Integration of Robotic Systems in a Packaging Machine: a Tool for Design and Simulation of Efficient Motion Trajectories

L. Biagiotti
DIEF - Univ. Modena Reggio Emilia
luigi.biagiotti@unimore.it

C. Melchiorri, M. Pilati
DEI - University of Bologna
claudio.melchiorri@unibo.it
matteo.pilati@gmail.com

G. Mazzuchetti, G. Collepalumbo, P. Ragazzini
IMA SpA
MazzuchettiG@ima.it, CollepalumboG@ima.it, RagazziniP@ima.it

Abstract

In this paper, the advantages of CACSD (Computer Aided Control System Design) tools for integrating a robotic system in a packaging machine are illustrated. Beside the mechanical integration of the robot into the machine architecture, it is necessary a functional integration, that requires a precise synchronization with the other parts of the system. In the proposed application, a robot with a parallel kinematics is used for pick-and-place tasks between two conveyor belts. It is therefore necessary a proper motion planning which allows to synchronize the grasp and release phases with the conveyor belts, avoiding obstacles and guaranteeing the compliance with bounds on velocity, acceleration and limits in the workspace. A trajectory composed by quintic polynomials has been considered and a specific tool has been designed in the Matlab environment, which allows to modify the parameters of the trajectory and to analyze the obtained motion profiles from both the kinematic and dynamic point of view.

1. Introduction

The integration of robotic systems in automatic machines is becoming more and more widespread because of the flexibility which they allow, and the high velocities of the manipulators which are comparable with those of the other components of this kind of machines. In particular, parallel robots, i.e. robots with a closed chain kinematics, are often used for pick-and-place operations [3]. As a matter of fact, the actuators located in the base and the arms made of light composite materials contribute to drastically reduce the inertia of moving parts allowing very high speeds and high accelerations. Moreover, the parallel structure increases the robot stiffness and the precision, but reduces its workspace volume. As shown in Fig. 1, the robot used in the proposed application is characterized by a Delta structure [4], with four degrees of freedom: three translational and one rotational, given by the rotation of the end effector. However, in the proposed application only the positional motion is taken into account. The robot Codian D4-500 is designed to perform with payloads up to 1 kilogram and for applications up to 15G. The chosen version has a cylindrical singularities-free working envelope of 450 mm in diameter and 135 mm in height. The main features of the manipulator are reported in Tab. 1. Note that the high velocities and accelerations allow a back and forth movement pick-place-pick ($z_1 = 25$ mm, $y = 305$ mm, $z_2 = 25$ mm) considering a load of 0.1 kg with a total cycle time of 0.3 seconds, which corresponds to the maximum achievable speed of 200 p/min[1]. Another considerable characteristics of this manipulator is its open mechanical interface, that

Figure 1. Robot with the Delta structure used in the proposed application: Codian D4-500 [5].

[1] p/min stands for pick-and-place actions per minute.

Pick & place actions (max.)	200 p/min
Position repeatability X,Y,Z	$\pm$ 0.2 mm
Angular repeatability Rz	$\pm$ 0.3°
Limitation rotation axis torque	2 Nm

Table 1. D4-500 performance features.

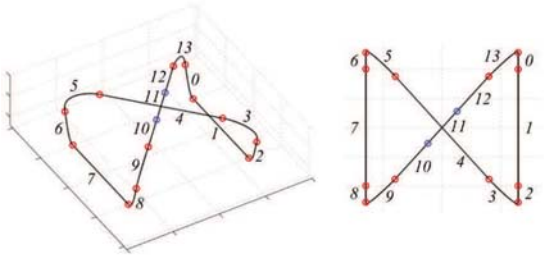


Figure 2. Generic trajectory for pick-and-place operations.

allows to choose various types of motors and gears. In this way, the robot integrated in an automatic machine can share with the other components of the machine the same type of actuators (in this case Yaskawa brushless motors [6]) and the same hw/sw control architecture. The robot manipulator can be seen as a set of three motion axes to be properly coordinated. Therefore, apart from the mechanical integration of the robot within the machine, the main issue remains the definition of the motion laws for the joint actuators.

In this paper, the use of CACSD (Computer Aided Control System Design) tools for the design and optimization of trajectories for the parallel robot is investigated and the advantages in terms of development time and correctness of the resulting motions are highlighted. Differently from other applications [2, 8], where the control and planning systems are firstly designed in the CACSD environment and then (automatically) converted in executable code for the plant's controller, in this case, because of the complexity of the machine and of the related control system, only the motion planner of the robot has been taken into account. After an initial design phase in the Matlab environment, the trajectory generator has been manually implemented in a C library (obtaining a very efficient code), and then integrated again in the Matlab environment in order to verify its correctness.

2. Trajectory design approach and trajectory generator structure

Since the specific task has not been defined yet, and it may change according to the type of automatic machine in which the robotic system is integrated, a class of trajectories for pick-and-place applications has been devised. In Fig. 2 a generic path is reported: the pick-and-place operations are supposed to be composed by a grasp phase of an object that moves with constant velocity on a conveyor belt followed by a transfer phase and by a release phase on a different conveyor belt with a different velocity. This trajectory must be repeated periodically with the same cycle time of the machine. Obviously, in many applications, it may happens that either the grasp or the release points are motionless. In these cases it is sufficient to set to zero the velocity along the tract that corresponds to the grasp or the release. The trajectory defined in the robot's workspace is composed by several polynomial segments, properly joined in order to guarantee the continuity of velocity and acceleration. In particular, the use of fifth-degree polynomials assures a smooth transition between the segments at the knots shown in the figure. The trajectory generator has the primary task of translating the specifications of a given application (position and velocity of grasp and release points, bounds on maximum velocity and acceleration, maximum height of the path) in the location of the waypoints, in the values of velocity and acceleration at these points and in the duration of each segment. Once the boundary conditions (initial and final position, velocity, acceleration, and time instants) are known the analytical expression of a generic tract

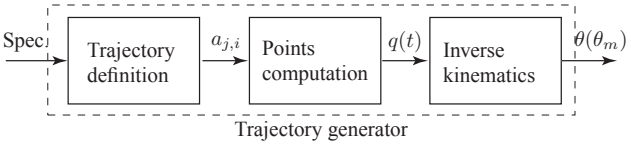


Figure 3. Structure of the trajectory generator.

$$q_i(t) = a_{0,i} + a_{1,i}(t - t_{0,i}) + \ldots + a_{5,i}(t - t_{0,i})^5, t_{0,i} \leq t \leq t_{1,i} \quad (1)$$

can be easily deduced according to the standard procedure for the computation of the coefficients of fifth-degree point-to-point trajectories, see [1]. Obviously, the parameters $a_{j,i}$, $j = 0, \ldots, 5$ are defined in $\mathcal{R}^3$. After the definition of the trajectory, the motion profile must computed as a function of the time. This operation, can be performed off-line by storing the results, i.e. the points of the trajectory, in a table or online by calculating (1) runtime for each tract. In order to guarantee a perfect synchronization of the robot with the other subsystems of the machine, the trajectory profile is also computed as a function of the angle of the master axis governing the overall machine. Since the actuators of the robot act at the joint level it is necessary to transform each point of the trajectory from the workspace to the joint space. Therefore, in the trajectory generator an additional module that computes the inverse kinematics of the robot is necessary. The final structure of the trajectory planner, composed by a set of routines written in C language, is reported in Fig. 3. Note the modularity of the structure which allows local changes in the trajectory definition, if the task changes, or in the inverse kinematics, if a different robot is used.

As illustrated in Fig. 4(a), this C library can be directly embedded in the code of the supervisor that controls the overall machine. This system runs on a standard industrial PC equipped with VxWorks and sends the reference points to the motors' drives via fieldbus (Yaskawa MECHATROLINK). On the other hand, by using the functionalities of MEX-functions (Matlab EXecutable functions) [7], it is possible to use the same C code in the Matlab environment. As matter of fact, MEX-functions provide an interface between Matlab and the C library, see Fig. 4(b). In this way, the routines which com-

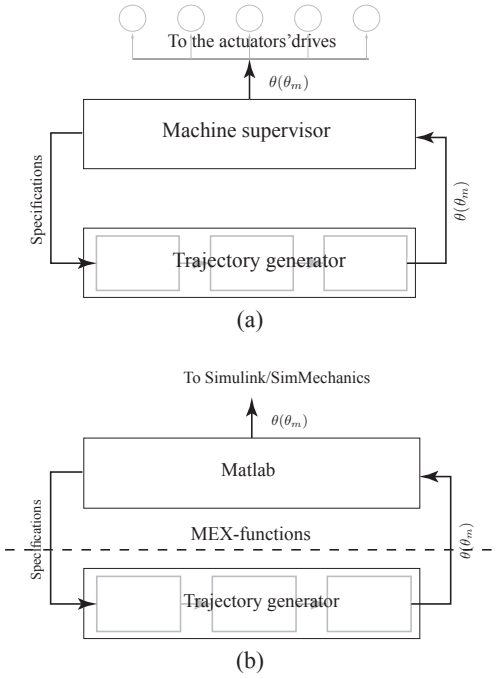


Figure 4. Integration of the C library of the trajectory generator in the machine controller (a) and in the Matlab environment (b).

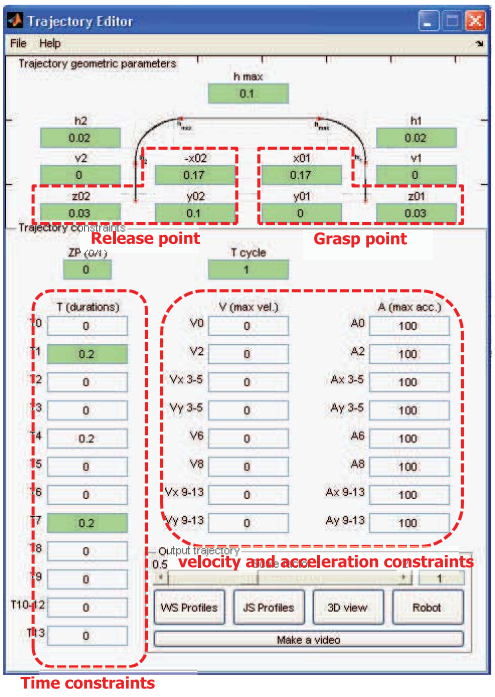


Figure 5. Front-end of the trajectory generator in the Matlab environment.

pose the trajectory generator can be used as standard .m functions and it is possible to verify the functional correctness of the code by exploiting the tools already present in the Matlab environment or other tools designed ad hoc, like the ones presented in the following section.

3. Matlab tools for trajectory definition and analysis

In order to verify the correctness of the C routines which compose the trajectory generator a graphical user interface has been developed in the Matlab environment. This tool, shown in Fig. 5, allows to set the specifications for the desired trajectory, such as grasp and release points, velocity of the conveyor belts, velocity and acceleration limits, and to analyze the resulting motion profiles, in both the workspace and the joint space of the robot. In Fig. 6 the windows that show these profiles are reported. Note that the trajectory defined in the workspace is computed as a function of the time, while the corresponding motions of the robot joint actuators are reported as a function of the angular position θ_m of the master axis. The conversion from time to angle is performed by assuming a given constant velocity of the virtual master. Note that, the workspace profiles are useful for evaluating the compliance of the trajectory with the bounds on velocity and acceleration, while the joint profiles $\theta(\theta_m)$ represent the values that the machine supervisor will send to the actuator's drives. The Matlab application allows to export these values in a CSV (comma-separated values) file. Therefore, the trajectory generator could be directly used from the Matlab environment for computing and storing in a file a trajectory to be used by the machine's controller, even if this is not the main purpose of the application, whose aim is the analysis of the motion. The other tools provided by the Matlab application are reported in Fig. 7. Both the windows show the trajectory in the 3D space. In particu-

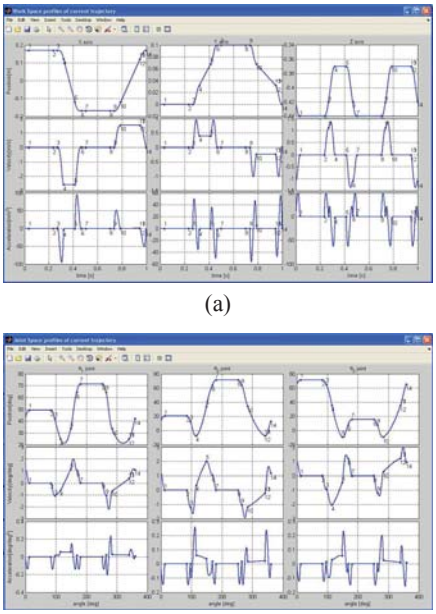
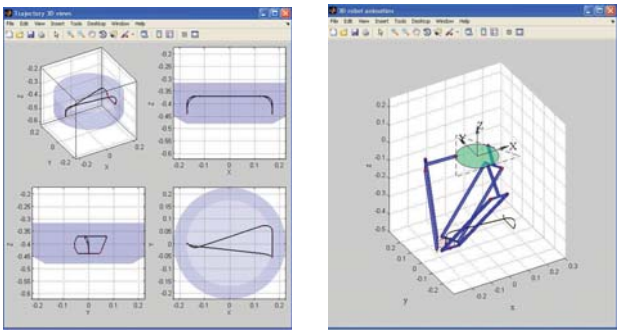


Figure 6. Workspace (a) and joint-space (b) trajectory profiles windows.



(a) (b)

Figure 7. Windows with 3D representation of the trajectory (a) and simulation with the Delta robot (b).

lar, the tool of Fig. 7(a) compares, from different perspectives, the geometric path with the feasible workspace of the robot, while the window of Fig. 7(b) shows the animation of the robot which tracks the trajectory.
As already mentioned, the use of these tools designed in a high-level environment like Matlab has a twofold purpose. On the one hand, the graphical interface allows to quickly check the correctness of the C routines that compose the trajectory generator and to find/solve possible errors or problems. On the other hand, the possibility of modifying the parameters of the trajectory allows to easily test the feasibility of the desired task and eventually to change these parameters, in a interactive manner.

3.1. Dynamic validation of the trajectory

From the Matlab application it is possible to export the trajectory in an external file or in the Matlab workspace. This allows further analyses on the trajectory by means of the toolboxes available in Matlab. In particular, the use of Simulink and SimMechanics [9] can be extremely advantageous for studying the dynamic behavior of the system during the trajectory tracking. The simulative model of the robot has been obtained from a 3D CAD model[2] and imported in Simulink, by exploiting the SimMechanics toolbox. By applying the trajectory to the robot model, as illustrated in Fig. 8, it is possible to estimate the torques that are necessary to perform such a task, see Fig. 9. In this way, before the implementation of the trajectory on the real system it is possible to test if the motion is compliant with the dynamic limits of the actuators. Or vice versa, if the actuators are not available yet, the simulation can be used for sizing it.

4. Conclusion

In this paper the use of CACSD tools (such as Matlab) for rapid prototyping motion trajectories for a high dynamic robot to be integrated in a packaging machine is illustrated. The proposed approach is based on the development of a front-end, which exploits the power of Matlab in terms of analysis and simulation capabilities, directly interacting with the C library containing the routines which define the desired trajectory. In a first phase, this method allowed to rapidly test the C routines from a functional point of view. Currently, the proposed tool is used for customizing the trajectories of the robot, which may change because the differences among the tasks performed by different machines.

[2]Since a detailed 3D model was not available, a simplified model of the robot has been designed in the CAD environment by tacking into account only the main components and inertias of the system.

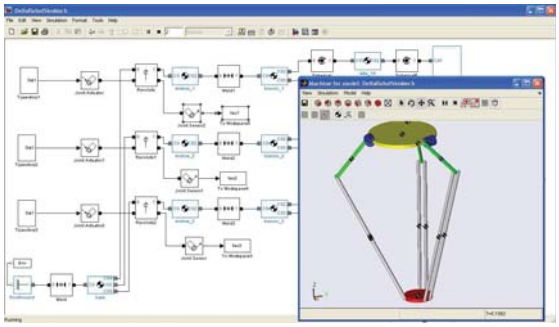


Figure 8. SimMechanics simulative scheme of the robotic system.

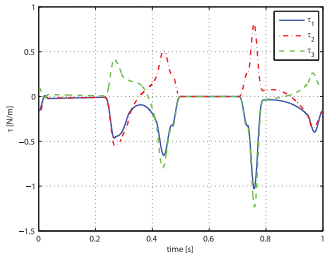


Figure 9. Torques at the robot joints during trajectory tracking.

References

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

[2] C. Bonivento, A. Tonielli, and C. Melchiorri. A pc-based rapid prototyping workstation for the design of motion control systems. In *1st IFAC Conf. on Mechatronic Systems*, Darmstadt, G, Sept. 18-20 2000.

[3] G. Chen, L. Zhai, L. Li, and J. Shi. Trajectory planning of delta robot for dynamic tracking, pick and placement. *Advanced Materials Research*, 680:473–478, 2013. cited By (since 1996)0.

[4] R. Clavel. *Conception d'un robot paralléle rapide á 4 degrés de liberté*. PhD thesis, EPFL, Lausanne, Switzerland, 1991.

[5] http://www.codian robotics.com/en/.

[6] http://www.yaskawa.com/site/Home.nsf/home/home.html.

[7] T. M. Inc. *MATLAB External Interfaces*.

[8] F. Luo and Z. Huang. Embedded c code generation and embedded target development based on rtw-ec. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, pages 532 – 536, Chengdu, July 9-11 2010.

[9] The MathWorks.Inc. *SimMechanics Users Guide*, March 2007.