**Workshop on Robot Motion Planning:**
**Online, Reactive, and in Real-time**
2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems, IROS 2012
Vilamoura, Algarve, Portugal, October 7-12, 2012

# Online task space trajectory generation

Daniel Sidobre[1] and Wuwei He[2]

***Abstract*— As the kinematic of the robots becomes complex and the task to realize are more and more demanding, we need tools to better define and manipulate the movements of the robots. To cope with this problem, we propose a family of trajectory, which we name the Soft Motion trajectories, defined by polynomial functions of degree three. Based on these trajectories we propose a set of tools to generate trajectories and control the robots. We present some experimental results showing the interest of the approach that unify the data exchanged from the planning level to the control level of the robot.**

## I. Introduction

As machines become more and more complex and precise, they can not settle for following a path, they need better defined moves to design and think motions. The concept of trajectories that defines the position as a function of times allows building more powerful tools to animate machines. Some systems already use trajectories, but not in an integrated way from planning to control. This paper focuses on trajectories defined as series of segments of polynomial function of degree three and proposes a set of tools to generate and manipulate them.

The classical approach utilized by most of machines consists in defining a path and expect that the system can follow it. Unfortunately, many of these paths cannot be followed efficiently and precisely. For instance some paths are defined as polygonal lines that require the system stops at each vertices or approximates the lines around the vertices. It is well known that the path must be at least of class $C^2$ to be feasible. But this smoothness condition is practically not sufficient as the maximal speed depends on the local radius of curvature of the path.

The use of a trajectory to define a move gives all the necessary elements to verify that the move is feasible. Using a dynamic simulator, all the physical characteristic of the move can be verified: collisions, maximum velocity, maximum power, maximum torque etc.

From a control point of view, trajectories are also very interesting because they allow simpler control strategies. Torsten Kroeger showed the possibility to switch very easily between different controller [1]. For robot interacting with humans, trajectories allow to express easily the safety and comfort constraints as kinematic limits. Thanks to advances in computers sciences, the trajectories can now be manipulated very efficiently.

This paper is organized as follows. Trajectory generality are presented in section II. A set of trajectory generators are described in section III. The section IV details a method to approximate trajectories with polynomial third degree trajectories. A solution to generate trajectories from polygonal lines is described in section V. A trajectory controller is presented in the section VI. Experimental results are presented in section VII. Finally we give some concluding remarks in section VIII.

## II. Trajectories

To clarify the subject, we first introduce trajectories and give their main properties. Then, we detail the model of trajectories based on series of cubic polynomial functions and introduce different tools to manipulate them.

Trajectories are time functions defined in geometrical spaces, like essentially Cartesian space and joint space. The rotations can be described using different coordinates system: quaternion, vector and angle etc. The books from Biagiotti [2] on one hand and the one from Kroger [1] on the other hand summarize background trajectory material.

Given a system whose position is defined by a set of coordinate $X$ if the coordinates are in Cartesian space or $Q$ if the coordinates are in joint space, a trajectory $\mathcal{T}$ is a function of time defined as:

$$\mathcal{T} : [t_I, t_F] \longrightarrow \mathbb{R}^n \tag{1}$$

$$t \longmapsto \mathcal{T}(t) = X(t) \tag{2}$$

The trajectory is defined from the time interval $[t_I, t_F]$ to $\mathbb{R}^n$ where $n$ is the dimension of the motion space. The $\mathcal{T}(t)$ function can be a direct function of time or the composition $\mathcal{C}(s(t))$ of a function giving the path $\mathcal{C}(s)$ and a function $s(t)$ describing the time evolution along this path.

At first glance the latter offer more possibilities as the time evolution is independent of the geometrical path and so the two elements can be modified independently. Unfortunately, this approach is limited by the difficulty to integrate the derivative of the path to obtain the curvilinear abscissa. Without this parameterization, the function $s(t)$ doesn't give the tangential velocity and the kinematic of the motion is difficult to manipulate and interpret. So, as the former has a simpler expression, it provides simpler solutions to define and manipulate trajectories.

A trajectory $\mathcal{T}(t)$ defined from $t_I$ to $t_F$ can be defined by a series of trajectories defined between intermediate points. Given, $t_u$ which satisfies $t_I < t_u < t_F$, an equivalent

This work has been supported by the European Community's Seventh Framework Program FP7/2007-2013 "SAPHARI under grant agreement no. 287513 and by the French National Research Agency project ANR-07-ROBO-0011 "ASSIST" and ANR-10-CORD-0025 "ICARO".

[1]D. Sidobre is with CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France; and Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France daniel.sidobre@laas.fr

[2]W. He is with CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France Wuwei.He@laas.fr

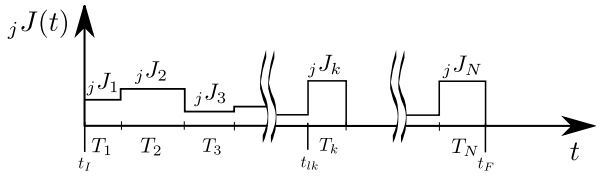


Fig. 1. Jerk profile for a series of segments of cubic polynomial trajectory

representation of $\mathcal{T}(t)$ is defined by the series of two trajectories $T_1$ and $T_2$ defined respectively by :

$$\begin{array}{ll} \mathcal{T}_1 : [t_I, t_u] \longrightarrow \mathbb{R}^n & \mathcal{T}_2 : [t_u, t_F] \longrightarrow \mathbb{R}^n \\ t \longmapsto \mathcal{T}_1(t) = \mathcal{T}(t) & t \longmapsto \mathcal{T}_2(t) = \mathcal{T}(t) \end{array} \tag{3}$$

Similarly a trajectory can be defined by a series of sub-trajectories if some continuity criteria specified for the trajectory and its derivative are verified. Generally this criterion is defined as a differentiability class $C^k$ with $k \geq 2$.

The possible choices to define trajectory functions are very large, but as we intend to compute motions in real time, we choose a simple solution like polynomial functions. As we need $C^2$ functions, we choose polynomial function of third degree and name this trajectories Soft Motion trajectories. Using a long series of polynomial function, trajectories following very complex path can be defined. It is also possible to approximate or interpolate a set of points to define Soft Motions trajectories.

In the sequel, we firstly present Soft Motion trajectories and then a set of consistent trajectory generator to solve robotic problems.

### *Series of $3^{rd}$ degree polynomial trajectories*

We define Soft Motion trajectories as series of $3^{rd}$ degree polynomial trajectories. Such a trajectory is composed of a vector of one-dimensional trajectories: $\mathcal{T}(t) = ({}_1Q(t), {}_2Q(t), \cdots, {}_nQ(t))^T$ for joint motions or $\mathcal{T}(t) = ({}_1X(t), {}_2X(t), \ldots, {}_nX(t))^T$ in Cartesian space. Without loss of generality, we suppose that all ${}_jX(t)$ or all ${}_jQ(t)$, $0 \leq j < n$ share the same time intervals and that $t_I = 0$. A one dimensioned trajectory ${}_jX(t)$ is defined by its initial conditions (${}_jX(0) = {}_jX_I$, ${}_jV(0) = {}_jV_I$ and ${}_jA(0) = {}_jA_I$) and $K$ elementary trajectories ${}_jX_i(t)$ defined by the jerk ${}_jJ_i$ and the duration $T_i$ where $1 \leq i \leq K$ and $\sum_{i=1}^{K} T_i = t_F - t_I$. By integration we can define the acceleration ${}_jA(t)$, the velocity ${}_jV(t)$ and then the position ${}_jX_i(t)$.

Assuming $k \leq K$ is such that $\sum_{i=1}^{k-1} T_i \leq t < \sum_{i=1}^{k} T_i$, the trajectory ${}_jX(t)$ and its derivative are defined by :

$${}_jJ(t) = {}_jJ_k \tag{4}$$

$${}_jA(t) = {}_jJ_k\left(t - \sum_{i=1}^{k-1} T_i\right) + \sum_{l=1}^{k-1} {}_jJ_l\ T_l + {}_jA_I \tag{5}$$

$${}_jV(t) = \frac{{}_jJ_k}{2}\left(t - \sum_{i=1}^{k-1} T_i\right)^2 + \sum_{l=1}^{k-1} {}_jJ_l\ T_l\left(t - \sum_{i=1}^{l} T_i\right) + \sum_{l=1}^{k-1} \frac{{}_jJ_l\ T_l^2}{2} + {}_jA_I t + {}_jV_I \tag{6}$$

$${}_jX(t) = \frac{{}_jJ_k}{6}\left(t - \sum_{i=1}^{k-1} T_i\right)^3 + \sum_{l=1}^{k-1} \frac{{}_jJ_l\ T_l}{2}\left(t - \sum_{i=1}^{l} T_i\right)^2 + \sum_{l=1}^{k-1} \frac{{}_jJ_l\ T_l^2}{2}\left(t - \sum_{i=1}^{l} T_i\right) + \sum_{l=1}^{k-1} \frac{{}_jJ_l\ T_l^3}{6} + \frac{{}_jA_I}{2}t^2 + {}_jV_I t + {}_jX_I \tag{7}$$

This general expression of the trajectories and their derivatives can be used directly to control a arm, for example, but it is not easy to obtain directly. So we will now describe different generators to build them.

## III. Trajectory generators

As different motion problems exist, we need a coherent set of trajectory generator. To classify these trajectory generators we use the different types of motions we wish to define Soft Motions for. The first one is the point-to-point motion that can be done in minimum time or in an imposed time. A point-to-point move is a move where the mobile starts from rest and stops after the move. A more general problem is defined between two general situations; in this case the initial and final conditions are arbitrary. The motions can also be defined by a set of via point to approximate or interpolate. A very interesting problem is to approximate any trajectory by a Soft Motion one. Finally Soft Motions can be classified by the dimension of the motion space.

In the following sections, we present generators for each of these trajectory problems, beginning by the simpler to build the more complex.

### *A. One-dimensional generator*

To cope with system physical limits using $3^{rd}$ degree polynomial functions, we can limit the jerk, the acceleration and the velocity:

$$-J_{max} \leq J(t) \leq J_{max} \tag{8}$$

$$-A_{max} \leq A(t) \leq A_{max} \tag{9}$$

$$-V_{max} \leq V(t) \leq V_{max} \tag{10}$$

This limits define a domain for the one dimensional systems presented in the figure 2 bottom using the Acceleration-Velocity frame. In this diagram, motions with constant jerk draw parabolas.

A canonical generation of trajectory problem is defined in this domain by initial and final conditions:

$$X(t_I) = X_I \quad V(t_I) = V_I \quad A(t_I) = A_I \tag{11}$$

$$X(t_F) = X_F \quad V(t_F = V_F \quad A(t_F) = A_F \tag{12}$$

The quasi-optimal solution in minimum time to this problem is presented in [3], [4]. The well known canonical case of long point-to-point motion is depicted in figure 2. In this

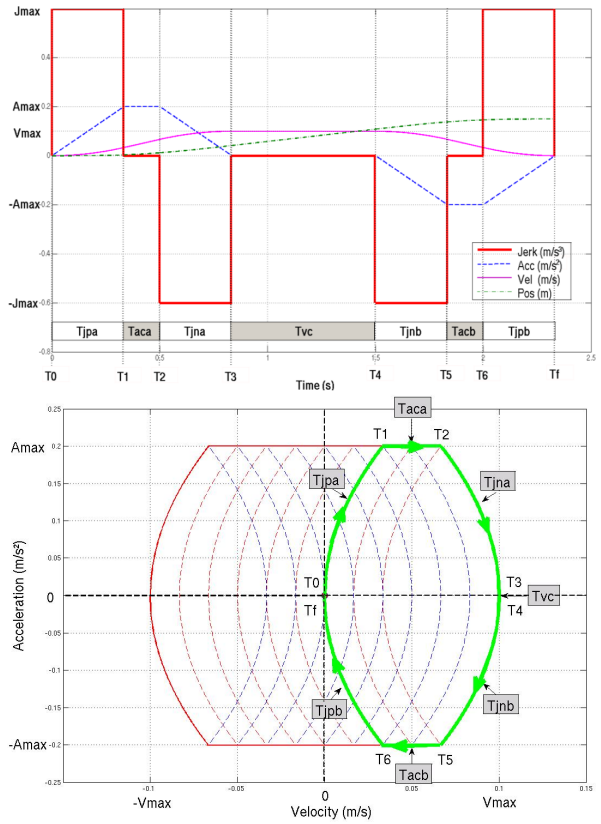


Fig. 2. Top: Position, velocity, acceleration and jerk for a point-to-point move in function of time. Bottom: the same move in the frame Acceleration-Velocity with the bounds of the validity domain.

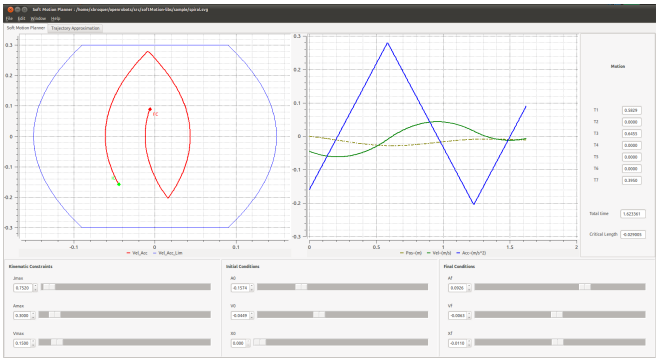

Fig. 3. Graphic user interface of the Soft Motion planner displaying a general short motion.

case the motion is composed of seven segments of $3^{rd}$ degree polynomial functions. This is the maximum number of segments as either the point with maximal velocity or with minimal velocity can be reached with three segments from any situation and similarly to return to any situation. From a computation point of view, the figure 3 exhibit the worse case where the intersection of three parabola has to be computed. This case generates a polynomial equation of degree 6 numerically solved by Raphson-Newton method.

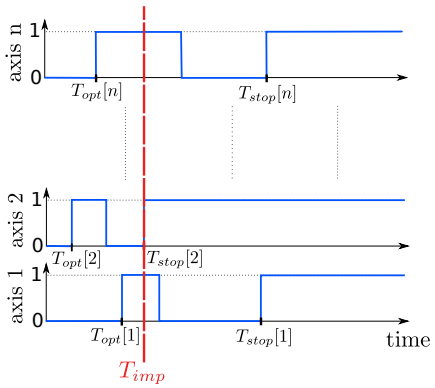


Fig. 4. Valid interval of time for a set of axes and the time $T_{imp}$ corresponding to the move in minimum time.

### B. N-dimensional point-to-point generator

The generation of trajectories in multidimensional spaces is far more complex as we will see in the next section, but the point-to-point motion is a particular case that can be bring back to a unidimensional problem. Suppose the line joining the initial and final points in $\mathbb{R}^N$ is defined relatively to a basis $\{v_i\}_{0<i\leq N}$ by a vector $v = \sum_{i=1}^{N} \alpha_i v_i$ with $\sum_{i=1}^{N} \alpha_i^2 = 1$. The velocity, the acceleration and the jerk are respectively limited for each axis $i$ by $V_{iM}$, $A_{iM}$ and $J_{iM}$.

The minimum time trajectory is directly obtained by projecting on each axis the solution of the one-dimensional problem defined on the line segment by the limits [4]:

$$J_{\max} = \min_{1\leqslant i\leqslant N} \frac{1}{\alpha_i} J_{iM} \tag{13}$$

$$A_{\max} = \min_{1\leqslant i\leqslant N} \frac{1}{\alpha_i} A_{iM} \tag{14}$$

$$V_{\max} = \min_{1\leqslant i\leqslant N} \frac{1}{\alpha_i} V_{iM} \tag{15}$$

For each segment of the trajectory, one of the velocity acceleration, or jerk functions of one of the $N$ initial axes is saturated. The others are inside their validity domain.

### C. N-dimensional general generator

In this case, initial and final velocity and acceleration are no longer zero and the problem can no longer be linearized. In a first step, we compute the minimum time movement using the method of the paragraph III-A for each axis and select the longest one $T_{min} = \max_{1\leq i\leq n} T_{opt}i$. The minimum time movement for the move cannot be lower than this time $T_{min}$. In some case, it is possible to compute for each of the other axes a move in this time $T_{min}$. Unfortunately the minimum time for the move can be larger than $T_{min}$ as it is not always possible to increase the time of a motion for all values. For example, suppose a one axis mobile moving at $V_{max}$ during a short time $t_i$ so it travels a distance of $X_i$. Therefore, we wish to increase $t_i$ of $\delta_t$. For some $\delta_t = \delta_l$ we obtain a limit case where the movement is composed of two segments, the first with the jerk $-J_{max}$ and the last with

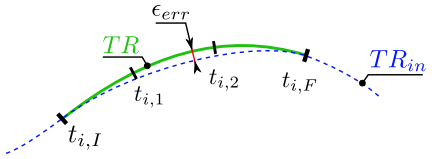


Fig. 5. The initial trajectory $\mathcal{T}_{in}$ and the approximated $\mathcal{T}$.

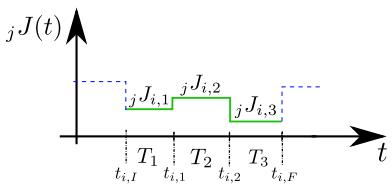


Fig. 6. Jerk profile for the axis $j$ of the trajectory $\mathcal{T}$.

$J_{max}$. For a $\delta_t < \delta_l$ an infinity of solution exists to do the move in $t_i + \delta_t$, but for $\delta_l < \delta_t < \delta_u$ there is no solution. $\delta_u$ corresponds to the necessary time to the mobile to stop going beyond the final point, to go back and return. As we choose an initial motion at $V_{max}$ it is not possible to do the move in a time less than $t_i$. Figure 4 shows the choice of the minimum valid time $T_{imp}$ for a set of axes. In the case of the picture, the minimum time for each axis is $T_{opt}$ of the first axis and the minimal time feasible is $T_{stop}$ of the second axis.

So we can determine the minimum time $T_{imp}$ for an N-dimensional move. But an infinity of solutions exists. Our system proposes one solution, but an optimum criterion is still to build hoping it gives simple computations. The shape of the path defining the trajectory depends on this choice. For motion planning in presence of obstacles this choice has an important influence.

Now we have a solution to generate a trajectory to define a move between two situations. In the following we introduce generators that master the shape of the trajectory between the initial and final situations.

## IV. Trajectory approximation:

We now wish to define any motion with a set of polynomial trajectories of third degree. To do this, we propose to approximate any trajectory by a Soft Motion trajectory.

Suppose $\mathcal{T}_{in}$ is an arbitrary trajectory defined, for example, by a path $P$ and a motion law $u = u(t)$. Both the path $P$ and the law $u$ can be defined by a large variety of curves (Bézier, NURBS, sinusoid etc.). If the differentiability class of this trajectory is at least $C^2$, a good approximation can be computed. But in case of discontinuity, we must accept a higher error. This error can be balanced between a geometric error and a time error. In case of geometric errors, the initial and realized paths are different but outside these difficult zones the trajectory can be precisely realized. In case of time errors, the mobile can stop to stay on the path and ensure velocity and acceleration continuity, but such a modification introduces a delay for the remaining trajectory.

### A. The three segments method

If we consider a portion of the trajectory $\mathcal{T}_{in}$ defined by an initial instant $t_{i,I}$ and a final instant $t_{i,F}$, $\mathcal{T}_{in}$ defines the initial and final situations to approximate: $(X_I, V_I, A_I)$ and $(X_F, V_F, A_F)$.

An interesting solution to approximate this portion of trajectories is to define a sequence of three trajectory segments with constant jerk that bring the mobile from the initial situation to the final one in the time $T_{imp} = t_F - t_I$. We choose three segments because we need a small number of segments and there is no always solution with one or two segments.

The system to solve is then defined by 13 constraints : the initial and final situations (6 constraints), the continuity in position velocity and acceleration for the two switching situations and the time. Each segment of trajectory is defined by four parameters and one time. If we fix the three duration $T_1 = T_2 = T_3 = \frac{T_{imp}}{3}$, we obtain a system with 13 parameters where only the three jerks are unknown. As the final control system is periodic with period $T$, the times $T_{imp}/3$ must be a multiple of the period $T$ and $T_{imp}$ chosen to be a multiple of $3T$.

The 3 jerks are then defined by:

$$\begin{bmatrix} J_1 \\ J_2 \\ J_3 \end{bmatrix} = A^{-1}. \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{16}$$

with

$$A^{-1} = \frac{1}{T_{imp}} \begin{bmatrix} 1 & -9 & 27 \\ -7/2 & 27 & -54 \\ 11/2 & -18 & 27 \end{bmatrix} \tag{17}$$

and

$$B_1 = A_F - A_I$$
$$B_2 = V_F - V_I - A_I T_{imp}$$
$$B_3 = X_F - X_I - V_I T_{imp} - A_I \frac{T_{imp}^2}{2}$$

More details can be found in [5] and [4].

### B. Distance between trajectories:

An important characteristic of the approximation is the maximum error between the two trajectories. As several distances exist to compare trajectories, we choose the Hausdorff distance and the synchronous Euclidean distance. Another interesting measure of the difference is the synchronous Euclidean distance between the velocities.

The synchronous Euclidean distance is defined by:

$$d_{SE} = \max_{t \in [t_I, t_F]} \sqrt{\sum_{j=1}^{n} ({}_j\mathcal{T}(t) - {}_j\,\mathcal{T}_{in}(t))^2} \tag{18}$$

The synchronous euclidean distance between velocities is defined by:

$$d_{SEV} = \max_{t \in [t_I, t_F]} \sqrt{\sum_{j=1}^{n} \left( \frac{d_j\mathcal{T}(t)}{dt} - \frac{d_j\mathcal{T}_{in}(t)}{dt} \right)^2} \tag{19}$$

The Hausdorff distance is defined by:

$$d_{Haus} = \max( \sup_{t_{in}\in[t_I,t_F]} \inf_{t\in[t_I,t_F]} d(\mathcal{T}_{in}(t_{in}), \mathcal{T}(t)), \quad (20)$$

$$\sup_{t\in[t_I,t_F]} \inf_{t_{in}\in[t_I,t_F]} d(\mathcal{T}(t), \mathcal{T}_{in}(t_{in}))) \quad (21)$$

Depending on the type of problem, one of these distances is generally more appropriated. If the geometry of the path is important as for example for machining application, the Hausdorff distance is a good choice. For moves in free space, coordination is more important and so the synchronous Euclidean distance is suitable. The distance between velocities is more sensible to identify the variation due to the motion law.

### C. Error of approximation for a trajectory

We suppose now that $\mathcal{T}_{in}$ is bounded respectively in jerk, acceleration and velocity by $J_{max}$, $A_{max}$ and $V_{max}$. We show in this paragraph that a relation exists between the error of approximation, the time $T_{imp}$ and the bound $J_{max}$.

Let $\mathcal{V}_{in}$ and $\mathcal{A}_{in}$ denote respectively the velocity and acceleration of $\mathcal{T}_{in}$. In a first time, we examine the case where the trajectory $\mathcal{T}_{in}$ to approximate satisfies:

$$\mathcal{T}_{in}(t_I) = \mathcal{T}_{in}(t_F) = 0 \quad (22)$$

$$\mathcal{V}_{in}(t_I) = \mathcal{V}_{in}(t_F) = 0 \quad (23)$$

$$\mathcal{A}_{in}(t_I) = \mathcal{A}_{in}(t_F) = 0 \quad (24)$$

One can verify that this initial and final conditions gives three null jerks (See eq. 16).

The trajectory to approximate $\mathcal{T}_{in}$ that gives the maximum error is symmetric. As the trajectory to approximate $\mathcal{T}_{in}$ is kinematically bounded and due to the symmetry, the maximum error between the two trajectories is at the middle of the trajectory. Likewise, the maximum error is obtained for a saturated function. For a short trajectory, the acceleration is not saturated and the more difficult function to approximate is defined by the four segments trajectory:

$$T_1 = T_4 = T_{imp} * \frac{2-\sqrt{2}}{4} \quad (25)$$

$$T_2 = T_3 = T_{imp} * \frac{\sqrt{2}}{4} \quad (26)$$

and the jerks are $J_1 = J_3 = J_{max}$ $J_2 = j_4 = -J_{max}$.

The maximum error between the two trajectories is then:

$$\epsilon = \frac{\sqrt{2}-1}{48 * \sqrt{2}} = 0.0061 \times J_{max} * T_{imp}^3 \quad (27)$$

*General case:* Suppose $\mathcal{T}(t)$ is the approximation by the 3 segments method of the trajectory $\mathcal{T}_{in}(t)$between $t_I$ and $t_F$.

We can write the $\mathcal{T}_{in}(t)$ trajectory as $\mathcal{T}_{in}(t) = \mathcal{T}(t) + (\mathcal{T}_{in}(t) - \mathcal{T}(t))$

By design the trajectory $\mathcal{T}_0(t) = \mathcal{T}_{in}(t) - \mathcal{T}(t)$ verifies the conditions 22, 23, and 24.

So the approximation error of $\mathcal{T}_0(t)$ on $[T_I, T_F]$ by a trajectory composed of three segments of cubic polynomial trajectory is less than $0.0061 \times T_{imp}^3 * (2 \times J_{max})$.

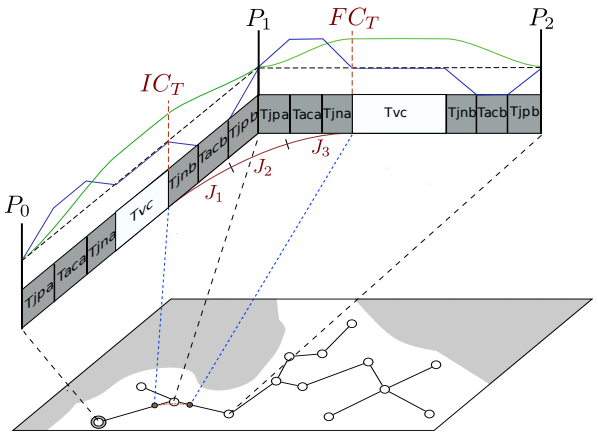


Fig. 7. From a polygonal path to a Soft Motion trajectory.

As $\mathcal{T}(t)$ is approximated without error, $\mathcal{T}_{in}(t)$ that is the sum $\mathcal{T}(t) + \mathcal{T}_0(t)$ can be approximated with an error less than: $0.0061 \times T_{imp}^3 \times (2 \times J_{max})$.

This result is extremely interesting as it gives the length of the interval to approximate a function while insuring the approximation error is smaller than a defined limit.

### D. Example of a circular trajectory:

To approximate a trajectory following a circle of radius $R$ at constant speed $\omega R$, we can compute the maximum time interval $T_{imp}$ to approximate the circle with a maximum error of $\epsilon$.

The trajectory of the motion is defined by:

$$X_x(t) = R \times cos(\omega t) \quad (28)$$

$$X_Y(t) = R \times sin(\omega t) \quad (29)$$

and the jerk by:

$$J_X(t) = \omega^3 R \times sin(\omega t) \quad (30)$$

$$J_Y(t) = -\omega^3 R \times cos(\omega t) \quad (31)$$

So the constant jerk is $\omega^3 R$ and the maximum time interval is then

$$T = \frac{\epsilon\sqrt{3}}{0.0061 \times 2 \times J} = \frac{\epsilon\sqrt{3}}{0.0061 \times 2 \times \omega^3 R} \quad (32)$$

For a mobile completing a turn in one second about a circle of radius $R = 0.1m$ with a maximum error of $\epsilon = 10^{-6}$, $T$ is $T = 0.0149s$ corresponding to 68 points (6 points for an error of $\epsilon = 10^{-3}$, $T$). This result can be used directly when radius of curvature is known and the path is traversed at constant speed.

## V. Generating trajectory from polygonal path

The main advantage of the Soft Motion trajectories is to insure a continuity of data and reasoning from the high planning level to the control level. Most of the motion planners as, for example RRT planners [6], [7], produce only paths in the form of polygonal lines. The assimilation of a path to a trajectory commonly performed at planning level is not acceptable for control. So these paths should be converted

in trajectories. We suppose the line segments of the path are relatively long after a path planner optimization phase and so the robot can reach the maximum velocity and stop to traverse each segment. Given a set of kinematic limits $\{J_{max}, A_{max}, V_{max}\}$, a trajectory stopping at each vertex is easily build using the linear generator of paragraph III-B.

In [5] we proposed to use a twofold strategy to smooth this trajectories. The first idea is to smooth the vertex of the polygonal line between the points where the robot must begin to decelerate ($IC_T$) and can stop to accelerate ($FC_T$) following precisely the path (see figure 7). Between the two situations $FC_T$ and $IC_T$ the 3 segments method gives a simple path. The second strategy take into account that the initial polygonal trajectory that stop at the vertex has no collision and can be used when the smoothed trajectory causes a collision.

The time to go from $FC_T$ to $IC_T$ is defined by the method presented in paragraph III-C. As we wish a continuous motion, we use the 3 segments method of the paragraph IV. The trajectory is then checked for collision.

This strategy to compute a smoothing segment of trajectory can be improved by optimizing the choice of the initial and final points defining the segment [4] as a shorter segment generating a smaller error is preferable in some situations. The trajectory computed by choosing initial and final points between $IC_T$ and $P_1$ and $P_1$ and $FC_T$ respectively is sometimes feasible.

## VI. A TRAJECTORY CONTROLLER

We suppose each axe of the mobile is equipped with a low level controller, for example a PID controller. This low level controller can be directly fed from a trajectory definition using the expressions 7 or 6.

In general, controllers can also get feedback from complex localization systems: a mobile manipulator robot which exchange an object with a human needs to localize itself, the human and the object. To obtain its position this robot can use different sensors (cameras, lasers, odometry) and localization techniques based on different sensors, different geometric elements and different filtering techniques. To localize the object, it can use stereovision or point cloud obtained from sensors like Kinect.

So, a general control problem can be defined by:

1) a frame in which the trajectory to follow, which is generated by high level, is defined.
2) the current position, velocity and acceleration of the mobile.
3) the current position, velocity and acceleration of the target.
4) a trajectory to be executed by the controller.

### A. The frame of a trajectory

Given a situation where a robot is supposed to grasp an object handed by the human. At the beginning of the task, a planner computes a trajectory for the robot. At this instant, the trajectory can be expressed in any moving or fixed frame equivalently. But after a short moment, because of the movement of humans and robots, the trajectories expressed in the alternative frames are different. For example, the end of the trajectory defines an approaching path defined to avoid collisions. This local path must be defined in a frame associated to the object, so that the gripper approaches the object along this path even though the object is moved by the human. In the same way, if the beginning of the trajectory is defined in the frame associated to the object and the human rotate a little the object at the beginning, the start of the trajectory can made a big move relatively to a fixed frame.

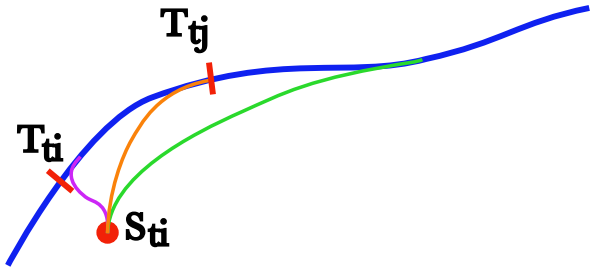


Fig. 8. Trajectory control: At instant $t_i$, the mobile should be at point $T_{t_i}$ on the blue trajectory, but it is in point $S_{t_i}$ because of the situation change. The orange trajectory reaches the blue at the instant $t_j$. The green trajectory reaches the blue trajectory in a longer time and the purple one in a shorter time.

So each part of a trajectory must be associated to a frame in which it must be controlled. The choice of the instant to switch the controller from one frame to another must also be defined. In the previous example, the system can switch from the robot base frame to the object frame when the gripper reaches some distance from the object. Eventually, it is possible to define an intermediate segment of trajectory controlled in a third frame, for example a frame associated to the human hand.

When a robot is very close to an obstacle, controlling the robot in a frame fixed to this obstacle is generally a good solution to minimize the uncertainty and limit the risk of collision.

In conclusion, the planner must generate a trajectory and associate a control frame to each part of the trajectory.

### B. Computing a control trajectory

Given a segment of trajectory that has to be controlled in some frame, we present now a control strategy to cope with the inherent uncertainty associated to the position. Because of the large position error associated to the base position, of the possibility to switch from a controller to another or of the possibility to switch from a sensor to another, the distance between the real position and the setting position can be large.

The control problem is illustrated in figure 8 where some mobile must follow the blue trajectory. At the instant $t_i$, the mobile should be in point $T_{t_i}$ but it is in $S_{t_i}$. Only the position is depicted on the figure but the system take also into account velocity and acceleration.

From this initial situation, the controller must compute a control trajectory that reaches the input trajectory as soon

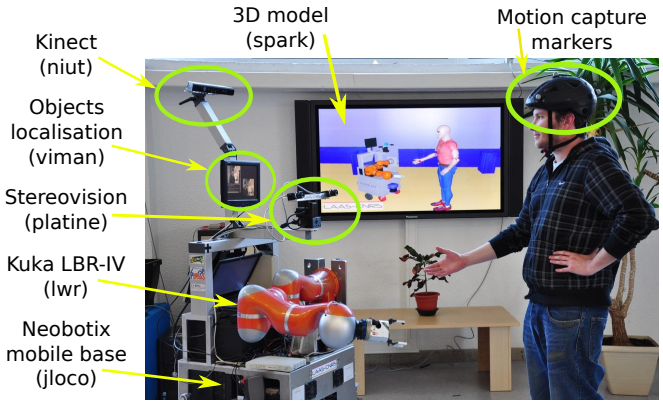


Fig. 9. A human interacting with the robot Jido and the main elements of the system.

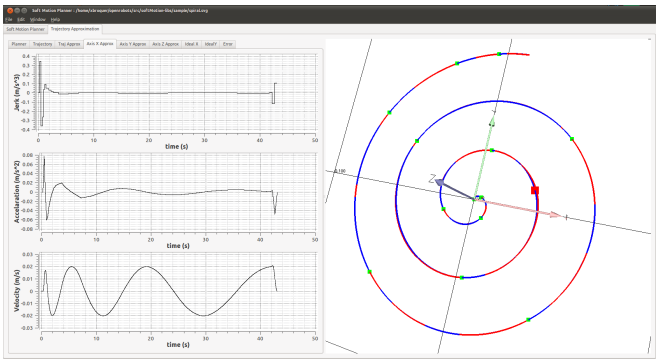

Fig. 10. A spiral trajectory approximated. Left the jerk, the acceleration and the velocity profiles.

as possible while complying with constraints. This can be done simply by computing a trajectory to reach the input trajectory for each instant $t_k$ for $k > i$ until a valid trajectory is obtained. The computed control trajectory is drawn in orange in the figure 8. Due to real time constraints, it must be necessary to define a more efficient strategy in case of important error and $k$ large.

The error between the real situation and the desired one at an instant can be due to many reasons. The strategy to build a control trajectory that reaches the input trajectory can be different in function of the problem. The previous solution is the solution when the objective is really the trajectory, but in some case the path is more important than the time and it can be preferable to choose a shorter path to minimize the Hausdorff error to the path. The purple trajectory of the figure 8 shows an example of such a trajectory. Of course in this case we accept a delay for the mobile. Later the mobile can or cannot catch up with this delay depending on the problem and conditions.

Similarly, a smoother trajectory could be preferable, the green trajectory of the figure 8 gives an example.

This control trajectory are computed with the three segments algorithm presented in section IV-A as the initial and final situations are precisely defined.

### C. Target tracking

Trajectory control can also be used in the absence of input trajectory. For example for the robot reach a relative position between the robot hand and an object, the local control trajectory can be directly defined between the current state of the robot hand (position, velocity, acceleration) and the future state of the target. The future state of the target is estimated assuming a continuous and regular motion.

In this case the time to reach the target is not imposed and the trajectory generator presented in section III-A is used. If we need that all the axes move synchronously, the method presented in section III-C can be used.

This approach can also be used at the end of a controlled trajectory move to maintain the relative position of the hand.

## VII. Experimental results

To illustrate the implementation of this tools based on Soft Motion trajectories, we present results carried out with our Jido robot. Jido is built up with a Neobotix mobile platform MP-L655 and a Kuka LWR-IV arm. Jido is equipped with one pair of stereo cameras and a Kinect motion sensor. The figure 9 describes the main elements of its architecture. To integrate the software, we use GENOM[1] [8], a development environment for complex real time embedded software, and robotpkg[2], a compilation framework and packaging system for installing robotics software.

To simplify the robot software, the different modules are organized around the SPARK module, which receives the data from all the software modules and builds a model of the scene. An example of this model is presented in the background of the figure 9. SPARK processes internal data from software modules that interface sensors and actuators (position of the arm and hand, position of the stereovision plate, odometer etc) and data about environment and humans (Kinect, motion capture, stereovision etc). The advantage of this centralized approach is the possibility for the module SPARK to compute accurate positions and kinetic parameters using different input data and filtering techniques. The robot is also equipped with a collision checker that verifies in line the risks of collision and stop the robot if necessary. In the actual implementation this module cannot take into account moving objects in the environment.

A human aware planner [9] computes a trajectory for the robot from the SPARK model and a description of the task to achieve. The use of position from SPARK can avoid some switch of controller input, but it introduces a delay to filter and fusion inputs.

The figure 10 shows the result of the approximation of a spiral trajectory. The original spiral was made using inkscape[3], so it is defined by a series of Bézier curves and a motion law defined as a one dimensional point-to-point trajectory. The error of approximation is depicted in figure 11.

[1] http://www.openrobots.org/wiki/genom
[2] http://homepages.laas.fr/mallet/robotpkg/
[3] http://inkscape.org/

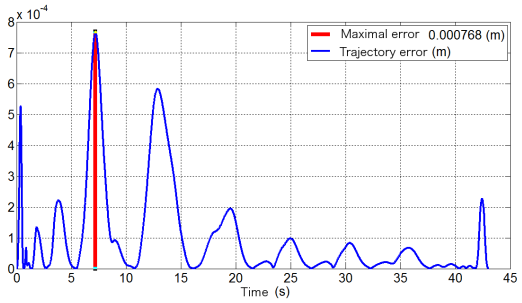


Fig. 11. The approximation error of the spiral curve.

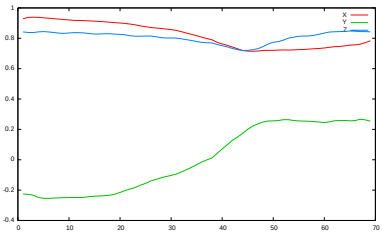

Fig. 12. A plot of the trajectory realized by the jido arm following a box handle by an human.

Figure 12 shows the trajectory executed by Jido during a task where the robot try to maintain its hand in a pre-defined position relative to a box. The trajectory was recorded when a human moved the box in front of the robot. The position of the box is obtained by stereovision tracking a tag plotted on the box.

The figure 13 shows Jido writing the word Dexmart. The trajectory was defined by a path drawn using inkscape and then approximated using the results of the section IV. The input trajectory is traveled at constant velocity. The Soft Motion approximation stops at cusp. The controller uses impedance control to maintain a constant force in the normal direction.

The figure 14 shows a trajectory build from a polygonal line to grasp an object.

A significant advantage of using trajectory controller is that the controller can work with different frequencies. For example, the position can be measured every 0.1s by vision and a low level controller that uses an impedance controller can run at 1kHz. This possibility simplifies the design of the controllers.

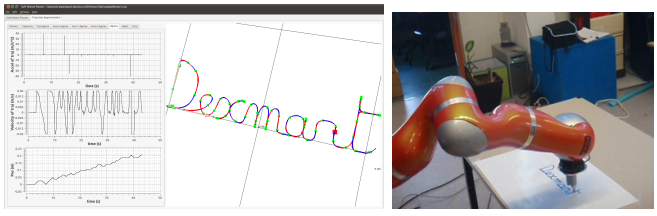

Fig. 13. Jido writing along a approximated trajectory.

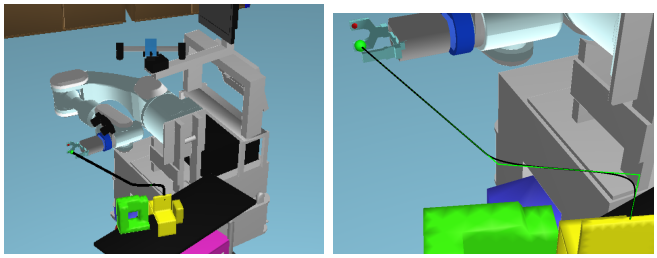

Fig. 14. Left: A simulated move to a gripper grasp an object. Right: The detail of the path defining the trajectory and the initial polygonal line.

## VIII. Conclusion

In this paper we have presented a set of trajectory tools to animate machines. A first interesting point is the proposal to use polynomial third degree trajectories (Soft Motions trajectories) because they define a simple and powerful class of trajectories. We proposed a set of trajectory generators that can be used to effectively build Soft Motions trajectories.

From these tools we described how to generate trajectories at planning level and then how to control a robot in several situations.

We can conclude with the hope to build a robot based on Soft Motion trajectories. This robot will embed a trajectory planner like the one we outlined but it should associate to each generated trajectories the frames in which the move must be controlled and the condition to switch between controllers. We have proposed a set of basic controllers; this set must be enlarged with force controllers. A tool to switch between these controllers and manage the state of this meta-controller should be defined and built.

Lastly, the coordination of the motions of a full robot (base, arms, hands, head) or of two or more robots is also very challenging, trajectories could help to synchronize this motions.

## References


[1] T. Kröger, *On-Line Trajectory Generation in Robotic Systems*, 1st ed., ser. Springer Tracts in Advanced Robotics. Berlin, Heidelberg, Germany: Springer, jan 2010, vol. 58.

[2] L. Biagiotti and C. Melchiorri, *Trajectory Planning for Automatic Machines and Robots*. Springer, 2008.

[3] X. Broquère, D. Sidobre, and I. Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *IEEE/RSJ Int. Conf. on Intel. Rob. And Sys.*, 2008.

[4] X. Broquère, "Planification de trajectoire pour la manipulation d'objets et l'interaction homme-robot," Ph.D. dissertation, LAAS-CNRS and Université de Toulouse, Paul Sabatier, 2011.

[5] X. Broquère and D. Sidobre, "From motion planning to trajectory control with bounded jerk for service manipulator robots," in *IEEE Int. Conf. Robot. And Autom.*, 2010.

[6] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, 2010.

[7] S. M. LaValle and J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Workshop on the Algorithmic Foundations of Robotics*, 2001.

[8] S. Fleury, M. Herrb, and R. Chatila, "Genom: A tool for the specification and the implementation of operating modules in a distributed robot architecture," in *IEEE/RSJ Int. Conf. on Intel. Rob. And Sys.*, 1997.

[9] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Siméon, "Human aware mobile robot motion planner," *IEEE Transactions on Robotics*, 2007.