Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

Generation of "optimal" speed profile for motion planning.

Fabien Gravot, Yutaka Hirano and Shintaro Yoshizawa

***Abstract*— This paper presents how to compute "optimal" speed profile on a path that can be made by a path planner. "Optimality" is defined as the minimization of the trajectory total time under constraints on any number of its derivatives (speed, acceleration, jerk, ...). Then the resulting trajectory can be used in real system to have precise control over it.**

I. INTRODUCTION

An autonomous mobile robot needs to be able to find its way. For that, two main methods exist: behavior base [4] and motion planning [3]. Motion planning tries to find a path without collision generally by relaxing time constraints.

In this paper we will focus on non-dynamic problems. In a dynamic problem (for instance air plane) the path is a function of speed (if an air plane slows down it falls). Still, non-dynamic planning includes non-holonomic problem like car (slipping motion is not taken into account).

In this case the path planning has two steps: computing a path without collision (with or without kinematics constraints), then computing a speed profile on the resulting path to obtain a feasible trajectory.

Path planning has been widely studied, in particular random planning technique [3]. So in this paper we only focus on the second step: computing a speed profile on a given path. Surprisingly the number of researches on this domain is not very large, they focus on specific systems [1], [5]. The most common approach is based on a simple re-sampling to respect a maximum speed. However, for precise control we often have constraints on acceleration and jerk. We think that this problem can profit from a rigorous study.

In this paper we will present a general approach for generating speed profile that minimizes the total trajectory's time under constraints on speed, acceleration or any other derivatives. For that we first introduce the notations and terms used. Then we present a general algorithm to compute a speed profile along a sequence of direct path. Afterward, we will introduce a mathematical tool: "Ternary Polynomial" to help us to compute "optimal" speed profile for two specific "direct paths": linear path and smoothed path.

A. Notations

A configuration q is the vector of degrees of freedom of the system. A "direct path" i is a function that give the evolution of the configuration with a parameter p:

$$q = path_i(p) \tag{1}$$

F. Gravot is with Knowledge*Net*, JAPAN, gravot@nano.tec.toyota.co.jp
Y. Hirano and S. Yoshizawa are with Toyota Motor Corporation, JAPAN yutaka@hirano.tec.toyota.co.jp shintaro@yoshizawa.tec.toyota.co.jp

To simplify the equation we suppose that $p \in [0, 1]$. A path is a sequence of N direct paths defined on $[0, N]$:

$$q = Path(p) = path_i\,(p - i) \text{ with } p \in [i, i+1] \tag{2}$$

A speed profile is given by a re-parametrization of the path by a time function. For each direct path i we can define a speed profile function between $[0, t_{f\,i}]$:

$$p = f_i(t),\ f_i(0) = 0,\ f_i(t_{f\,i}) = 1 \text{ and } f_i'(t) \geq 0 \tag{3}$$

A direct trajectory is a direct path in function of time.

$$q = traj_i(t) = (path_i \circ f_i)\,(t) = path_i\,(f_i(t)) \tag{4}$$

A trajectory is then the sequence of direct trajectory defined on $[0, TF]$ with $TF = \sum_{i=1}^{N} t_{f\,i}$:

$$\begin{aligned} q = Traj(t) \quad &= \quad traj_i\left(t - \textstyle\sum_{j=1}^{i-1} t_{f\,j}\right) \\ &\text{with } t \in \left[\textstyle\sum_{j=1}^{i-1} t_{f\,j}, \sum_{j=1}^{i} t_{f\,j}\right] \end{aligned} \tag{5}$$

We define a "temporal direct path" as direct path on which it is possible to compute:

1) The speed profile $f_i(t)$ knowing $v_0 = f_i'(0)$ and $v_f = f_i'(t_f)$ and the derivatives constraints.
2) The maximal possible value of $f_i'(t_f)$ with $v_0 = f_i'(0)$ (function $MaxEndSpeed_i(v_0)$).
3) The maximal possible value of $f_i'(0)$ with $v_f = f_i'(t_f)$ (function $MaxStartSpeed_i(v_f)$).

We define the constraints on the trajectory by n norms (one for each derivative) $|\,.\,|_1$, ..., $|\,.\,|_n$ that limit the trajectory derivatives:

$$\left|Traj^{(k)}(t)\right|_k \leq M_k \text{ for } k \in [\![1, n]\!] \tag{6}$$

Where $|\,.\,|_k$ can be any kind of norm, for instance, pondered norm max to give limits on each degree of freedom, euclidean norm, ... $[\![1, n]\!]$ is defined by $\{1, \ldots, n\}$.

II. GENERAL ALGORITHM

In this section we present a general algorithm to compute a valid (the constraints (6)) speed profile along a path composed of a succession of "temporal direct path".

A. Properties

The algorithm finds the fastest trajectory (minimal total time under (6)) when the speed is the only free parameter. It also guarantees the speed continuity, but not the other derivatives continuity.

Most of the resulting trajectory's properties depend on the temporal direct path's properties. In the following sections

(§IV, V) we will present two "temporal direct paths" that have the following properties for their speed profile $f(t)$:

$$\left|(path \circ f)^{(k)}(t)\right|_k \leq M_k \quad (7)$$

$$(path \circ f)^{(k)}(0) = \vec{0} \text{ for } k \in [\![2, n-1]\!] \quad (8)$$

$$(path \circ f)^{(k)}(t_f) = \vec{0} \text{ for } k \in [\![2, n-1]\!] \quad (9)$$

The consequence of the property (7) is that each direct trajectory satisfy the constraints (6). The consequence of properties (8, 9) is the continuity of the n-1 first derivatives between direct trajectories. Then the whole trajectory respects the constraints (6).

Moreover, each temporal direct path has also good properties for finding the fastest direct trajectory. This leads to a very efficient algorithm.

B. Algorithm

The algorithm has three steps (Fig. 1). A forward phase can be seen as the acceleration constraints computation. In the same way the backward phase takes deceleration constraints into account. The final phase computes the speed profile for each temporal direct path according to the results of the two previous phases.

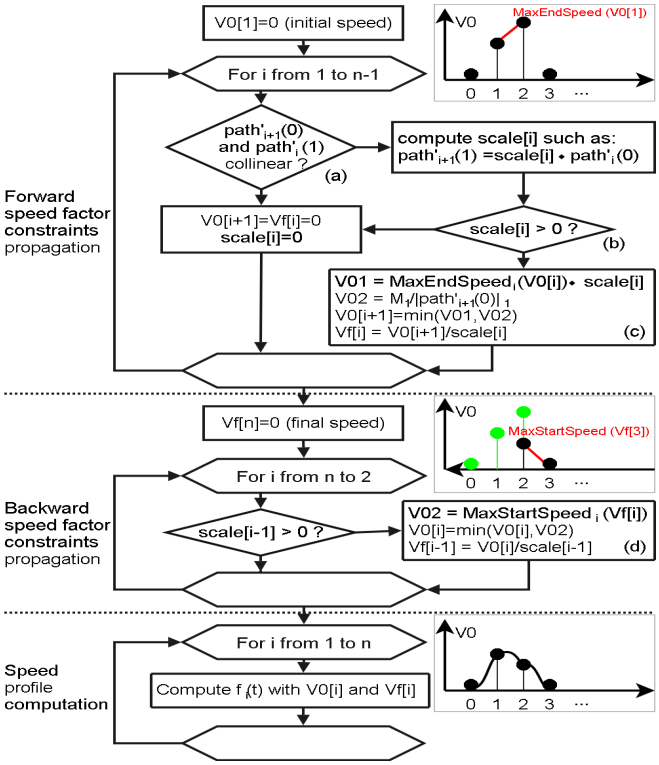

Fig. 1. Whole trajectory speed profile computation.

The goal of the algorithm is to compute the optimal values for $f_i'(0)$ and $f_i'(t_{f\,i})$ to use in the last phase. Those values are stored respectively in arrays V0 and Vf. The constraints on speed is $|traj_i'(t)|_1 \leq M_1$. Moreover, we need to have speed continuity (12). So we must have:

$$V0[i] \cdot |path_i'(0)|_1 \leq M_1 \quad (10)$$

$$Vf[i] \cdot |path_i'(1)|_1 \leq M_1 \quad (11)$$

$$Vf[i] \cdot path_i'(1) = V0[i+1] \cdot path_{i+1}'(0) \quad (12)$$

Equation (12) imposes the collinearity (Fig. 1.a) and the same orientation (Fig. 1.b) since $f_i'(t) \geq 0$. If those conditions are not met, the only continuous solution is to stop the system. In the other case V0 and Vf must respect (10) and (11) (included in $MaxEndSpeed_i(v)$) (Fig. 1.c). Moreover, they must respect other derivative constraints ($MaxEndSpeed_i(v)$ in Fig. 1.c and $MaxStartSpeed_i(v)$ in Fig. 1.d).

We have presented here a generic algorithm to compute the speed profile of a trajectory composed of "temporal direct paths". We now focus on the study of two types of "temporal direct path" and for that we start by the definition of a mathematical tool: the "ternary polynomial".

III. TERNARY POLYNOMIAL

The "ternary polynomial" is a mathematical tool used to compute optimal speed profile. It is based on polynomials defined on interval recursively divided in 3 parts (Fig. 2).

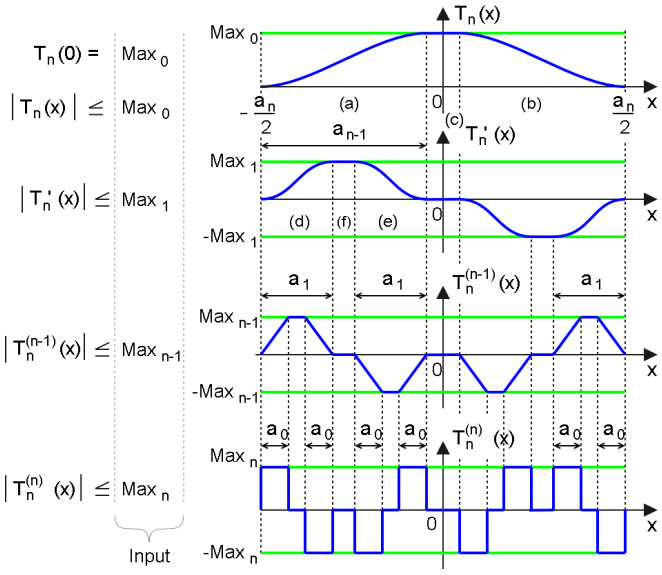

Fig. 2. Ternary polynomial and its constraints.

If we see the "ternary polynomial" as an "optimal" speed profile, we always have an acceleration phase (Fig. 2.a), a constant speed phase (Fig. 2.c) and a deceleration phase (Fig. 2.b). This is also true for other derivative. The acceleration phase is also composed of 3 phases: positive jerk phase (Fig. 2.d), constant acceleration phase (Fig. 2.f) and negative jerk phase (Fig. 2.f).

The ternary polynomial $T_n(x)$ is build to be the fastest (smallest a_n) function to go from 0 to Max_0 with limitation constraints on its n first derivatives. It satisfies the properties:

$$T_n(0) = Max_0 \quad (13)$$

$$T_n^{(k)}\left(-\tfrac{a_n}{2}\right) = T_n^{(k)}\left(\tfrac{a_n}{2}\right) = 0 \text{ for } k \in [\![0, n-1]\!] \quad (14)$$

$$|T_n^{(k)}(x)| \leq Max_k \text{ for } k \in [\![0, n]\!] \quad (15)$$

A. Parameters computation

In our description in Fig. 2 we see that the only parameters of $T_n(x)$ are the intervals length a_i. We must compute them to be able to determine $T_n(x)$.

To compute constraints on a_i we use one property of the ternary polynomial (Fig.3):

$$T_k^{(i)}(x) = T_{k-1}^{(i-1)}\left(x + \tfrac{a_k - a_{k-1}}{2}\right) - T_{k-1}^{(i-1)}\left(x - \tfrac{a_k - a_{k-1}}{2}\right) \quad (16)$$

With $T_0(x) = Max_n$ for $x \in]-a_0/2, a_0/2[$ and $T_0(x) = 0$ otherwise. We also set $a_{-1} = 0$. All T_k must also satisfy the properties (14) and (15). The inequality (15) must be modified to make the distinction between n index and n number of constraints:

$$|T_m^{(k)}(x)| \leq Max_{n-m+k} \text{ for } k \in [\![0, m]\!],\ m \in [\![0, n]\!] \quad (17)$$

From (16) we find the following properties:

$$max\left(|T_m^{(k)}(x)|\right) = max\left(|T_{m-k}(x)|\right) \quad (18)$$

$$max\left(|T_m(x)|\right) = \int_{-\infty}^{+\infty} T_{m-1}(x) \cdot dx \quad (19)$$

$$\int_{-\infty}^{+\infty} T_m(x) \cdot dx = Max_n \cdot \prod_{l=0}^{m} (a_l - a_{l-1}) \quad (20)$$

Equations (17, 18, 19, 20) allow to define a_i's constraints:

$$Max_n \cdot \prod_{l=0}^{i-1} (a_l - a_{l-1}) \leq Max_{n-i} \text{ for } i \in [\![0, n]\!] \quad (21)$$

$$\text{by definition: } a_i \geq 2 \cdot a_{i-1} \quad (22)$$

To minimize a_n and solve (13) we need to recursively maximize the value of $a_0, a_1, \ldots a_n$. To compute one a_i we need to take into account all the constraints on $T_n^{(j)}(x)$ for $j \geq i$. From (13, 21, 22) we can compute the constraints on a_i independently of a_j with $j > i$.

For each a_i starting with a_0, we compute the polynomial $P_{i:j}(x)$ that represents the constraints on $Max_{n-j-i-1}$ for $j \in [\![1, n-i]\!]$:

$$P_{i:j}(x) = x^j - a_{i-1} \cdot x^{j-1} - \frac{Max_{n-j-i}}{2^{\frac{(j-2)(j-1)}{2}} \cdot Max_n \prod_{l=0}^{i-1} (a_l - a_{l-1})} \quad (23)$$

We now can define $r_{i:j}$ as the maximal positive root of $P_{i:j}(x)$. To be valid a_i must respect:

$$2 \cdot a_{i-1} \leq a_i \leq r_{i:j} \text{ for } j \in [\![1, n-i]\!] \quad (24)$$

In order to minimize a_n, we must successively maximize the a_i starting from a_0. Hence, the optimal value for a_i is:

$$a_i = \min_{j \in [\![1, n-i]\!]} (r_{i:j}) \quad (25)$$

Note that (23, 25) for $i = n-1$ is equivalent to solve the property (13) on T_n.

The computation of the maximal root of $P_{i:j}(x)$ is very efficient since for $i = 0$ or $j = 1$, $P_{i:j}(x)$ is trivial, otherwise we have $r_{i:j} \in I_{i:j} = [2 \cdot a_{i-1}, max_{r_{i:j}}]$ and $P_{i:j}(x)$ strictly increasing on $I_{i:j}$. Then it is possible to use Newton's method to find $r_{i:j}$ with:

$$max_{r_{i:j}} = \left(\frac{Max_{n-j-i}}{2^{\frac{(j-2)(j-1)}{2}} \cdot a_{i-1} \cdot Max_n \prod_{l=0}^{i-1} (a_l - a_{l-1})} \right)^{\frac{1}{j-1}} \quad (26)$$

B. Ternary polynomial computation

Once we have the parameters a_i we can compute the ternary polynomial, i.e. compute the polynomials that compose T_n on each of its sub-intervals.

For that we use the property (16) to build an algorithm (Fig. 3) which uses symmetries to limit the number of integrations: We start with $T_0(x)$ defined by only one polynomial: $P(x) = Max_n$ on $]-a_0/2, a_0/2[$ (Fig. 3.a). Then we build $T_1'(x)$ based on $T_0(x)$ and its symmetric. $T_1'(x)$ is composed of three polynomials: $P(x) = Max_n$ on $]-a_1/2, -a_1/2+a_0[$, $P(x) = 0$ on $[-a_1/2+a_0, a_1/2-a_0]$ and $P(x) = -Max_n$ on $]a_1/2 - a_0, a_1/2[$ (Fig. 3.b). We can integrate those polynomials to obtain $T_1(x)$ (Fig. 3.c). We continue until we find $T_n(x)$.

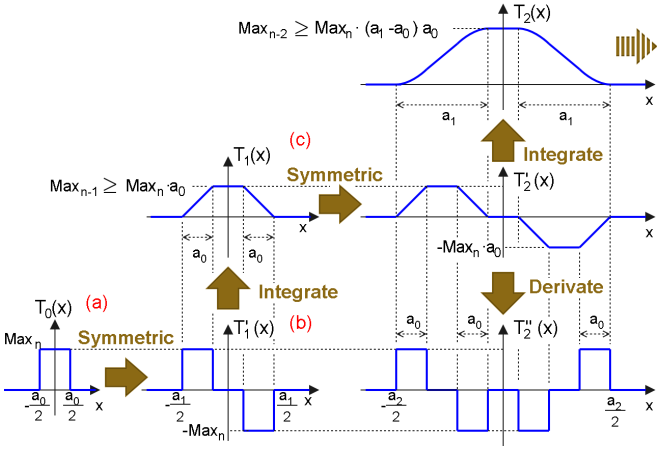

Fig. 3. Recursive computation of a Ternary Polynomial.

Ternary polynomials are a mathematical tool for computing the fastest function to reach Max_0 with respect to constraints on its derivatives. We will use them in the next two sections to compute "optimal" temporal direct paths.

IV. DIRECT LINEAR PATH

A direct linear path is a path defined by the function:

$$path(p) = q_{start} \cdot (1-p) + q_{end} \cdot p \quad (27)$$

This is the most common direct path found in motion planning. In this section we define a direct linear path that

satisfy the 3 "temporal direct path" properties (§I-A) and the constraints (6). For that we use the "ternary polynomial".

We can remark than solving (6) is equivalent to solve:

$$\begin{aligned} |f^{(k)}(t)| &\cdot |q_{end} - q_{start}|_k \leq M_k \\ |f^{(k)}(t)| &\leq FMax_k = \frac{M_k}{|q_{end} - q_{start}|_k} \end{aligned} \quad (28)$$

A. Maximal possible speed

$MaxEndSpeed_i(v)$ and $MaxStartSpeed_i(v)$ are symmetric problems, here we will solve $MaxEndSpeed_i(v)$.

The problem is to find the maximal value of $f'(t_f)$ with respect of (28) and the distance ($f(t_f) = 1$). We can use one quarter of a ternary polynomial to define $f(x)$ (Fig. 4).

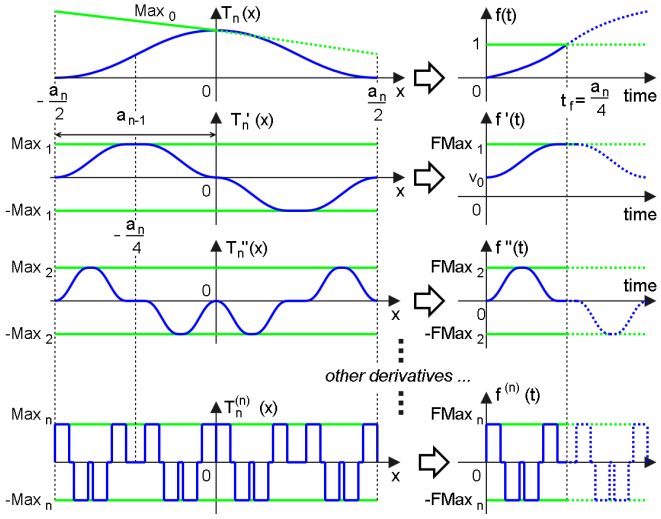

Fig. 4. Computing maximum speed with Ternary Polynomial.

Since $v_0 = f'(0)$ is not null we have to redefine the speed constraint: $Max_1 = FMax_1 - v_0$ and to report its influence on Max_0: $Max_0 = 2 - V_0 \cdot (x + a_n)$. We can then redefine the polynomial constraint $P_{i:j}$ (23) for $j = n - i$ with:

$$Pm_{i:j}(x) = x^j - a_{i-1} \cdot x^{j-1} - \frac{2 - v_0 \cdot 2^{j-1} \cdot x}{2^{\frac{(j-2)(j-1)}{2}} \cdot Max_n \prod_{l=0}^{i-1} (a_l - a_{l-1})} \quad (29)$$

The root of $Pm_{i:j}(x)$ is still in the interval (26): $[2 \cdot a_{i-1}, max_{r_{i:j}}]$.

Then we can compute the Ternary Polynomial $T_n(x)$ with the same method than in §III with:

- $Max_i = FMax_i$ for $i > 1$
- $Max_1 = FMax_1 - v_0$
- We use $Pm_{i:j}(x)$ for $j = n - i$

We obtain the optimal speed profile (maximum $f'(t_f)$ and minimum t_f) with:

$$f(t) = T_n\left(t - \frac{a_n}{2}\right) + v_0 \cdot t \text{ on } [0, t_f] \text{ and } t_f = \frac{a_n}{4} \quad (30)$$

B. "Optimal" speed profile

The optimal speed profile is the the one that minimizes t_f with respect of constraints (28) and input values $f^{(k)}(0)$ and $f^{(k)}(tf)$: $f^{(k)}(0) = f^{(k)}(tf) = 0$ for $k \in [\![2, n]\!]$, $f'(0) = v_0$, $f'(t_f) = v_f$, $f(0) = 0$ and $f(tf) = 1$. We assume that $v_0 \leq MaxStartSpeed(v_f)$ and $v_f \leq MaxEndSpeed(v_0)$ (if not there is no solution).

We will present briefly two methods to compute $f(t)$: analytic and recursive methods. Both methods are based on two "ternary polynomials": one for the acceleration phase $AT_n(x)$ and one for the deceleration phase $DT_n(x)$.

1) Analytic method.: The analytic method has two steps (Fig. 5). The first step is to find the key constraint. The second to solve it. The key constraint is the constraint that invalidates the distance $f(tf) \leq 1$.

$i \Leftarrow j \Leftarrow$ -1 and $A_{dist} \Leftarrow D_{dist} \Leftarrow 0$
repeat
  Compute A_{vmax} and D_{vmax}
  if $A_{vmax} > D_{vmax}$ **then**
    $j \Leftarrow j + 1$
    Compute d_j for $DT_{n-1}(x)$.
    Compute D_{vmax} and D_{dist}
  else
    $i \Leftarrow i + 1$
    Compute a_i for $AT_{n-1}(x)$.
    Compute A_{vmax} and A_{dist}
  end if
until (i==n and j==n) or ($A_{dist} + D_{dist} \geq 1$)
Solve key constraints on i, j
Compute $f(t)$

Fig. 5. Analytic method algorithm.

In the first step we compute the parameters a_i of the acceleration ternary polynomial ($AT_{n-1}(x)$) and d_j for the deceleration ternary polynomial ($DT_{n-1}(x)$). Those constraints are identical $Max_k = FMax_{k+1}$ beside Max_0 that is $FMax_1 - v_0$ for $AT_{n-1}(x)$ and $FMax_1 - v_f$ for $DT_{n-1}(x)$. We alternate the computation of a_i and d_j until we invalidate the distance constraint (Fig. 5):

$$\begin{aligned} A_k(m) &= FMax_n \cdot 2^{\frac{(m-1)(m-2)}{2}} \prod_{l=0}^{i} (a_l - a_{l-1}) \\ D_k(m) &= FMax_n \cdot 2^{\frac{(m-1)(m-2)}{2}} \prod_{l=0}^{j} (d_l - d_{l-1}) \\ A_{vmax} &= v_0 + a_i^{n-i-2} \cdot A_k(n - i - 1) \quad (31) \\ D_{vmax} &= v_f + d_j^{n-j-2} \cdot D_k(n - j - 1) \quad (32) \\ A_{dist} &= \frac{v_0 \cdot a_i \cdot 2^{n-i-1} + a_i^{n-i-1} \cdot A_k(n-i)}{2} \quad (33) \\ D_{dist} &= \frac{v_f \cdot d_j \cdot 2^{n-j-1} + d_j^{n-j-1} \cdot D_k(n-j)}{2} \quad (34) \end{aligned}$$

In the second step, to solve the key constraints we must solve the system of 2 equations in a_i and d_j: $A_{vmax} = D_{vmax}$ and $A_{dist} + D_{dist} = 1$. This system is equivalent to a

polynomial in a_i (respectively d_j) of degree $(n-j-1)(n-i)$ (respectively $(n-i-1)(n-j)$). So the maximal polynomial degree for a given number n of constraints is $n \cdot (n-1)$.

This algorithm can be time consuming. However, we must note that for $n = 3$ (jerk constraints) we have a simplification and only need to solve a polynomial of degree 4.

After this step we have $a_k = 2^{k-i} \cdot a_i$ and $d_l = 2^{l-j} \cdot d_j$ for $k > i$ and $l > j$. The speed profile is defined.

2) Iterative method.: The iterative method tries to avoid polynomial roots computation of the previous method by iteratively reducing the constant speed time D_t.

It is also an anytime algorithm (if interrupted it gives an aproximative solution).

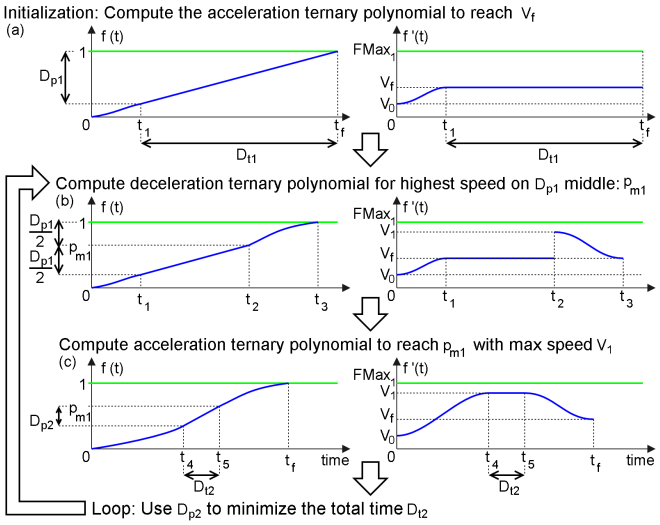

Fig. 6. Computing maximum speed with Ternary Polynomial.

Let's suppose that $v_0 < v_f$. We use $MaxEndSpeed$ algorithm to compute the speed profile with maximum speed v_f (Fig. 6.a). If we reach v_f before t_f we can optimize the speed profile on the distance $D_{p\,1}$.

We take the $D_{p\,1}$ middle ($p_{m\,1}$) and try to find the highest possible speed V_1 from v_f ($MaxStartSpeed$) on the distance $1 - p_{m\,1}$ (Fig. 6.b).

Now we have to recompute the acceleration phase starting from v_0 to reach V_1 on $p_{m\,1}$ (Fig. 6.c). Then again it is possible to have a delta time $D_{t\,2}$ with constant speed V_1 that can be optimized. We can do again the last two steps (Fig. 6.b and c) to obtain the "optimal" speed profile.

V. SPATIAL SMOOTHING

In the previous section we have shown an optimal temporal direct linear path. But a trajectory composed by them need to stop between them to deal with the direction discontinuity (as in §II). To avoid this, we can smooth the path.

In [2], the smoothing phase is based on the convolution of two consecutive direct paths with an exponential kernel:

$$k(x) = \frac{e^{-\frac{1}{1-x^2}}}{c} \tag{35}$$

The resulting smoothed path is infinitely derivable. But this method has two drawbacks:

- No analytic solution, the resulting trajectory configuration must be computed by integration that can be time consuming.
- No optimality when trying to limit the bounds of the trajectory derivatives.

Here we present how to solve those two problems by using a "ternary polynomial" as a kernel function.

We compute the "optimal" kernel that satisfies the constraints (6) and minimizes the total time with a linear speed profile:

$$f(t) = \frac{t}{t_f} \tag{36}$$

A. Smoothed trajectory computation.

The input of this algorithm is a set of three configurations: q_{i-1}, q_i and q_{i+1}. We want to have the direct trajectory with:

$$traj(0) = q_{i-1} \qquad traj(t_f) = q_{i+1} \tag{37}$$

$$traj'(0) = K \cdot (q_i - q_{i-1}) \qquad traj'(t_f) = K \cdot (q_{i+1} - q_i) \tag{38}$$

$$traj^{(k)}(0) = traj^{(k)}(t_f) = \vec{0} \text{ for } k \in [\![2, n-1]\!] \tag{39}$$

$$|traj^{(k)}(x)|_k \leq M_k \text{ for } k \in [\![1, n]\!] \tag{40}$$

First we must remark that a convolution by any symmetric kernel $k(x)$ on a path defined by:

$$\begin{aligned} p(x) &= (1+x) \cdot q_i - x \cdot q_{i-1} \quad \text{for } x < 0 \\ p(x) &= (1-x) \cdot q_i + x \cdot q_{i+1} \quad \text{for } x > 0, \end{aligned} \tag{41}$$

has the following properties:

$$(p * k)(-1) = q_{i-1} \cdot \int_{-\infty}^{\infty} k(x) \cdot dx \tag{42}$$

$$(p * k)(1) = q_{i+1} \cdot \int_{-\infty}^{\infty} k(x) \cdot dx \tag{43}$$

$$\begin{aligned} (p * k)'(s) = {} & (q_{i+1} - q_i) \cdot \int_{-\infty}^{s} k(x) \cdot dx \\ & + (q_i - q_{i-1}) \cdot \int_{s}^{\infty} k(x) \cdot dx \end{aligned} \tag{44}$$

$$(p * k)''(s) = (q_{i+1} - 2 \cdot q_i + q_{i-1}) \cdot k(s)\ . \tag{45}$$

So we have a normalization constraint on $k(x)$ (42,43):

$$\int_{-\infty}^{\infty} k(x) \cdot dx = 1 \tag{46}$$

We can define the path by: $path(p) = (p * k)(2 \cdot p - 1)$ and verify (37, 38). If we took for kernel $k(x)$ the function:

$$k(x) = \left(\frac{a_{n-2}}{2}\right)^2 \cdot T_{n-2}\left(\frac{a_{n-2}}{2} \cdot x\right) \tag{47}$$

Then if we take $t_f = a_{n-2}$ the constraints (46, 39, 40) become:

$$\int_{-\infty}^{+\infty} T_{n-2}(x) \cdot dx = \frac{2}{a_{n-2}} \tag{48}$$

$$T_{n-2}^{(k)}\left(-\frac{a_{n-2}}{2}\right) = T_{n-2}^{(k)}\left(\frac{a_{n-2}}{2}\right) = 0 \text{ for } k \in [\![0, n-3]\!] \tag{49}$$

$$|T_{n-2}^{(k)}(x)| \leq \frac{M_{k+2}}{|q_{i+1} - 2 \cdot q_i + q_{i-1}|_{k+2}} \text{ for } k \in [\![0, n-2]\!] \tag{50}$$

$$\frac{2}{t_f}\left|(p*k)'\left(2\cdot f(t)-1\right)\right|_1 \leq M_1 \quad (51)$$

The constraints (49, 50) are "ternary polynomial" properties. Since $k(x)$ is a symmetric positive kernel, the constraint (51) is equivalent to:

$$t_f \geq \frac{2}{M_1}\cdot \max\left(|q_{i+1}-q_i|_1, |q_i-q_{i-1}|_1\right), \quad (52)$$

The constraint (48) is equivalent to:

$$T_n(0) = 2 \quad (53)$$

Then to find the "optimal" kernel we have to compute the "ternary polynomial" that satisfies the constraints:

$$Max_0 = 2 \quad (54)$$

$$Max_k = \frac{M_k}{|q_{i+1}-2\cdot q_i+q_{i-1}|_k} \text{ for } k\in[\![2,n]\!] \quad (55)$$

We have no constraint on Max_1. Once we have $T_n(x)$ we can compute $k(x)$ and an explicit function for $path(p)$.

B. Temporal direct path

To define $f(t)$ we need t_f (36). By construction of $T_n(x)$ we have $t_f \geq a_{n-2}$. With the constraint (52) we obtain:

$$t_f \geq t_{f\,min} = \max\left(\frac{2\cdot|q_{i+1}-q_i|_1}{M_1}, \frac{2\cdot|q_i-q_{i-1}|_1}{M_1}, a_{n-2}\right) \quad (56)$$

Then the "temporal direct path" properties are:

$$MaxStartSpeed(v) = \min\left(v, 1/t_{f\,min}\right) \quad (57)$$

$$MaxEndSpeed(v) = \min\left(v, 1/t_{f\,min}\right) \quad (58)$$

$$f(t) = v_0\cdot t = v_f\cdot t \quad (59)$$

C. Results

Fig. 7 shows a speed profile computed on a path and on its smoothed version, with only two parameters x, y.

Constraints are defined by a pondered norm max. As we can see both trajectories respect all constraints (in this example until jerk derivative to have jerk continuity). We can see that the original path needs to stop between direct paths. The total time needed to go through the smoothed path is then shortened.

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

We have presented several generic algorithms to compute the speed profile along an existing path and even to smooth the path in order to increase its performances.

The general algorithm can be used with any direct path that respect the "temporal direct path" properties. Thanks to the "ternary polynomial" we have been able to define the "optimal" speed profile for direct linear path with any number of constraints on the trajectory derivatives (6).

In the same way we have been able to build an "optimal" smoothed path that satisfies the given constraints (6).

We have succeeded into building method to create a speed profile along a whole path with constraints that can be on speed, acceleration, jerk or any other derivatives of the trajectory.

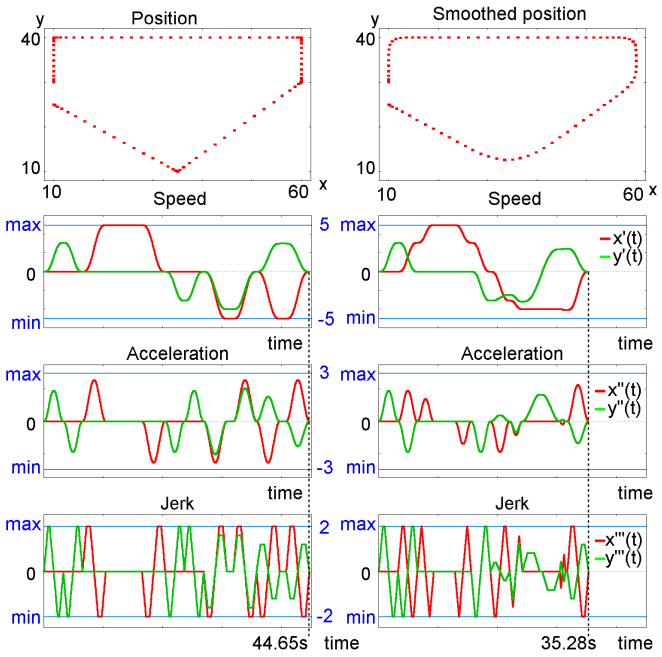

Fig. 7. Speed profile along a smoothed and not smoothed path.

B. Future Works

Future works can include two aspects: to avoid null acceleration between direct path, and to use constraints that cannot be expressed by (6). Both problems face a very high complexity.

As we have seen for the "temporal direct linear path" computation, adding a delta-speed ($f'(0) \neq f'(t_f)$) implies n^2 polynomial. The complexity of the speed profile computation is exponential with the number of free parameters. Adding delta-acceleration ($f''(0) \neq f''(t_f)$) can be prohibitive.

The second problem is very interesting for real application, especially for robot arm manipulator. In this case we often want to limit speed, acceleration (and some time jerk) of actuators but also of the end effector. End effector motion constraints cannot be expressed in our formalism (6). A new formalism must be studied.

REFERENCES

[1] C. Guarino Lo Bianco and M. Romano, "Bounded velocity planning for autonomous vehicles", *IEEE Int. Conference on Intelligent Robots and Systems*, Alberta, Canada, Aug. 2005, pp. 4068-4073

[2] Y. Hirano, K. Kitahama, S. Yoshizawa, "Image-based Object Recognition and Dexterous Hand/Arm Motion Planning Using RRTs for Grasping in Cluttered Scene", *IEEE Int. Conference on Intelligent Robots and Systems*, 2005

[3] J. Kuffner and Steven LaValle, "Rrt-connect : An efficient approach to single-query path planning", *IEEE Int. Conference on Robotics and Automation*, 2000

[4] J. Minguez and L. Montesano and L. Montano, "An architecture for sensor based navigation in realistic dynamic and troublesome scenarios", *IEEE Int. Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep. 2004

[5] J.Pettre and J. P. Laumond and T. Simeon, "A 2-stages locomotion planner for digital actors", *Eurographics Symposium on Computer Animation*, San Diego, USA, Jul. 2003, pp. 258-264