2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

From Motion Planning to Trajectory Control with Bounded Jerk for Service Manipulator Robots

Xavier Broquère *(IEEE Member)*, Daniel Sidobre and Khoi Nguyen

***Abstract*— To build autonomous robots capable to plan and control tasks in human environments, we need a description of trajectories that allows the robot to reason on his moves. In this paper we propose to use series of cubic polynomial curves to define the trajectories with bounded jerk, acceleration and velocity. This solution is well adapted to plan safe and acceptable moves of the robot in the vicinity of humans. It is also a simple solution to approximate any trajectory and synchronize different robots or element of the robots. These curves have a simple representation, can be computed quickly and when used in a fitting algorithm can build controller.**

I. INTRODUCTION

Autonomous robots that operate in a human environment have to plan and execute safe moves. These moves can be defined by soft trajectories composed of a motion law and a smooth path defined in cartesian or articular space. At the planning level, tools like Move3D [1] or Human Aware Motion Planner HAMP [2] define paths that guarantee safety and comfort for humans [3] [4] and dynamic properties for the robot. HAMP is a path planner that takes explicitly into account the presence of humans in robot's environment. When an event occurs, an execution controller can ask the planner to modify the previously defined trajectory and ensure the continuity in acceleration, velocity and position. Using proprioceptive sensors, exteroceptive systems like vision systems and force sensors, the trajectory can be controlled at different levels.

In this paper we propose to use a single family of curves, sequence of cubic polynomial functions, to cover all the requirements of a robot in manipulating trajectories. Traditionally, path planners compute a path as a polygonal curve and then apply a method to smooth the curve. Sequence of cubic functions can be used to this purpose. In [5] we presented a general solution to compute a time optimal sequence of at most seven cubic polynomial curves that bound the velocity, the acceleration and the jerk to connect two states defined by position, velocity and acceleration. Approximation of any trajectory by a sequence of cubic polynomial curves is a well known problem and, in this paper, we propose a smart solution using sequences of three cubic polynomial curves. This new method extends our previous work [5] for the time imposed trajectory planning. Traditionally, the problem of robot control is divided in two hierarchical levels, the lower level called *control* or *path tracking* and the upper level called *trajectory planning* (Fig. 1). Control along a trajectory can be done by the real-time computation of a sequence of cubic polynomial curves that satisfy the constraints.

Xavier Broquère, Daniel Sidobre and Khoi Nguyen are with the Robotics and Interactive System Group at CNRS, LAAS, 7 Avenue du Colonel Roche, F-31077, Toulouse, France and with the University of Toulouse; UPS, INSA, INP, ISAE, LAAS, F-31077 Toulouse, France (`lastname@laas.fr`).

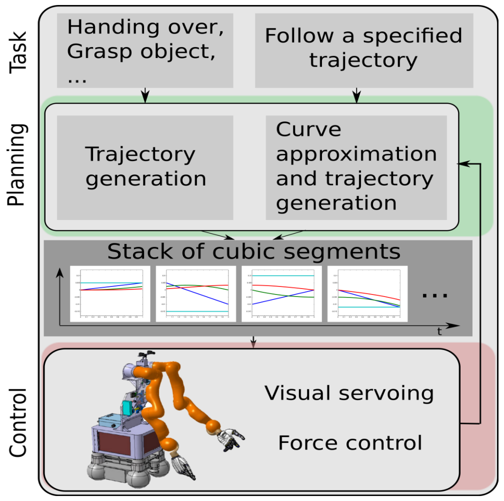
Task
Handing over, Grasp object, ...
Follow a specified trajectory
Planning
Trajectory generation
Curve approximation and trajectory generation
Stack of cubic segments
t
Control
Visual servoing
Force control

Fig. 1. The trajectory planning and control system

This paper focuses on the planning and control for a service manipulator robot. We present related work in Section II. Section III describes the trajectory planner in two parts, the trajectory approximation and then the optimal-time trajectory generation. We have implemented our planner into a mobile robot composed of a 6 DOF arm and then we explain how the arm is controlled in the Section IV. Two applications are presented in Section V.

II. RELATED WORK

The book of Biagiotti and Melchiorri [6] gives an overview on trajectory planning. The objectives of most trajectory planners are to improve tracking accuracy and reduce machine wear by providing continuous references to the servo-motor control. Many approaches have been proposed to plan smooth trajectories, most of them are related to production machines in off-line [7] or on-line [8] mode.

Liu [9] uses seven cubics to update on-line a smooth mono-dimensional motion. Andersson [10] uses a single quintic polynomial for representing the entire trajectory, while Macfarlane [11] extends Andersson's work and uses seven quintic polynomials for industrial robots. Lloyd [12] proposes to adjust the spatial shape of the transition curve of adjacent path segments.

In the case of human interaction, Amirabdollahian et al. [13] use a seventh order polynomial, while Seki and

Tadakuma [14] propose the use of fifth order polynomial, both of them for the entire trajectory with a minimum jerk model. Herrera and Sidobre [15] propose seven cubic equations to obtain *Soft Motions* for robot service applications.

III. TRAJECTORY PLANNING LEVEL

In this section we present two trajectory planners to control a service manipulator robot. The first one approximates a given trajectory, and the second one plans optimal-time trajectories. Both trajectories are composed of series of cubic polynomials.

A. Trajectory Approximation

A trajectory tr_{in} is defined along a path $p = p(u)$ by a motion law $u = u(t)$. The path p can be described by a very large variety of curves. However the robot controller can only realize a small subset. Thus, the trajectory tr_{in} is usually approximated. First, we select a set of points P_{tr} of tr_{in}. Then, between each consecutive pair of points, we compute an interpolated sub-trajectory str_i using three cubic polynomials.

1) Definition of the generated trajectory: We consider a trajectory tr_{out} (Fig. 2) composed of N cubic segments of time length T_k. Curves $J_k(t)$, $A_k(t)$, $V_k(t)$, $X_k(t)$ respectively represent jerk, acceleration, velocity and position functions according to the segment k. Each segment equation (1) is characterized by its initial conditions, the constant jerk value (J_k), the initial time $t_l = \sum_{i=1}^{k-1} T_i$ and the final time $t_l + T_k$.

$$X_k(t) = \frac{J_k}{6}(t-t_l)^3 + \frac{A(t_l)}{2}(t-t_l)^2 + V(t_l)(t-t_l) + X(t_l) \quad (1)$$

where J_k, $A(t_l)$, $V(t_l)$, $X(t_l)$ and t_l are constants.

The initial conditions of the trajectory are $A_1(0) = A_0$, $V_1(0) = V_0$ and $X_1(0) = X_0$ and the final conditions are $A_N(t_f) = A_f$, $V_N(t_f) = V_f$ and $X_N(t_f) = X_f$ where $t_f = \sum_{i=1}^{N} T_i$.

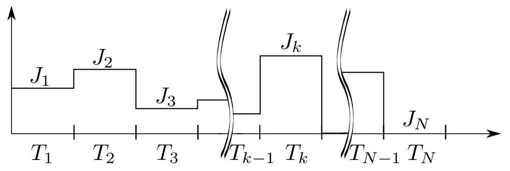

Fig. 2. Series of cubic segments

Given the N couples (J_k, T_k), we can compute the motion state at time $t \in [t_l, t_l + T_k]$ with the following equations:

$$A_k(t) = J_k\left(t - \sum_{i=1}^{k-1} T_i\right) + \sum_{i=1}^{k-1} J_i T_i + A_0 \quad (2)$$

$$\begin{aligned} V_k(t) = {} & \frac{J_k}{2}\left(t - \sum_{i=1}^{k-1} T_i\right)^2 + \sum_{i=1}^{k-1} J_i T_i \left(t - \sum_{i=1}^{k-1} T_i\right) \\ & + \sum_{i=1}^{k-1}\left(\frac{J_i T_i^2}{2} + \sum_{j=i+1}^{k-1} J_i T_i T_j\right) + A_0 t + V_0 \end{aligned} \quad (3)$$

$$\begin{aligned} X_k(t) = {} & \frac{J_k}{6}\left(t - \sum_{i=1}^{k-1} T_i\right)^3 + \left(\sum_{i=1}^{k-1} \frac{J_i T_i}{2}\right)\left(t - \sum_{i=1}^{k-1} T_i\right)^2 \\ & + \left[\sum_{i=1}^{k-1}\left(\frac{J_i T_i^2}{2} + \sum_{j=i+1}^{k-1} J_i T_i T_j\right)\right]\left(t - \sum_{i=1}^{k-1} T_i\right) \\ & + \sum_{i=1}^{k-1}\left(\frac{J_i T_i^3}{6} + \sum_{j=i+1}^{k-1} \frac{J_i T_i T_j^2}{2}\right) \\ & + \sum_{i=1}^{k-1} T_i \sum_{j=1}^{i-1}\left(\frac{J_j T_j^2}{2} + \sum_{k=j+1}^{i-1} J_j T_j T_k\right) \\ & + \frac{A_0}{2}t^2 + V_0 t + X_0 \end{aligned} \quad (4)$$

2) Why a three segments sub-trajectory?: An imposed time motion between two point involves seven constraints: three initial conditions, three final conditions and the imposed time T_{imp}. A single cubic polynomial (eq. 1) is defined by only five parameters and can not represent such motion. We need at least two cubic polynomials (10 parameters). Then, the system can be solved using the three constraints of continuity between the two cubics. From equations (2), (3) and (4) and the imposed time constraint, we obtain the system of four equations with four parameters (J_1, J_2, T_1 and T_2):

$$A_f = J_1 T_1 + J_2 T_2 + A_0 \quad (5)$$

$$V_f = J_2 \frac{T_2^2}{2} + J_1 T_1 T_2 + J_1 \frac{T_1^2}{2} + A_0(T_1 + T_2) + V_0 \quad (6)$$

$$\begin{aligned} X_f = {} & J_2 \frac{T_2^3}{6} + J_1 T_1 \frac{T_2^2}{2} + J_1 T_2 \frac{T_1^2}{2} + J_1 \frac{T_1^3}{6} \\ & + \frac{A_0}{2}(T_1 + T_2)^2 + V_0(T_1 + T_2) + X_0 \end{aligned} \quad (7)$$

$$T_{imp} = T_1 + T_2 \quad (8)$$

Unfortunately, the switching time T_1 must be constrained inside the interval $[t_l, t_l + T_k]$. Therefore, the obtained solution is generally invalid. Another solution consists in using a third segment. In this case, there are 15 parameters and 13 constraints. However, if we set the two switching times T_1 and T_2, we also obtain 15 constraints and we can solve the linear system (9) composed of 3 parameters J_1, J_2 and J_3. This system is simpler than the previous one and can be directly solved. It is clear that adding a fourth segment introduces 20 parameters and 16 constraints. Setting the three switching time provides only three additional constraints and the system can not be directly solved.

3) Computing the three segments sub-trajectory: From III-A.1, we determine the system of equation for 3 segments ($N = 3$). Hence, the systems becomes:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} J_1 \\ J_2 \\ J_3 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} \quad (9)$$

where the coefficients of the matrix A and B are computed as below:

$A_{11} = T_1$; $A_{12} = T_2$; $A_{13} = T_3$; $A_{21} = T_1^2/2 + T_1 T_2 + T_1 T_3$
$A_{22} = T_2^2/2 + T_2 T_3$; $A_{23} = T_3^2/2$

$A_{31} = T_1^3/6 + (T_1^2/2)(T_2 + T_3) + T_1(T_2^2 + T_3^2)/2 + T_1T_2T_3$
$A_{32} = T_2^3/6 + T_2^2T_3/2 + T_2T_3^2/2; \quad A_{33} = T_3^3/6$
$B_1 = A_f - A_0; \quad B_2 = V_f - V_0 - A_0T_{imp}$
$B_3 = X_f - X_0 - V_0T_{imp} - A_0T_{imp}^2/2$

We choose to set the two switching times on one-third of T_{imp}: $T_1 = T_2 = T_3 = \frac{T_{imp}}{3}$ (III-A.2). In this particular case, the matrix A becomes :

$$A = \begin{bmatrix} \frac{1}{3}T_{imp} & \frac{1}{3}T_{imp} & \frac{1}{3}T_{imp} \\ \frac{5}{18}T_{imp}^2 & \frac{3}{18}T_{imp}^2 & \frac{1}{18}T_{imp}^2 \\ \frac{19}{162}T_{imp}^3 & \frac{3}{162}T_{imp}^3 & \frac{1}{162}T_{imp}^3 \end{bmatrix} \qquad (10)$$

Thus, we can directly compute the three jerk values. Note that there is no particular reason to divide the segment into three equal times. Further optimization could be made by varying these three time intervals. Both methods are compared in Section V-A.

4) Trajectory verification and correction: Once the trajectory is planned, we have to verify if each cubic segment satisfies the kinematic constraints of jerk J_{max}, acceleration A_{max} and velocity V_{max}. If one of the kinematic constraints is out of bounds, the trajectory can be slowed provided that the task allows this local motion law change. If the trajectory is not planned in the joint space, we also have to verify the bounds in the joint space. For a given T_{imp} and intial and final conditions, the matrix A and B are constant. Thus, the error is bounded.

B. Planning an Optimal-time Trajectory

This method uses the Soft Motion Trajectory Planner presented in [5] inside the motion planner Move3D [1]. This cartesian trajectory planner produces series of cubic for each operational axes (translation and rotation). The inputs of the planner are the initial and the final kinematic states (position, velocity and acceleration) and the desired bounds (velocity, acceleration and jerk). The resulting trajectory is then a polynomial trajectory merged from six series of polynomial cubics. Since this planner does not use an optimization step, it can be used on-line in a control loop to modify the trajectory, for example, object exchange task (robot taking the object from the human). The direct definition of trajectories as a function of time avoids the function composition of a path and a motion law and thus, simplify the manipulation of these trajectories (adapt the overall motion time or modifying bounds).

The soft motion trajectory planner is based on sampling techniques [1]. The model of the $\mathcal{C}$-space is not explicitly constructed. We use a probabilistic method that generates a feasible robot trajectory in the cartesian space for the end effector. A set of nodes (configuration of the manipulator) represents the trajectory. Each motion between nodes is a point to point motion (straight line path) composed of series of cubic segments for translations and rotations that saturate jerk, acceleration and velocity (Fig.3). This point to point motion is composed of seven segments at most [5]:

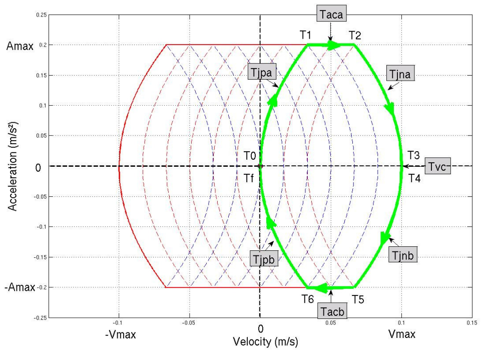

Fig. 3. Time optimal point-to-point motion

$T_{jpa} = T_1 - T_0$	Jerk positive time
$T_{aca} = T_2 - T_1$	Acceleration constant time
$T_{jna} = T_3 - T_2$	Jerk negative time
$T_{vc} = T_4 - T_3$	Velocity constant time
$T_{jnb} = T_5 - T_4$	Jerk negative time
$T_{acb} = T_6 - T_5$	Acceleration constant time
$T_{jpb} = T_f - T_6$	Jerk positive time

Once the probabilistic planner has found a path without collision in the generated roadmap, the soft motion planner is applied to build a soft trajectory. This algorithm is presented for a n axes planning:

Algorithm 1 *Path Following*

```
1: $Tr_{out} = \varnothing$
2: $Tr_{ptp}$ = Compute point-to-point motion between each
   consecutive pair of the $NB\_NOD$ nodes
3: for $node = 1$ to $(NB\_NOD - 1)$ do
4:     $IC_T = getIC_T(node, r_{ptp})$
5:     $FC_T = getFC_T(node, r_{ptp})$
6:     $Tr_{unsync}[node] = computeOptimTraj(IC_T, FC_T)$
7:     $Tr_{sync}[node] = synchronizeTraj(Tr_{unsynchr}[node])$
8:     $appendToTraj(Tr_{ptp}, Tr_{synchr}[node], Tr_{out})$
9: end for
10: return $Tr_{out}$;
```

The computed trajectory Tr_{ptp} stops at each intermediate node (line 2). Then, for each node, we get the initial conditions, IC_T, and final conditions, FC_T, for the transition motion (lines 4,5). IC_T is the end point of the *Tvc* segment of the point-to-point motion before the considered node. FC_T is the first point of the *Tvc* segment of the point-to-point motion after the node. Then, we compute $Tr_{unsync}[node]$ (line 6) using the algorithm developed for the mono-dimensional case in [5]. We obtain the optimal motion times T_{opt} for each of the n axes (6 in our case). Finally, we constrain the motion duration of each axis to a minimal common time $T_{imp} \geq max_{i\in[1,n]}(T_{opt}[i])$ in order to obtain a synchronized trajectory $Tr_{sync}[node]$ (line 7).

In [5] we propose to compute a *Slowing velocity motion* to adjust the time of the multi-axes transition motion. Even if acceleration is null in this case, this solution could take into account non null initial and final kinematic condition.

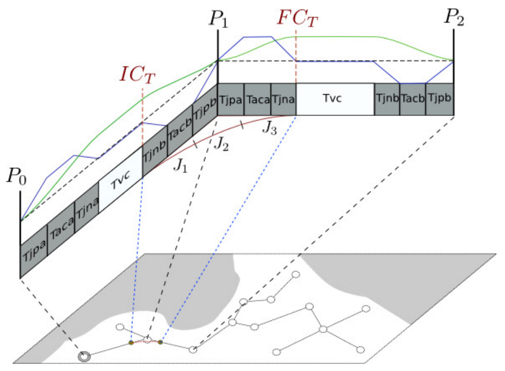

Fig. 4. Planning to remove halt at node

However, this method needs a loop to find the value of the imposed time. In this paper, we propose a new method using the results of the approximation method described in the section III-A.1. Given the imposed time $T_{imp} = max_{i\in[1,n]}(T_{opt}[i])$, we compute the imposed time motions for the $n-1$ axes as shown in the Section III-A.1. Hence, the transition motion is composed of one seven-segment optimal motion and $n-1$ three-segment motions.

IV. CONTROL LEVEL

In this section we focus our attention on the arm manipulator's control loop. Since the user's comfort depends on the arm motion, the control loop needs to consider the Soft Motion constraints.

The configuration of a six joint arm manipulator is defined by a vector θ of six independent *joint coordinates* which correspond to the angles of the articulations.

$$\theta = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix}^T$$

The *Pose* of the manipulator's end effector is defined by a position vector $\mathbf{P}$ and by a quaternion $\mathbf{Q}$ with seven coordinates total, 3 for $\mathbf{P}$ and 4 for $\mathbf{Q}$, named *Operational Coordinates* which gives the position and the orientation of the final body in the reference frame.

$$\mathbf{P} = \begin{bmatrix} x \ y \ z \end{bmatrix}^T \qquad \mathbf{Q} = \begin{bmatrix} n \ \mathbf{q} \end{bmatrix}^T \quad \text{where} \qquad \mathbf{q} = \begin{bmatrix} i \ j \ k \end{bmatrix}^T ; \parallel \mathbf{Q} \parallel = 1$$

These seven coordinates are the axes used in the trajectory planning level. Thus, we have the acceleration, the velocity and the position evolution for the seven coordinates. The relation between joint velocities and cartesian velocities is given by:

$$\begin{bmatrix} \mathbf{V} \ \mathbf{\Omega} \end{bmatrix}^T = \mathbf{J}(\theta)\dot{\theta} \tag{11}$$

where $\mathbf{V}$ and $\mathbf{\Omega}$ represent the linear and angular velocities of the robot's end effector and $\mathbf{J}$ the (6x6) Jacobian matrix. The speed of the axes is given by :

$$\dot{\theta} = \mathbf{J^{-1}}(\theta) \begin{bmatrix} \mathbf{V} \ \mathbf{\Omega} \end{bmatrix}^T \tag{12}$$

The linear velocities $\mathbf{V}$ obtained by the planner can be directly applied as velocity references. The evolution of the quaternion $\dot{\mathbf{Q}}$ must be transformed into angular velocities. We use the transformation function proposed in [16]:

$$\begin{bmatrix} \mathbf{\Omega} \\ 0 \end{bmatrix} = 2\mathbf{Q_r}^\top \dot{\mathbf{Q}} \qquad \text{where} \qquad \mathbf{Q_r} = \begin{bmatrix} n & k & -j & i \\ -k & n & i & j \\ j & -i & n & k \\ -i & -j & -k & n \end{bmatrix}$$

In a closed loop control [17], the control law is replaced by

$$\dot{\theta} = \mathbf{J^{-1}}(\theta) \begin{bmatrix} \mathbf{V} - \mathbf{K_p e_p} \\ \mathbf{\Omega} - \mathbf{K_o e_o} \end{bmatrix} \tag{13}$$

where $\mathbf{K_p}$ and $\mathbf{K_o}$ are diagonal gain matrices, and $\mathbf{e_p}$ and $\mathbf{e_o}$ respectively represent the position and orientation error vectors defined by :

$$\mathbf{e_p} = \mathbf{P} - \mathbf{P_d} \qquad \mathbf{e_o} = n_d\mathbf{q} - n\mathbf{q_d} - \mathbf{q_d} \times \mathbf{q} \tag{14}$$

where the desired position and quaternion are $\mathbf{P_d} = \begin{bmatrix} x_d \ y_d \ z_d \end{bmatrix}^T$ and $\mathbf{Q_d} = \begin{bmatrix} n_d \ \mathbf{q_d} \end{bmatrix}^T$ and the current state is defined by $\mathbf{P}$ and $\mathbf{Q}$.

To improve the accuracy in the position and orientation loop, Yuan [18] use a proportional controller and shows global asymptotic convergence for $K_p > 0$ and $K_o > 0$. The control law (13) can be interpreted as a position proportional controller plus velocity feedforward for each direction. In our application we change the proportional controller at each direction by a proportional integral digital controller of the form [19]:

$$u[kTs] = u[(k-1)T_s] + \Delta u[kT_s] \tag{15}$$

$$\Delta u[kTs] = C\left((e[kTs] - e[(k-1)Ts]) + \frac{Ts}{T_i}e[kTs]\right) \tag{16}$$

where T_s is the sampling time, k represents the current values and T_i the integral time.

We have limited the control law to achieve soft motion. By limiting $\Delta u[k]$, we limit the acceleration for each axis. By limiting $u[k]$, we limit the velocity for each axis and we avoid the integral saturation problem. Considering this controller and the robot as an integrator, there are two integrators in the control loop, that provide a velocity tracking system.

V. PRACTICAL APPLICATIONS AND RESULTS

In this section we present two applications along their results. The first one compares our trajectory approximation method to an existing one. The second application consists in the integration of the *Path Following plannner* in a real robot platform.

A. Approximation of a 3D Trajectory in the Task Space

In [6] a path $p = p(u)$ composed of lines and arcs of circle is approximated using cubic *B-spline* functions (Fig. 5). The motion law $u(t)$ along the path is a simple point to point soft motion computed by our soft motion planner [5]. The kinematic constraints are $J_{max} = 200m/s^3$, $A_{max} = 40m/s^2$ and $V_{max} = 2m/s$. The A_{max} value cannot be reached as $\frac{V_{max}}{A_{max}} < \frac{A_{max}}{J_{max}}$. We choose this curve to test our approach because it is a common approach for industrial robots. Besides, this type of trajectory is interesting because

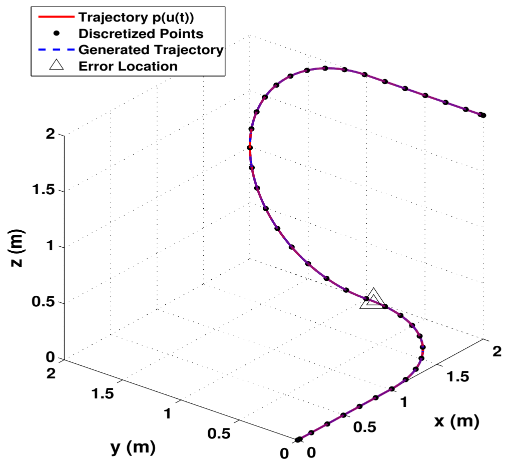

Fig. 5. Theoretical path, computed trajectory and maximum error location

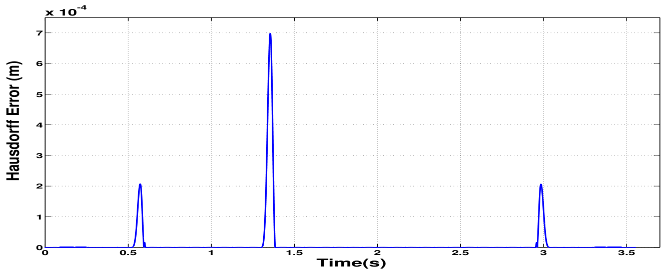

Fig. 6. Hausdorff error between the given path and the computed path

of the acceleration discontinuity at the point that links a line and an arc of circle.

To approximate this trajectory, we use 10 then 42 points separated by equal-time intervals and interpolate using the 3 equal-time segments method described in Section III-A. Fig. 6 shows the Hausdorff distance error. At the instant 1.35s, the maximum error appears because of the circle's switching which modifies the centripedal acceleration direction. In Fig. 7, which presents the acceleration and velocity curves, centrifugal acceleration appears between instant 0.5s and 3s.

The Table I presents the comparison between the three methods: the trajectory computed using the 3 equal-time segments (*Traj. 1*) (see Section III-A), the trajectory with 3 optimized-time segments to obtain a minimum trajectory error (*Traj. 2*) and the cubic *B-spline* approximation used in [6] (*Traj. 3*). In each case, we compute the Hausdorff distance error, the maximum motion law error and the maximum trajectory error.

B. Optimal Trajectory Planning for a Specified Task

We implemented the *Path Following planner* presented in this paper on our robot Jido (Fig. 8), a mobile Neobotix platform MP-L655 with top mounted manipulator PA10 from Mitsubishi. The software control is developed using Open Robots tools: GenoM [20]. The sampling time is fixed to 10 ms. The linear and angular end-effector motions are limited for the cartesian axes by $J_{max} = 0.9m/s^3$, $A_{max} = 0.3m/s^2$ and $V_{max} = 0.15m/s$ and for the

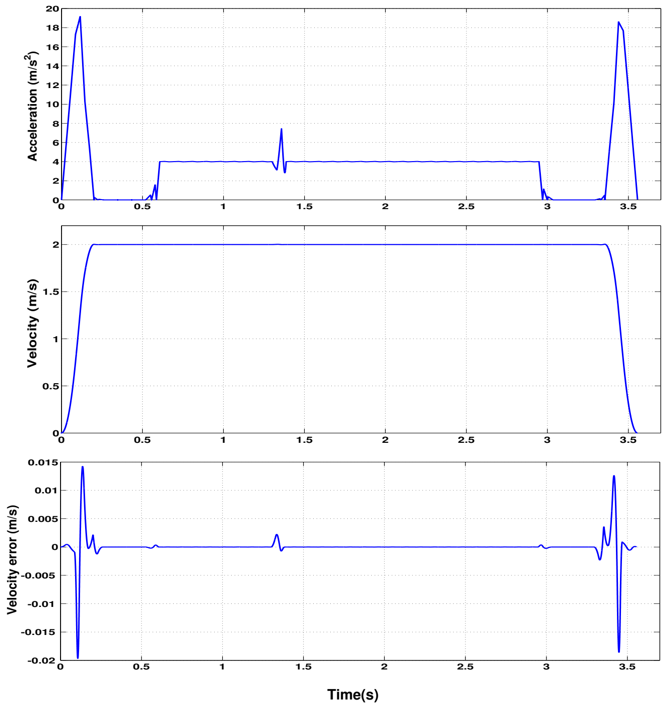

Fig. 7. Computed acceleration law, computed motion law and error between desired and computed motion law

TABLE I

ERRORS FOR DIFFERENT DISCRETIZATIONS

Nb point	Error	Traj. 1	Traj. 2	Traj. 3
10 points	Hausdorff (mm)	1.5	0.28	19.4
	Motion law (mm/s)	333	7.50	-
	Trajectory (mm)	27.2	0.31	-
42 points	Hausdorff (mm)	0.7	0.035	3.2
	Motion law (mm/s)	19.2	0.894	-
	Trajectory (mm)	0.7	0.0345	-

rotation axes by $J_{max} = 0.6m/s^3$, $A_{max} = 0.2m/s^2$ and $V_{max} = 0.1m/s$. For this hand-over task illustrated in Fig. 8, the planner HAMP [2] generates a path not only safe but also comfortable and legible (easily comprehensible without any further need of communication) by taking into account kinematics, field of view, posture, state and preferences of the human partner.

The 6D trajectory generated using the method in Section III-B and executed by the arm manipulator is shown in Fig. 9. Fig. 10 presents the trajectory of the X axis where the jerk curve shows the differents parts of the trajectory realized. The first three segments came from the first point-to-point motion which reaches A_{max}. Since the second point-to-point motion does not reach A_{max} the trajectory uses only two segments. The three segments in the center are computed with the time imposed method.

VI. CONCLUSIONS AND FUTURE WORKS

We described a complete system for planning and control of robot trajectories for service robots. In this particular context, safe moves prohibit large velocities and so the need

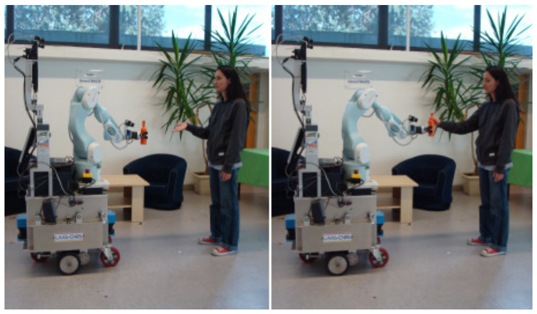

Fig. 8. Jido handing over the bottle

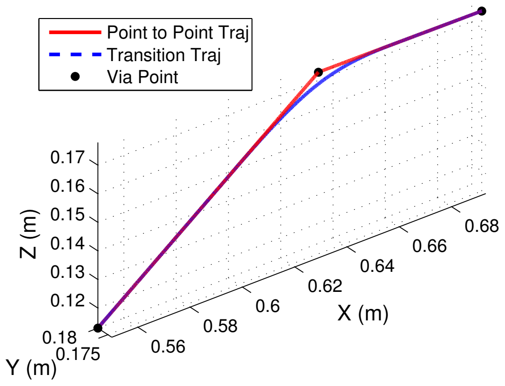

Fig. 9. Handing over trajectory executed by Jido

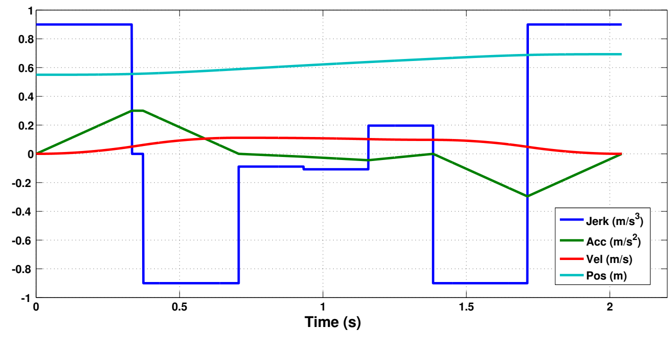

Fig. 10. Handing over trajectory along the X axis

for dynamic models. The robotic system proposed is a multi-layer system composed of planners (path planner, trajectory planner...) and controllers (join and task level). Each of these systems manipulates trajectories defined by series of cubic polynomial. The soft motion planner, extended to better compute the imposed time motions, provides a versatile real time planner. Trajectory approximation can be done efficiently with series of three cubic segments. The same type of trajectory is used to manipulate the path, the kinematic constraints (continuity in position, velocity and acceleration) and the time in the operational and in the joint space.

The possibility to manipulate trajectories offers the opportunity to modify the trajectory in real time. For example, the planner can modify the trajectory based on the current positions of objects. A visual control loop can adapt the trajectory at a given sampling time in parallel with a second loop with bigger sampling time that control the forces. A second challenge consists in building a humanoid robot manipulating such trajectory from planning to control levels and capable to deal with redundant and holonomic constraints.

VII. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement n 216239 with DEXMART project. This work is also supported by the ANR project 07-ROBO-0011 ASSIST.

REFERENCES

[1] T. Siméon, J.-P. Laumond, and F. Lamiraux, "Move3d: a generic platform for motion planning," in *4th International Symposium on Assembly and Task Planning*, Japan, 2001, pp. 25–30.

[2] E. A. Sisbot, L. F. Marin, and R. Alami, "Spatial reasoning for human robot interaction," in *IEEE/RSJ IROS*, San Diego, USA, 2007.

[3] I. european project PHRIENDS, http://www.phriends.eu/.

[4] A. Bicchi and G. Tonietti, "Dealing with the safety-performance trade-off in robot arms design and control," *IEEE Robotics and Automation Magazine*, vol. (in press), 2004.

[5] X. Broquere, D. Sidobre, and I. Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *IEEE/RSJ IROS*, Nice, France, 2008.

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

[7] P. F. Jingyan Dong and J. Stori, "Feed-rate optimization with jerk constraints for generating minimum-time trajectories," *International Journal of Machine Tools and Manufacture*, 2007.

[8] C. Zheng, Y. Su, and P. Müller, "Simple online smooth trajectory generations for industrial systems," *Mechatronics*, vol. 19, no. 4, pp. 571–576, 2009.

[9] S. Liu, "An on-line reference-trajectory generator for smooth motion of impulse-controlled industrial manipulators," in *7th International Workshop on Advanced Motion Control*, 2002, pp. 365–370.

[10] R. L. Andersson, "Aggressive trajectory generator for a robot ping-pong player," *IEEE Control Systems Magazine*, vol. 9, pp. 15–20, February 1989.

[11] S. Macfarlane and E. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 42–52, February 2003.

[12] J. Lloyd and V. Hayward, "Trajectory generation for sensor-driven and time-varying tasks," *International Journal Robotics Research*, vol. 12, pp. 380–393, August 1993.

[13] R. L. Farshid Amirabdollahian and W. Harwin, "Minimum jerk trajectory control for rehabilitation and haptic applications," in *ICRA*, May, 2002, pp. 3380–3385.

[14] K. Seki and S. Tadakuma, "Minimum jerk control of power assiting robot based on human arm behavior characteristic," in *International Conference on Systems, Man and Cybernetics*, 2004, pp. 722–721.

[15] I. Herrera and D. Sidobre, "On-line trajectory planning of robot manipulator's end effector in cartesian space using quaternions," in *15th Int. Symposium on Measurement and Control in Robotics*, 2005.

[16] B. H. and S. J., "Introduction to intelligent robotics," Katholieke Universiteit de Leuven, Tech. Rep., 2001.

[17] C. Wu and P. R.P., "Resolved motion force control of robot manipulator," *IEEE Trans. Syst., Man Cybern.*, pp. 266–275, May 1982.

[18] J.-C. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE J. Robotics and Automotion*, vol. 4, pp. 434–440, August 1988.

[19] M. S. Mahmoud, *Computer Operated Systems Control*. Marcel Dekker, Inc, 1991.

[20] S. Fleury, M. Herrb, and R. Chatila, "Genom: a tool for the specification and the implementation of operating," 1997.