On-line trajectory planning of robot manipulator's end effector in Cartesian Space using quaternions

Ignacio Herrera Aguilar[†♠] and Daniel Sidobre[†‡]
(iherrera, daniel)@laas.fr

[†] LAAS-CNRS
7, avenue du Colonel Roche
31077 Toulouse, France

[‡] Université Paul Sabatier
118, route de Narbonne
31062 Toulouse, France

[♠] Inst. Tec. de Orizaba
Av. Inst. Tecnológico No. 852
94320 Orizaba, Ver. México

Abstract

This paper presents a Soft motion trajectory planner for service robotic. The planner produce a minimal time solution in point to point problem and a quasi minimal along a specified path. The path is defined by a set of points in the space and the trajectory is subject to jerk, acceleration and velocity constrains in the Cartesian space. Based on a quaternion representation of the robot manipulator's end effector orientation, we proposed a method for the trajectory planning in a seven dimension space. The problem is divided in two phases. Firstly, the method solves for each dimension by decomposing the trajectory in different cubic segments. Secondly to guarantee the tracking, the motion between successive points is considered monotonic, a time adjust is done.

1 Introduction

Arm manipulator control has been standardized little by little, industrial applications have been developed using different techniques, several restrictions have been satisfied by using robot-like arms with specific application for a limited number of tasks. However, it is in the same applications that robots are confined at structured and safe spaces, free of man interaction.

User's comfort and safety in robot presence is an important topic to service robotic. Service task like water glass transport or grasping a cup of coffee needs particular precautions. Smooth trajectory planning has been used in different works by conditioning the jerk to limit mechanical stress and to obtain optimal trajectory. We consider **Soft Motion** to limit jerk, acceleration and velocity. Soft motion is defined by a soft start and a soft stop with limited condition in acceleration and jerk. And the motion realized between these events is soft too, including direction changes and rotations.

The trajectory planner presented in this paper generate the necessary references to produce *soft motion* for robot manipulator's end effector. We propopose a control loop that guarantees the soft motion characteristics of the end effector in the Cartesian space, by using quaternion feedback. The objective is to provide a *soft motion* while the robot performs a service task.

In this work only internal feedback is used for control, although we consider to extend our work to trajectory generation for sensor-driven tasks using vision and force feedback.

This papers presents in the next section the related work in Cartesian space trajectory planning. In section III, we continue with the description of the *soft motion* trajectory planner. In section IV, the control loop with quaternion feedback is presented. Finally, in sections V and VI, experimental results and conclusions are presented respectively.

2 Related Work

According to Brady [1], Trajectory Planning converts a description of a desired motion to a trajectory defining the time sequence of intermediate configurations of the arm between the origin and the final destination. Literature shows two different approaches. The first one considers working in joint space and the second one it task space. We have chose the last one.

We can separate the trajectory in two: the *path* that defines the curve followed by the end effector tip and the *rotation curve* defined as the orientation evolution of the end effector.

The first works in the area refers to Paul [2] and Taylor [3]. Paul use homogeneous coordinates, presents a matrix equation that relates the representation of a configuration as a sequence of frames, local to arm joints to a representation that is external to the arm and is determined by the application. Paul considers constant acceleration. Taylor presents a technique for achieving straight lines, by choosing midpoints between two desired configurations. Taylor propose the use of quaternions for rotation. Andersson [4] use a single quintic polynomial for representing the entire trajectory, while Macfarlane [5] extend Andersson's work and uses seven quintic polynomials.

3 Soft Motion Trajectory Planner

We consider the trajectory planning of points generated by a motion planning technique. The motion planner calculate the trajectory which the end effector must follow in space. However, the temporal characteristics of this movement are independent.

3.1 Monodimensional Case

Firstly, we consider the canonical case of the figure 1 without lost of generality.

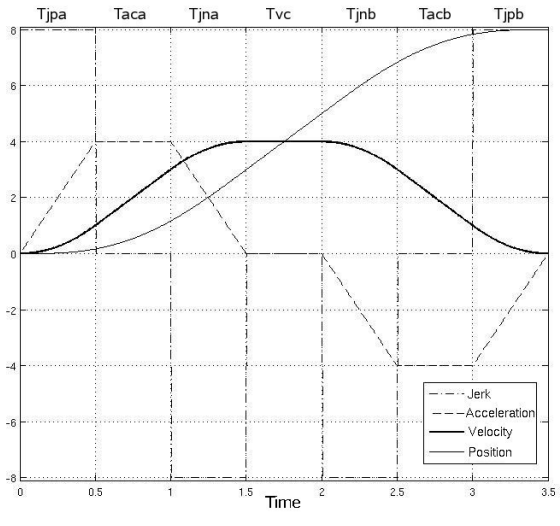

Figure 1: Jerk, Acceleration, Speed and Position curves

The motion can be separated in seven segments, defined by the time period. We have:

T_{jpa}	Jerk positive time	T_{jnb}	Jerk negative time (b for differencing from a)
T_{aca}	Acceleration constant time	T_{acb}	Acceleration constant time
T_{jna}	Jerk negative time	T_{jpb}	Jerk positive time
T_{vc}	Velocity constant time		

Considering one dimension motion and limit condition, we can find three different type sections by integration:

- The motion with a *maximum jerk* (J_{max}):
  $J(t) = J_{max}$
  $A(t) = A_0 + J_{max}t$
  $V(t) = V_0 + A_0t + \frac{1}{2}J_{max}t^2$
  $X(t) = X_0 + V_0t + \frac{1}{2}A_0t^2 + \frac{1}{6}J_{max}t^3$

- The motion with a *maximum acceleration* (A_{max}):
  $J(t) = 0$
  $A(t) = A_{max}$
  $V(t) = V_0 + A_{max}t$
  $X(t) = X_0 + V_0t + \frac{1}{2}A_{max}t^2$

- Finally, the equations for the motion with a *maximum velocity* (V_{max}):
  $J(t) = 0$
  $A(t) = 0$
  $V(t) = V_{max}$
  $X(t) = X_0 + V_{max}t$

where $J(t)$, $A(t)$, $V(t)$, $X(t)$ represents jerk, acceleration, velocity and position functions respectively. A_0, V_0 and X_0 are the initial conditions.

In the object to guarantee soft motion, we define the intervals:

$$J(t) \in [-J_{max}, J_{max}] \qquad A(t) \in [-A_{max}, A_{max}] \qquad V(t) \in [-V_{max}, V_{max}]$$

3.1.1 Point to point motion

According to figure 1, the motion is realized at limit conditions. To achieve A_{max} from initial condition $A(0) = 0$, we have a jerk time (T_j) that is equal to the time for going from A_{max} to 0. During T_j, the acceleration increase or decrease linearly according to the jerk. At this point, it is important to observe a symmetry in acceleration and an anti-symmetry in jerk. Now, we consider velocity, the symmetry effect is present too, but this time according to acceleration. During the constant acceleration time (T_a), the velocity increase or decrease linearly according to the acceleration. Finally, T_v is define as the constant velocity time. We have then

$$T_j = T_{jpa} = T_{jna} = T_{jnb} = T_{jpb} \qquad T_a = T_{aca} = T_{acb} \qquad T_v = T_{vc}$$

Our system calculates times T_j, T_a and T_v, whose make possible to obtain the desired displacement between an origin position and a final position. As the end effector moves under maximum motion conditions (J_{max},A_{max} or V_{max}), we obtain a *minimal time motion*. The complexity of the equations system depends on the distance between the positions and the maximal limits. Figure 2 presents different particular cases according to whether or not maximum acceleration or maximum speed are reached.

The point to point motion requires to reach the destination. Physical limitations are not considered, and in order to guarantee the emergency soft stop on desired path, null final conditions in acceleration and velocity are fixed ($A(t_f) = 0$ and $V(t_f) = 0$). Using this conditions we can find the necessary times T_{jmax} to achieve A_{max} and T_{amax} to achieve V_{max}.

$$T_{jmax} = \frac{A_{max}}{J_{max}} \qquad T_{amax} = \frac{V_{max}}{A_{max}} - \frac{A_{max}}{J_{max}} \tag{1}$$

According to Figure 2 we can build the Figure 3. Where can see maximal possible displacement in case 2, $T_v = 0$, $T_a = T_{amax}$ and $T_j = T_{jmax}$, and in case 3, $T_v = 0$ and $T_a = 0$ while $T_j = T_{jmax}$.

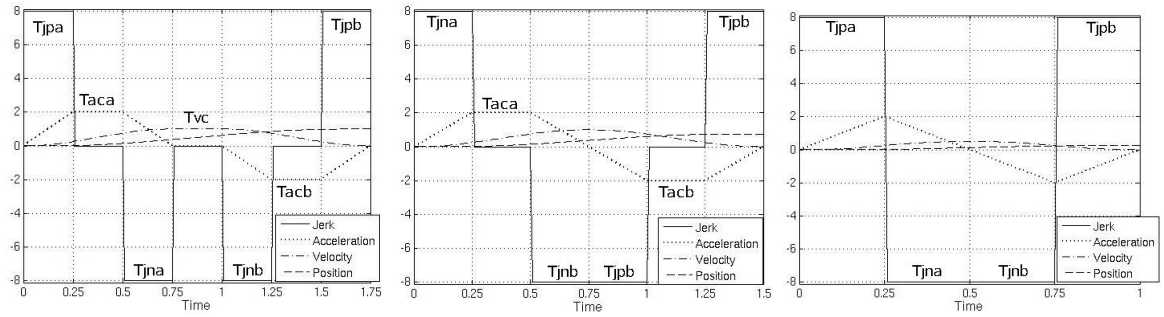

Figure 2: Case 1: V_{max} is reached and $T_v > 0$. Case 2: V_{max} is reached and $T_v = 0$. Case 3: A_{max} is reached and $T_a = T_v = 0$

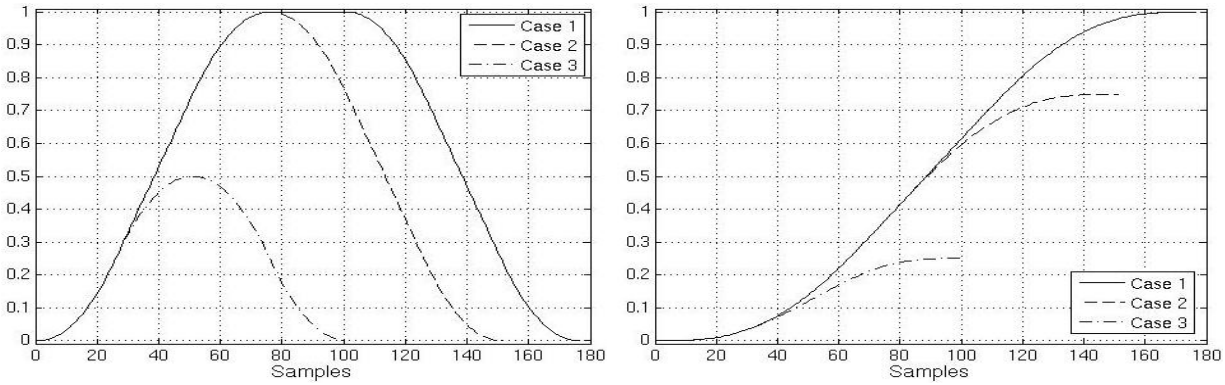

Figure 3: Velocities and Positions

We define the distance (D) as the difference between the origin (P_o) and destination (P_f) positions.

$$D = P_f - P_o \tag{2}$$

We have two limit conditions:

- Condition 1: Case 2, where V_{max} is reached. It means, A_{max} is reached too. Then we have to find the traversed distance(D_{thr1}). Using the limit times

$$T_j = T_{jmax} \qquad T_a = T_{amax} \qquad T_v = 0$$

we find

$$D_{thr1} = \frac{A_{max}V_{max}}{J_{max}} + \frac{V_{max}^2}{A_{max}} \tag{3}$$

- Condition 2: Case 3, where only A_{max} is reached. Using

$$T_j = T_{jmax} \qquad T_a = 0 \qquad T_v = 0$$

we can find a distance (D_{thr2})

$$D_{thr2} = 2\frac{A_{max}^3}{J_{max}^2} \tag{4}$$

Considering the conditions (Eqs. 3 and 4) we can formulate the algorithm 1

Since, the acceleration and speed curves are symmetrical. The optimal time for the trajectory in considering the constraints is given by:

$$T_f = 4 * T_j + 2 * T_a + T_v \tag{5}$$

Algorithm 1 Maximum Jerk Algorithm

Calculate distance D (Eq 2)
if $D \geq D_{thr1}$ **then**

$$T_j = T_{jmax} \qquad T_a = T_{amax} \qquad T_v = \frac{D - D_{thr1}}{V_{max}}$$

else
if $D \geq D_{thr2}$ **then**

$$T_v = 0 \qquad T_j = T_{jmax} \qquad T_a = \sqrt{\frac{A_{max}^2}{4J_{max}} + \frac{D}{A_{max}}} - \frac{3A_{max}}{2J_{max}}$$

else

$$T_v = 0 \qquad T_a = 0 \qquad T_j = \sqrt[3]{\frac{D}{2J_{max}}}$$

end if
end if

3.1.2 Multipoint Trajectory Planner

The strategy presented in previous section is extended for the multipoint case to go from P_0 to P_n. We define the current position P_c as a position in the interval P_i and P_{i+1} where $i = 0..n-1$. The trajectory is computed by successive application of seven cubic equations. For each segment, we consider initial conditions defined by previous segment at (P_c), and zero final conditions at (P_{i+1}) for acceleration and velocity.

Considering the trajectory generation knowing only the destination position (P_{i+1}), we compute the stop position (P_s) from the current motion conditions. Considering the stop position and the destination, we can find four possibilities.

- Start Motion
  The "easy" case, we applied previous algorithm. Because the current conditions are nulls.

- Same Direction Motion ($P_s > P_{i+1}$)
  The motion is in the same direction, the new destination is after the stop position. We applied the set of equations presented on section 3.1 with the current motion conditions.

- Halt Motion ($P_s = P_{i+1}$)
  The stop position and the new destination are equal. We consider to stop from current motion conditions.

- Change Direction Motion ($P_s < P_{i+1}$)
  This case is found when the final position P_{i+1} is before the stop position P_s. If we consider a natural evolution of the system of equations, some conditions have multiple solutions. For guarantee real-time applications, we have separate the Change Direction Motion in two, losing the optimal time. Firstly, a halt motion. Secondly, a start motion in the other direction.

We consider switch position P_c at the moment when the motion begin to slow down. This position defines the current position P_c as initial position for the next segment at time T_c defined by:

$$T_c = T_{jpa} + T_{aca} + T_{jna} + T_{vc} \tag{6}$$

The figure 4 shows the evolution for two destinations from the origin position ($P_0 = X(0) = 0$) to $P_1 = 0.5$ and $P_2 = 1.5$. Limits paramenters are $J_{max} = 8$, $A_{max} = 2$, $V_{max} = 1$ and sampling time is $T_s = 0.01$. According to figure, the motion going through P_1 in a non null velocity without losing the soft motion condition.

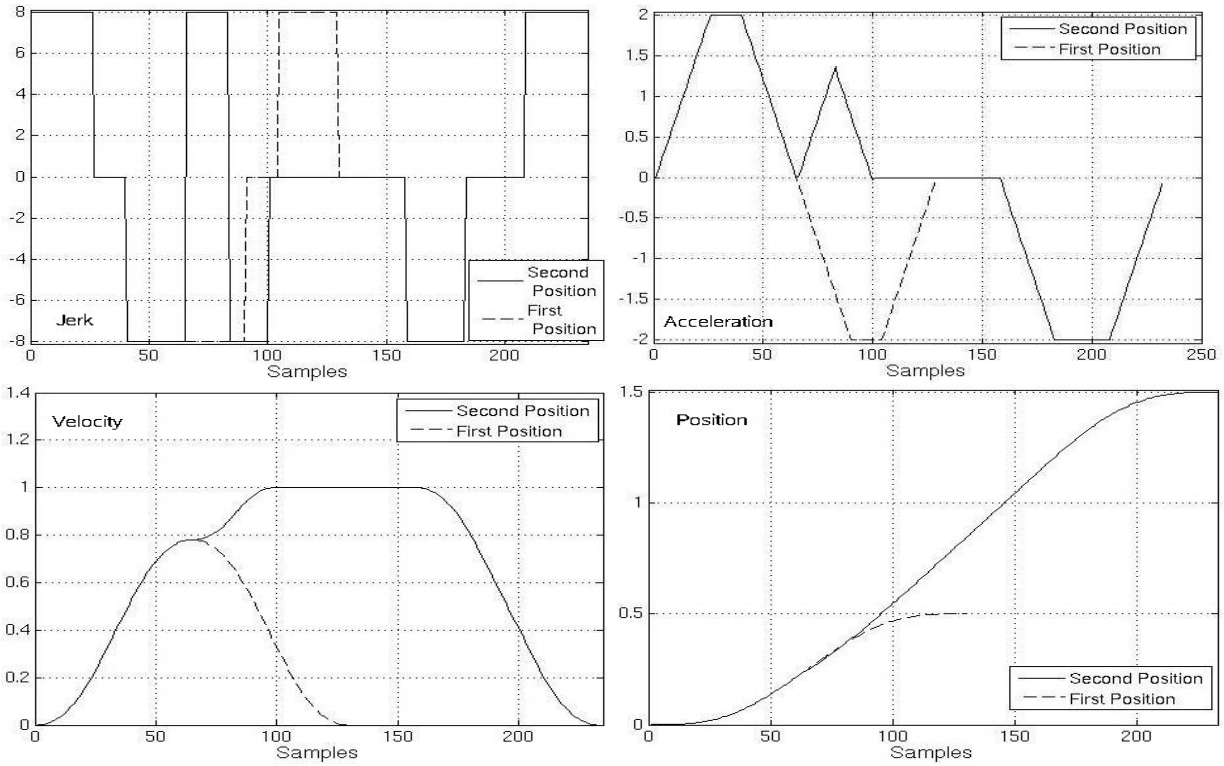

Figure 4: Trajectory planned for two positions

3.2 Multidimensional Case

For the multidimensional case, we keeps this strategy. Each dimension is independent each other. To guarantee trajectory tracking, we consider the motion between two points as a straight-line motion in n dimensional space. The only way for assuring straight-line tracking is assuring that each dimension motion has the same duration. Then, we compute the final time for each dimension. Considering the largest motion time, we readjust the other dimension motion to this time. Time adjusting is doing by decreasing limit conditions. In other words, the motion is minimum time for one direction. In the other directions, the motions are conditioned by the minimum one.

Figure 5 shows a two dimensional example with limit parameters $J_{max} = 8$, $A_{max} = 2$ and $V_{max} = 1$ for X direction and $J_{max} = 4$, $A_{max} = 1$ and $V_{max} = 0.5$ for Y direction. The origin is defined by the pair (0,0), the first destination point by (1,0.5) and the final destination point is (1.2,1.5).

In the case of robot's end effector we use seven motion dimensions. Three dimensions for translation and four for rotation modelized by quaternion. Linear velocities $\mathbf{V}$ obtained can be applied directly as velocity references. On another hand, the evolution of the quaternion $\dot{\mathbf{Q}}$ must be transformed into angular velocities. We use the transformation function proposed in [12].

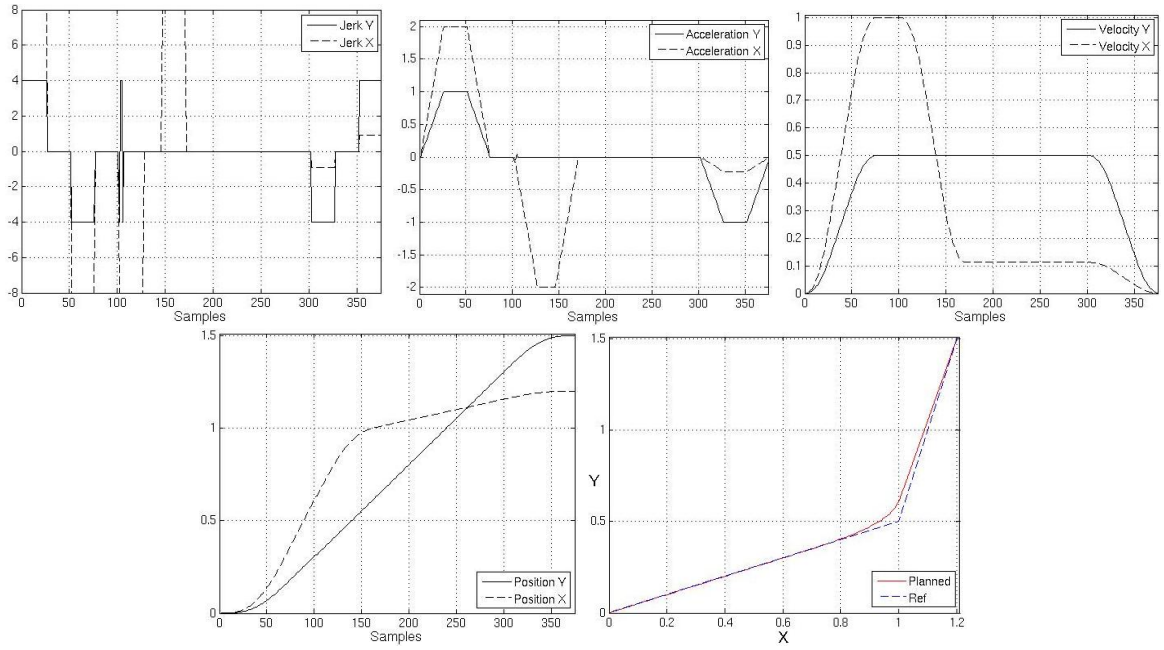

Figure 5: Trajectory planned for two positions in 2D. The last graph shows Y vs X curve

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

4 Resolved Motion Rate Control Loop

The configuration of six joints arm manipulator is defined by a vector θ of six independent *joint coordinates* which correspond to the angle 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 then is defined by M_b independent coordinates said *Operational Coordinates* which gives the position and the orientation of the final body in the reference frame. We use *quaternions* to represent the orientation of the end effector. The advantages of using quaternions are largely exposed in [6]. The conversion of the rotation matrix (**Rot**) into quaternion is detailed in [7].

We define **P** for the position and **Q** for the orientation

$$\mathbf{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \mathbf{Q} = n + \mathbf{q} \qquad \text{where} \qquad \mathbf{q} = \begin{bmatrix} i \\ j \\ k \end{bmatrix}$$

Resolved motion rate control means that the movements generated by the servo-motors of the articulations of the manipulator combine to produce a uniform displacement. In other words, the servo-motors evolve at different speeds with an aim of obtaining the desired total movement. Whitney [10] has shown that the speed of the axis is given by

$$\dot{\theta} = \mathbf{J^{-1}} \begin{bmatrix} \mathbf{V} \\ \mathbf{\Omega} \end{bmatrix} \tag{7}$$

where **V** and **Ω** represents the linear and angular velocities of the robot's end effector. And **J** is the Jacobian matrix.

In a closed loop control [11], the control law is replaced by

$$\dot{\theta} = \mathbf{J}^{-1} \begin{bmatrix} \mathbf{V} - \mathbf{K_p e_p} \\ \mathbf{\Omega} - \mathbf{K_o e_o} \end{bmatrix} \tag{8}$$

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. Yuan [8] uses quaternion feedback in a close loop resolved rate control. The position and orientation tracking error are defined by

$$\mathbf{e_p} = \mathbf{P} - \mathbf{P_d} \qquad \mathbf{e_o} = n_d \mathbf{q} - n \mathbf{q_d} + \mathbf{q_d} \times \mathbf{q} \tag{9}$$

where the index d indicates that they are set points.

Yuan showed global asymptotic convergence of the closed loop system (Eq. 8) for a gain matrix $\mathbf{K_o} > \mathbf{0}$. The figure 6 shows the control loop.

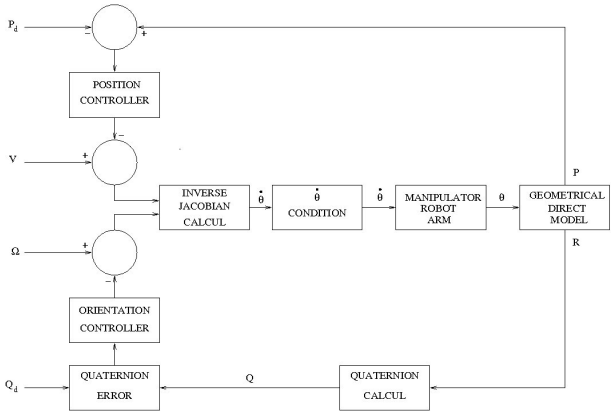

Figure 6: Modified Resolved Motion Rate Control Loop

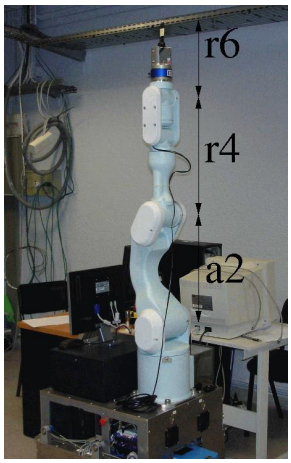

Figure 7: Jido-Arm

To improve the accuracy in the position and orientation loop, the controller for each dimension has been modified. Yuan use a Proportional controller and shows global asymptotic convergence for $K > 0$. In our application we change the controller by a PI digital controller.

5 Experimental Results

5.1 Experimental Platform

We have tested the control loop in a PA10-6CE Mitsubishi manipulator, called Jido-Arm (Figure 7). Jido-Arm is controlled by a PCI Motion Control CPU Board in a Pentium IV Personal Computer. Three lengths define the manipulator using Denavit-Hartemberg parameters: a2 = 0.480 m, r4 = 0.480 m and r6 = 0.30 cm.

The software control is developed using OpenRobots tools (LAAS Open Software for Autonomous Systems). The sampling time is fixed to 10 ms. The limits of joint velocities and robot's end effector are define by

Joint	Velocity limit (rad/seg)
q1	0.5
q2	0.5
q3	0.5
q4	1.0
q4	1.0
q6	1.0

	Linear Limit	(Angular limit)
J_{max}	$0.900 m/seg^3$	$0.600 rad/seg^3$
A_{max}	$0.300 m/seg^2$	$0.200 rad/seg^2$
V_{max}	$0.150 m/seg$	$0.100 rad/seg$

5.2 Tracking trajectory

This test is realized for a set of one hundred positions pre-calculated using a grasp planner [16]. The time needed for computing the soft motion is less than 30 ms. Figure 8 shows the results. We have separate the curves in acceleration, velocity and position. And a third image shows the manipulator evolution during the trajectory tracking. It is important to observe that initial position is singular.

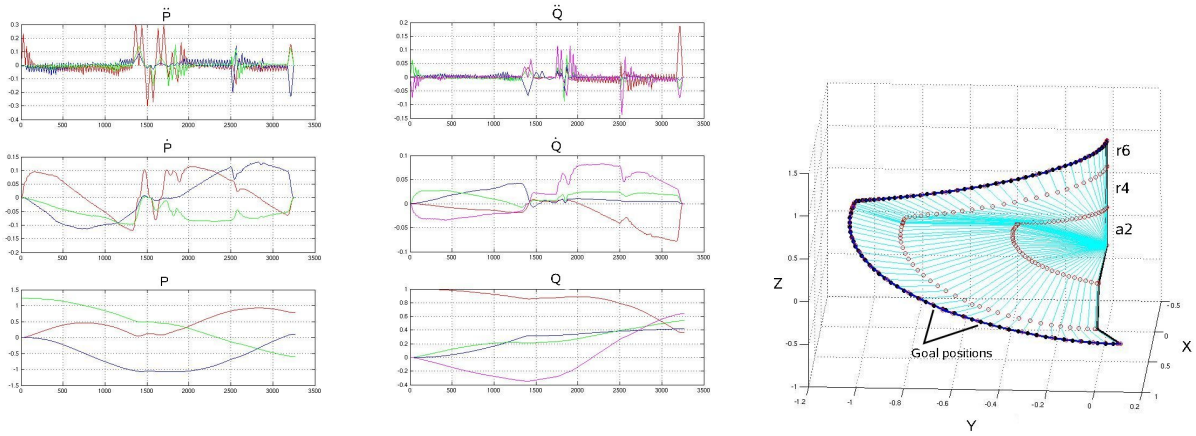

Figure 8: Soft Trajectory Planning. Left: Position trajectories. Center: Orientation Trajectories (Quaternion). Right: Manipulator evolution during trajectory tracking.

6 Conclusions

Soft motion conditions can be reached using the soft motion trajectory planner and the control loop presented in this paper. We presents a minimal time solution for point to point task and a non minimal for path tracking task. This paper shows that soft motion is possible by using seven cubic equations.

The soft motion trajectory planner is designed for on line operations. In the experimental case of trajectory tracking, we can compute all the set of references off line, but experimental results have done a good computational time for the set of positions (less than 30 ms). The sensor-driven trajectory planner has not been yet tested with sensor data point obtained by vision or force sensors. But simulation times for the Jido-platform have done times less than 10 ms for each position that guarantee the on line application. The soft motion trajectory planner is based on this equations and the systems of equations presented are analytically solved for guaranteed the on line execution. The maximal order of the polynomial equations founded is sixth, and to avoid numerical solution we have made some linearizations.

7 Acknowledgement

The work described in this paper was partially conducted within the EU Integrated Project COGNIRON ("The Cognitive Companion") and funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020. The authors acknowledge the support of the COSNET-DGEST-SFERE program.

References

[1] T. J. M. Brady, J. Hollerbach and T. Lozano-Perez, *Robot Motion, Planning and Control*. Cambridge, Massachusetts: The MIT Press, 1982.

[2] R. P. C. Paul, "Manipulator cartesian path control," *IEEE Trans. Syst., Man Cybern.*, vol. 9, pp. 702–711, May 1979.

[3] R. H. Taylor, "Planning and execution of straight line manipulator trajectories," *IBM Joournal of Research and Development*, vol. 23, pp. 424–436, 1979.

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

[5] 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.

[6] J. Funda, R. H. Taylor, and R. P. Paul, "On homogeneous transforms, quaternions and computational efficiency," *IEEE Trans. on Robotics and Automation*, vol. 6, pp. 382–388, Juin 1991.

[7] K. Shoemake, "Animating rotation with quaternion curves," *Proc. on Computer graphics and interactive techniques*, vol. 19, pp. 245–254, July 1985.

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

[9] D. E. Whitney, "The mathematics of coordinated control of prosthetic arms and manipulators," *ASME Trans. J. Dynamic, Systems, Meas., Cont.*, pp. 303–309, December 1972.

[10] D. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man-Machine Syst.*, vol. 10, pp. 47–53, Juin 1969.

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

[12] H. Bruyninckx and J. Shutter, "Introduction to intelligent robotics," Katholieke Universiteit de Leuven, http://www.mech.kuluveb.ac.be/ bruyninc/hb46/, Tech. Rep., 2001.

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

[14] D. E. Orin and W. W.Schrader, "Efficient jacobien determination for robot manipulators," in *Proc. First Int. Symp. on Robotics Research*, Bretton Woods, 1983, pp. 727–734.

[15] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," University of California, http://www.math.ucsd.edu/ sbuss/ResearchWeb/ikmethods/iksurvey.pdf, Tech. Rep., April 2004.

[16] E. Lopez-Damian, D. Sidobre, and R. Alami, "A grasp planner based on inertial properties," in *Proc. IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005, pp. 766–771.