitudes (max. of $0.55m^{-1}$) than those obtained through spline based smoothing (max. of $36.44m^{-1}$. and $11.95m^{-1}$). In general, we have observed in our simulations that if the lengths of different segments of the given trajectory vary significantly, then spline based smoothing fairs very poorly as compared with the proposed formulation. A similar disadvantage of spline based smoothing was also pointed out in [14]. However, it is important to mention that spline smoothing was around five times faster than our proposed formulation.



(a)



(b)

Fig. 3. (a) The top figure shows the given CS trajectory. Note how the different segments vary significantly in length. The bottom figure shows the smoothed trajectory based on proposed and cubic spline based formulation. The two spline trajectories are obtained by varying the trade-off between smoothness and closeness to the CS trajectory. (b) plot of curvature profile for the smoothed trajectories.

VI. Goal Point Correction and Collision Avoidance

The trajectory deformation for goal point correction and collision avoidance essentially builds on the ability to smoothen straight line and circular trajectories. The motivation behind such an approach is that planning with straight line trajectories is relatively simple.

Algorithm 2 Trajectory Smoothing

```
1: Initialization: Initial guess for slack, ${}^kZ^j_{x_1}$ and ${}^kZ^j_{x_2}$, values of Lagrange multipliers ${}^k\lambda^{Z^j}_{x_1}$, ${}^k\lambda^{Z^j}_{x_2}$, ${}^k\lambda^{P^j}_{x_1}$, ${}^k\lambda^{P^j}_{x_2}$ and constants ${}^k\rho^{P^j}_{x_1}$, ${}^k\rho^{P^j}_{x_2}$, ${}^k\rho^{Z^j}_{x_1}$, $\rho^{P^j}_{x_2}$, iteration
   counter $k = 0$.
2: while $\| \sum \frac{1}{\exp(P^j_{x_1}(t))} - \sum \frac{1}{\exp(P^j_{x_2}(t))} \|_1 \geq \epsilon, \forall j$ and $|{}^{k+1}J_{smoothness} - {}^kJ_{smoothness}| \geq \epsilon$ do
3:     $${}^{k+1}C^j_{x_1}, {}^{k+1}C_{x_1} = \arg\min J_{smooth} + \sum_{j=1}^{m} \overbrace{({}^k\lambda^{P^j}_{x_1})^T R^{P^j}_{x_1}({}^kZ^j_{x_1}) + \rho^{P^j}_{x_1}\|R^{P^j}_{x_1}({}^kZ^j_{x_1})\|^2 + ({}^k\lambda^{P^j}_{x_2})^T(R^{P^j}_{x_2}({}^kZ^j_{x_2})) + \rho^{P^j}_{x_2}\|R^{P^j}_{x_2}({}^kZ^j_{x_2})\|^2}^{Augmented \quad Lagrnagian}$$
       $$A^j_{x_1}C^j_{x_1} = A^{j+1}_{x_1}C^{j+1}_{x_1}$$
       $$A^j_{x_1}C^j_{x_1} = A^{j+1}_{x_1}C^{j+1}_{x_1}$$
       $$A^1_{x_1}C^1_{x_1} = B^1_{x_1}$$
       $$A^1_{x_2}C^1_{x_2} = B^1_{x_2}$$
       $$A^m_{x_1}C^m_{x_1} = B^m_{x_m}$$
       $$A^m_{x_2}C^m_{x_2} = B^m_{x_2}$$
4:     for j=1:m do
           $${}^{k+1}Z^j_{x_1}, {}^{k+1}Z^j_{x_2} = \arg\min \overbrace{({}^k\lambda^{Z^j}_{x_1})^T R^{Z^j}_{x_1}({}^{k+1}C^j_{x_1}) + {}^k\rho^{Z^j}_{x_1}\|R^{Z^j}_{x_1}({}^{k+1}C^j_{x_1})\|^2 + ({}^k\lambda^{Z^j}_{x_2})^T R^{Z^j}_{x_2}({}^{k+1}C^j_{x_2}) + {}^k\rho^{Z^j}_{x_2}\|R^{Z^j}_{x_2}({}^{k+1}C^j_{x_2})\|^2}^{Augmented \quad Lagrnagian}$$
           $$R^j_{eq} = 0$$
5:     end for
6:     ${}^{k+1}\lambda^{P^j}_{x_1} \leftarrow {}^k\lambda^{P^j}_{x_1} + {}^k\rho^{P^j}_{x_1} R^{P^j}_{x_1}({}^{k+1}C^j_{x_1}, {}^{k+1}C^j_{x_2}, {}^{k+1}Z^j_{x_1}, {}^{k+1}Z^j_{x_2}), \forall j$
7:     ${}^{k+1}\lambda^{P^j}_{x_2} \leftarrow {}^k\lambda^{P^j}_{x_2} + {}^k\rho^{P^j}_{x_2} R^{P^j}_{x_2}({}^{k+1}C^j_{x_1}, {}^{k+1}C^j_{x_2}, {}^{k+1}Z^j_{x_1}, {}^{k+1}Z^j_{x_2}), \forall j$
8:     ${}^{k+1}\lambda^{Z^j}_{x_1} \leftarrow {}^k\lambda^{Z^j}_{x_1} + {}^k\rho^{Z^j}_{x_1} R^{Z^j}_{x_1}({}^{k+1}C^j_{x_1}, {}^{k+1}C^j_{x_2}, {}^{k+1}Z^j_{x_1}, {}^{k+1}Z^j_{x_2}), \forall j$
9:     ${}^{k+1}\lambda^{Z^j}_{x_2} \leftarrow {}^k\lambda^{Z^j}_{x_2} + {}^k\rho^{Z^j}_{x_2} R^{Z^j}_{x_2}({}^{k+1}C^j_{x_1}, {}^{k+1}C^j_{x_2}, {}^{k+1}Z^j_{x_1}, {}^{k+1}Z^j_{x_2}), \forall j$
10:    $({}^{k+1}\rho^{P^j}_{x_1}, {}^{k+1}\rho^{P^j}_{x_2}, {}^{k+1}\rho^{Z^j}_{x_1}, {}^{k+1}\rho^{Z^j}_{x_2}) \leftarrow \Delta({}^k\rho^{P^j}_{x_1}, {}^k\rho^{P^j}_{x_2}, {}^{kk}\rho^{Z^j}_{x_1}, {}^k\rho^{Z^j}_{x_2}), \forall j$
11:    $k \leftarrow k + 1$
12: end while
```

The proposed goal point correction is similar to that presented in [4] as it starts with adding new segments to the original trajectory. As shown in Fig. 4(a), the goal point correction starts with constructing two straight line trajectories (dotted red lines), one each from the old and new goal point (Fig. 4(a)). The slope of the straight line trajectories is given by the heading angle at the old and new goal point respectively. A third straight line trajectory (shown in black) connects the previous two constructed trajectories. Thereafter, the newly constructed straight line trajectory is converted into a CS trajectory (Fig. 4(b)). Finally, the trajectory deformation discussed in the previous section is applied from start to goal location to get the final trajectory to the new goal point.

The trajectory deformation for collision avoidance also follows similar approach as above. Here, we exploit the fact that collision avoidance with straight line trajectories is relatively simple. For example, in our implementation, we use the concept of velocity obstacle [15] for holonomic robots to get the collision avoiding straight line trajectory (red dotted line in Fig. 5(a). This is subsequently followed by conversion to CS trajectory followed by trajectory deformation for smoothing.

VII. Discussions and Future Work

In this paper, we have presented a novel take on trajectory deformation through the generalization of time scaling concept. The proposed trajectory deformation essentially reduces to individually manipulating the derivatives of each trajectory component. We have exploited this observation to present optimization problems for terminal heading correction and trajectory smoothing of non-holonomic robots. We have shown that optimization problem itself is very interesting but at the same time extremely challenging. We used a combination of Augmented Lagrangian and Alternating Minimization technique to efficiently solve the formulated optimization problem. Further, solution process was shown to be amenable to parallelization as well. One of the key features of the proposed trajectory deformation is that it can directly deform straight line and circular trajectories into complex shapes. This is an improvement over the current state of the art like [4].

There are several directions to extend the current work. Firstly, we have not incorporated the bounds on velocities and acceleration magnitudes. In our opinion, it would be possible to use the GTS transformation to formulate a minimum time trajectory computation which simultaneously optimizes the spatial paths as well as the motion profile. To this end, it can act as a generalization to existing frameworks

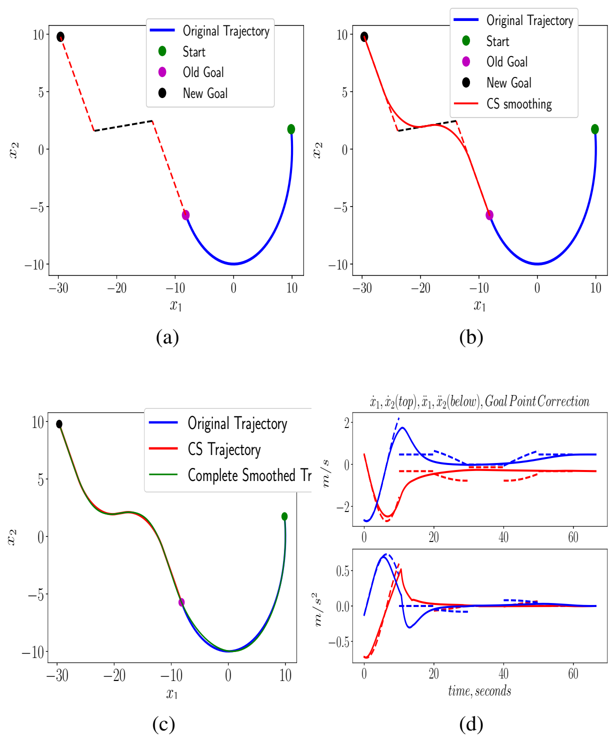


Fig. 4. (a) shows the straight line trajectories constructed to the new goal location. (b): Converting the straight line to CS trajectory. (c) shows the CS trajectory being smoothened, resulting in the requisite trajectory to the new goal point. (d) shows the velocity and acceleration plots for the CS (dotted) and smoothened trajectory (solid lines). The plots shown in red shows the x_1 component while that in blue shows the x_2 component.

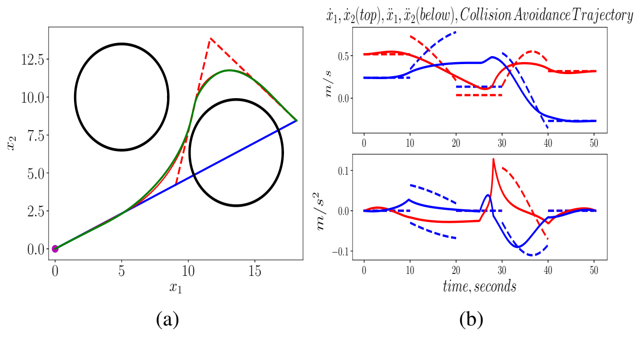


Fig. 5. (a) shows a straight line trajectory which passes through an obstacle. A new straight line trajectory is obtained for collision avoidance followed by its conversion to CS trajectory. Finally the CS trajectory is smoothened to get the final collision avoidance trajectory. (b) shows the velocity and acceleration plots for the CS (dotted line) and final smoothened trajectory (solid line). The plots shown in red shows the x_1 component while that in blue shows the x_2 component

like [5], [6], [7]. Finally, we are currently extending the proposed trajectory deformation to collision avoidance in dynamic environments.

VIII. Appendix: Velocity and Acceleration Constraints

In this section, we show that the velocity and acceleration boundary constraints can all be written in affine form. To this end, consider the following terminal constraint on velocity

$$\dot{x}_1(\tau_f) = \dot{x}_1^f \Rightarrow \dot{s}_{x_1}(t_f)\dot{x}_1(t_f) = \dot{x}_1^f \tag{18a}$$

$$\Rightarrow P_{x_1}(t_f) = \log \frac{\dot{x}_1^{term}}{\dot{x}_1(t_f)} \tag{18b}$$

Since, $\dot{x}_1(t_f), \dot{x}_1^f$ are given, (18b) is affine with respect to polynomial coefficients. A similar process would put the acceleration terminal constraints in the following affine form.

$$\dot{s}_{x_1}^2(t_f)\ddot{x}_1(t_f) + \ddot{s}_{x_1}(t_f)\dot{x}_1(t_f) = \ddot{x}_1^f \tag{19a}$$

$$\Rightarrow \dot{s}_{x_1}^2(t_f)(\ddot{x}_1(t_f) + \dot{P}_{x_1}\dot{x}_1(t_f)) = \ddot{x}_1^f \tag{19b}$$

$$\Rightarrow \ddot{x}_1(t_f) + \dot{P}_{x_1}\dot{x}_1(t_f) = \frac{\ddot{x}_1^f \dot{x}(t_f)^2}{(\dot{x}_1^f)^2} \tag{19c}$$

Following the above discussions, the constant matrices in optimizations (14a)-(14d) and (16a)-(16h) can be derived.

References

[1] S. Quinlan and O. Khatib, “Elastic bands: Connecting path planning and control,” in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 802–807.

[2] F. Lamiraux and D. Bonnafous, “Reactive trajectory deformation for nonholonomic systems: Application to mobile robots,” in *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, vol. 3. IEEE, 2002, pp. 3099–3104.

[3] K. M. Seiler, S. P. Singh, S. Sukkarieh, and H. Durrant-Whyte, “Using lie group symmetries for fast corrective motion planning,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 151–166, 2012.

[4] Q.-C. Pham, “Fast trajectory correction for nonholonomic mobile robots using affine transformations,” *2012 in Robotics: Science and Systems VII*, pp. 265–272, 2012.

[5] K. Hauser, “Fast interpolation and time-optimization with contact,” *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1231–1250, 2014.

[6] Q.-C. Pham, “A general, fast, and robust implementation of the time-optimal path parameterization algorithm,” *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1533–1540, 2014.

[7] A. K. Singh and K. M. Krishna, “A class of non-linear time scaling functions for smooth time optimal control along specified paths,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 5809–5816.

[8] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[9] S. Boyd, “Sequential convex programming,” *Lecture Notes, Stanford University*, 2008.

[10] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, “Time-optimal path tracking for robots: A convex optimization approach,” *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[11] Q. Zhang, S.-R. Li, and X.-S. Gao, “Practical smooth minimum time trajectory planning for path following robotic manipulators,” in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 2778–2783.

[12] M. S. Andersen, J. Dahl, and L. Vandenberghe, “Cvxopt: A python package for convex optimization, version 1.1. 6,” *Available at cvxopt. org*, vol. 54, 2013.

[13] H. Kano and H. Fujioka, “Velocity and acceleration constrained trajectory planning by smoothing splines,” in *Industrial Electronics (ISIE), 2017 IEEE 26th International Symposium on*. IEEE, 2017, pp. 1167–1172.

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

[15] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using the relative velocity paradigm,” in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 560–565.