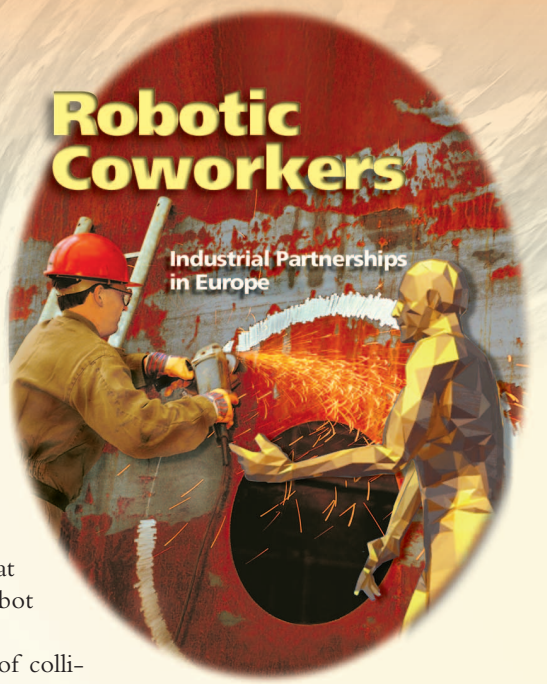
Robotic Polishing Systems

From Graphical Task Specification to Automatic Programming

BY LUIS BASAÑEZ AND JAN ROSELL



The use of robots to automate some tasks involving sensors and motion planning strategies has not been widespread because they are difficult to program. Therefore, there is a pressing need for systems that allow users to easily specify a high-level description of the task (i.e., what is to be done) and that automatically program the robot motions using this description (i.e., how it will be done).

Automatic polishing systems require the generation of collision-free trajectories and the use of compliant control. Most approaches have addressed this problem for robots that polish fixed parts. Nagata and Watanabe [1] propose a joystick-controlled teaching system that allows a polishing robot with an impedance control mode to perform polishing tasks on an object of an unknown shape. Active force control is also used by Wang and Wang [2]; they propose an automated finishing system for polishing a free-form surface using an active force controller mounted on the wrist of an industrial robot equipped with a grinding tool. The system includes a path planning module to plan zigzag and fractal paths on curved surfaces [3]. The control problem is also tackled by Nagata et al. [4]. They present a learning-based surface-following controller, which is based on the original trajectory originated by the cutter location data and an adaptive behavior.

The planning of collision-free paths in cluttered environments is tackled by Takeuchi et al. [5], [6], who developed an automatic programming system for a robot equipped with a polishing tool. This system allows the generation of collision-free paths for the polishing of workpieces that have complicated shapes. An automatic teaching system is proposed in [7] for a three-axis machining center and a two degrees of freedom (DOF) robot. The system is driven by a user-friendly program based on computer-aided-manufacturing (CAM) software to generate 5-DOF numerical control (NC) data. The program also includes a graphic simulator and a teaching mode. The system is expanded in [8] with a monitoring program that allows the operation of the polishing robot from a remote site.

The user interface for the teaching of polishing tasks is further discussed by Balijepalli and Kesavadas [9]. They propose a haptic-based virtual training tool. Finally, Tsai et al. [10] present a complete automatic mold polishing system, which includes a process planner that schedules a sequence of polishing sequences based on the analysis of the manual operation of the polishing procedure; a path planner that computes a

1070-9932/05/$20.00©2005 IEEE

Automatic polishing systems require the generation of collision-free trajectories and the use of compliant control.

path based on the mold curvature and the tool grain size; and a user interface for the specification of the input data and for monitoring by the user during process execution.

This article addresses the automatic programming of robotic polishing tasks from a graphical high-level description of these tasks. Polishing tasks of small parts, which are held by the robot gripper, are considered, and compliance is assumed for the polishing station.

This article presents a formal analysis of the problem and the proposed solution, which is divided into two parts: the task specification module and the task planning module. The task specification module is a graphical user interface (GUI) that allows the user to easily specify the polishing curves over a computer-aided-design (CAD) model of the part. The task planning module finds the time-optimum sequence of collision-free trajectories to execute the task.

Overview

Problem Statement

Let us define the following terms:

- *polishing curve:* the curve over the surface of a part to be polished, representing a strip that must be polished continuously. It is described by an ordered set of reference frames over the surface of the part to be polished. The z-axis of each reference frame is normal to the surface, and the x-axis points to the origin of the next reference frame.
- *polishing station:* the set of locations over a polishing band that allows the same surface finishing. Each location is described by a reference frame.
- *trajectory:* the set of ordered robot configurations (a configuration being defined by the robot joint variables). It can be either a polishing trajectory, which allows carrying out a polishing curve at a given polishing location, or a linking trajectory, which allows the robot to connect two polishing trajectories through a collision-free path.
- *polishing motion sequence:* the sequence of trajectories that allows carrying out all the polishing curves of a part in a minimum amount of time.

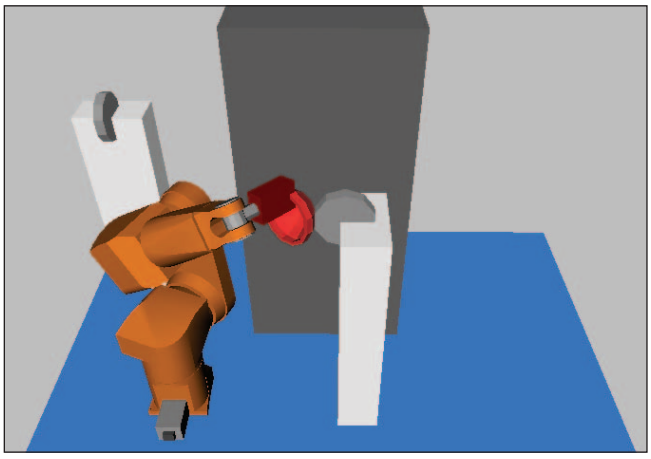


Figure 1. *The polishing cell model.*

The aim of the project is to automatically synthesize the polishing motion sequence from a user-defined graphical description of the polishing curves over a CAD model of the part to be polished. To achieve this objective, three topics must be tackled.

- *Task specification* is the determination, in a user-friendly manner, of the polishing curves over the CAD model of the part.
- *Optimization* is the process of determining the best, in terms of time, feasible sequence of polishing trajectories for a given sequence of polishing curves, taking into account that different trajectories can be used to follow each of the polishing curves. This is due to several reasons, such as the different locations of the selected polishing station, the different solutions of the inverse kinematics, and the two senses in which many curves can be carried out.
- *Path planning* is finding collision-free linking trajectories through the polishing cell.

Proposed Approach

The proposed system is composed of a task specification module and a task planning module.

The task specification module is a GUI that copes with the specification problem. The input file is a CAD model of the part to be polished represented by a triangular mesh. The output of this module is an ASCII file including information about the curves and the grasps.

- Each polishing curve is described by the following parameters:
  - sequence of reference frames
  - allowed execution senses
  - execution speed
  - type of surface finishing
  - width of the polishing band
  - force specification in the z-direction of each reference frame.
- Each grasp is described by a homogeneous transformation relating the reference frame of the part to the reference frame at the wrist of the robot.

The task planning module copes with the optimization and path planning aspects. The input files to this module are

- the polishing cell given by a VRML file with the geometry of a set of convex solids representing the objects in the cell (Figure 1)
- the set of polishing stations given by the corresponding reference frames
- the polishing curves over the CAD model of the part resulting from the task specification module.

The output files of the task planning module are

- the execution file, a program that allows the execution of the task by the robot. The program (in the present case a V+ program for a Stäubli RX-90 robot) is a sequence of motions between robot configurations, defined as joint-space motions for linking trajectories and as Cartesian-space motions for polishing trajectories. The program also includes the force set points that will be sent to the active compliance control of the polishing station.
- the simulation file, a VRML file which contains the robot motions for the simulation of the task.

Task Specification Module

The task specification module was introduced in [11]. The GUI built to implement this module is called the *polishing curves generator* (PCG). It is intended to be a user-friendly tool to specify the polishing curves over a CAD model of the part and can be used by an operator with little computer knowledge. It works using Microsoft Windows and it is programmed in C using the openGL graphics library:

Main Features

Visualization

The model of the part to be polished is a triangular mesh, specified as an input VRML file. The part can be visualized as a solid or wired model and can easily be rotated in any direction (Figure 2).

Specification of the Curve's Parameters.

Before entering the points of a curve, a dialog box appears for selecting the following curve parameters (Figure 3):

- *type of surface finishing:* an identifier of the type of surface finishing
- *velocity:* the linear velocity of the part at the contact point; for a given pressure, an increase in the velocity results in a decrease in material removal
- *pressure:* the force to be exerted by the polishing band at the contact point
- *width:* the width of the polishing band.

The parameters of any existing curve can also be modified from the menu.

Specification of the Curve's Geometry

The points of a curve are selected by positioning the mouse over the CAD model of the part.

A curve is specified as a series of piecewise rectilinear subcurves, each one connecting two consecutive points introduced by the user. Except for the first and last segments of a subcurve, each segment connects the middle point of two edges of a triangle, ensuring that the subcurve is always over the surface of the object.

When the user enters the two points of a subcurve, the Dijkstra algorithm [13] searches for the minimum sequence of triangles that connect the triangles containing these two points. If several minimum sequences exist, the corresponding shortest subcurve is selected.

The current subcurve (i.e., the one the user just defined) is interactively computed as the user drags its end point to the desired final position. The edit menu allows the modification (e.g., changing the last point) or deletion of the current subcurve of any curve. Figure 4 shows a curve composed of two subcurves; the current subcurve is shown in yellow.

Smoothing of the Curves

A smoothing algorithm is applied to each subcurve by moving the segment end points over the edges (Figure 5). Let P be one of these points over a given edge e, and let $\boldsymbol{v}$ and $\boldsymbol{w}$ be the unitary vectors on the subcurve with origin at P.

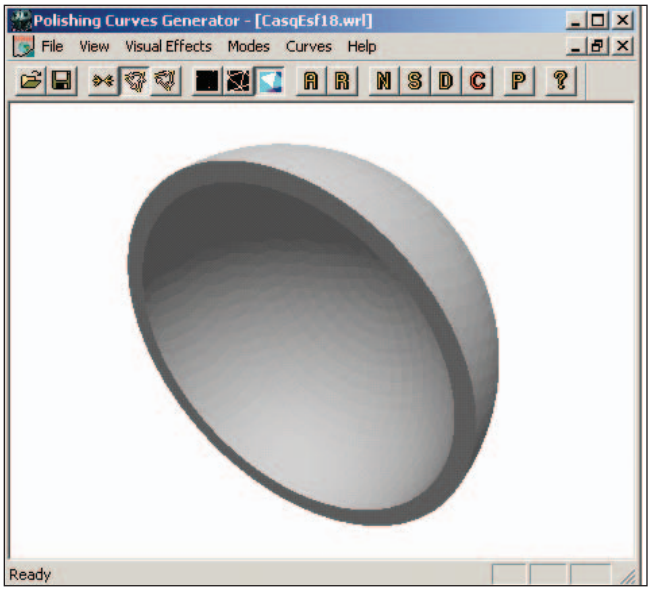


Figure 2. *The GUI.*

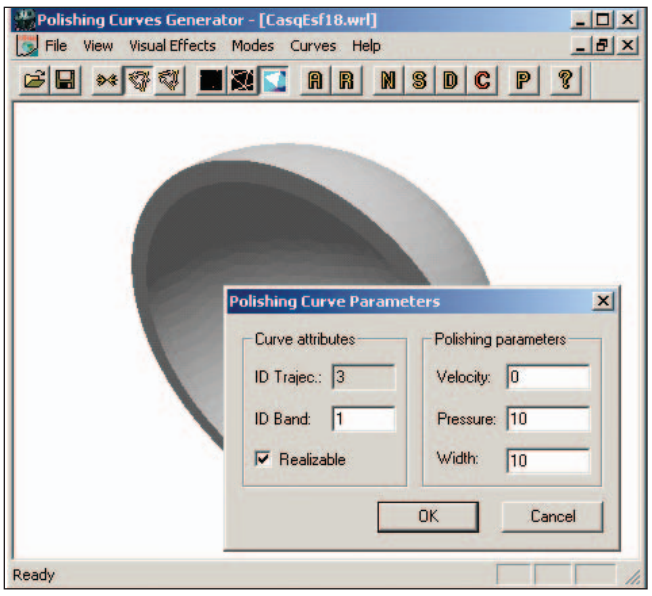


Figure 3. *The specification of polishing curves parameters.*

The polishing curves generator (PCG) is intended to be a user-friendly tool for specifying the polishing curves.

Then P is moved along e in the sense specified by the projection of $\boldsymbol{v}+\boldsymbol{w}$ on e an amount $(|\boldsymbol{v}|+|\boldsymbol{w}|)/2$. The procedure is iteratively applied until its convergence. Figure 6 shows the smoothing of the previously defined polishing curves of Figure 4.

Polishing Strip

The strip f polished by a given band depends on the part material and the specified force and velocity, and it is limited by the width a of the band (Figure 7). The strip is visualized over the part as the user specifies the geometry of the curve and allows the definition of the minimum number of curves to cover the entire part surface (Figure 8). Figure 9 shows the strips polished for two polishing curves, which are defined to be polished at different pressures.

An Example

There are different types of pieces that need a polishing process. Among them, bath taps and door knobs are usually found in robotized polishing workshops. Figure 10 shows the polishing curves defined over a door knob. There are 11 polishing curves with a total of 825 points completely covering the knob surface. The curves have been defined in less than 5 min by a trained user.

Task Planning Module

The task planning module was introduced in [12]. The module synthesizes the robot program in two steps. First, polishing trajectories for each polishing curve are obtained. This is accomplished by considering the polishing locations of the polishing stations that are consistent with the type of surface finishing described for each polishing curve and

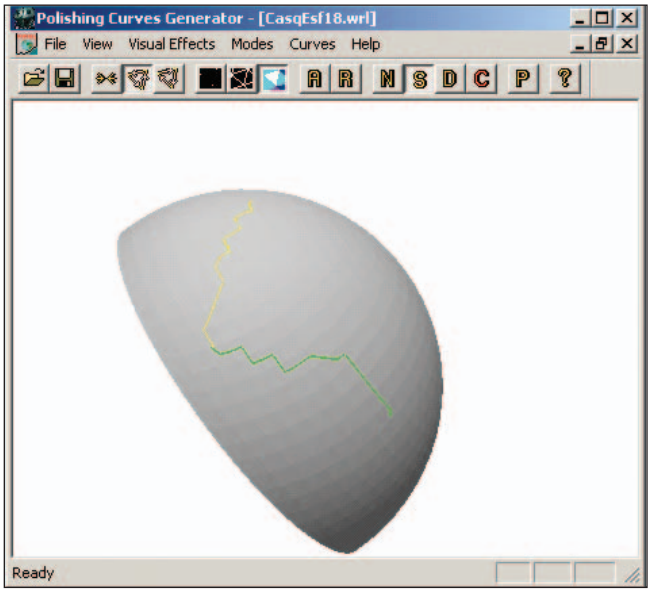


Figure 4. *The specification of polishing curves geometry.*

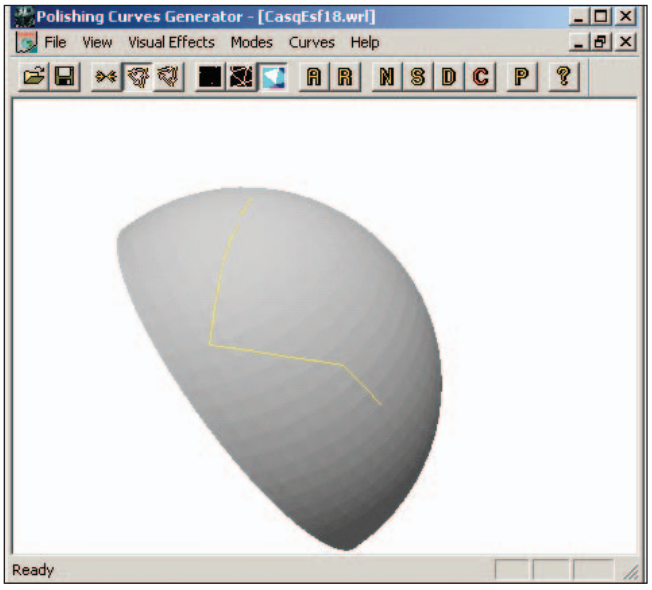


Figure 6. *The curve smoothing results.*

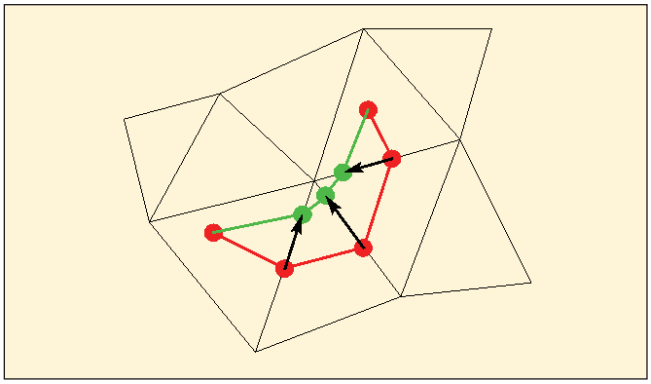


Figure 5. *The curve smoothing procedure.*

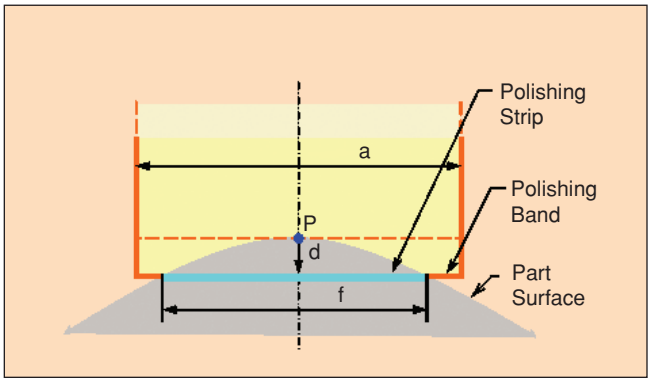


Figure 7. *Polishing strips generation.*

applying the robot inverse kinematics. For each trajectory, the robot's collision-free condition is verified by checking a sample set of its configurations. Then the optimization problem is solved for the obtained polishing trajectories. Second, the obstacle avoidance is tackled for the linking trajectories. The submodules to perform these steps are described in the following subsections.

Sequence Optimization Submodule

The optimization submodule finds the best feasible sequence of polishing trajectories. It initially considers that a linking trajectory is a linear path in joint space connecting the last and the first configurations of two consecutive polishing trajectories; i.e., it does not take into account possible collisions.

The problem of searching the optimum sequence of trajectories can be represented as the problem of searching the path of minimum cost through an oriented graph (Figure 11), where:

- each node represents a feasible trajectory to perform a polishing curve (the nodes are grouped in columns representing the same polishing curve); and the nodes n_i and n_f represent, respectively, the initial and the final configurations
- each arc represents a linking trajectory.

The cost of a trajectory is given by the time needed for the execution of the corresponding motions. It is computed as follows. Let i be the angular motion of joint i for a given linking trajectory and $v_i^{\max}$ its maximum angular velocity. The minimum time to perform the motion of joint i is $t_i = i/v_i^{\max}$. Then, the cost C of a trajectory is

$$C = t_i | t_i \geq t_j \forall i,\ j.$$

If a linking trajectory involves a regrasping operation, its cost is set to a very high value.

The cost of an arc of the graph is the sum of the cost of the linking trajectory it represents and the cost of the previous polishing trajectory, i.e., the one represented by the initial node of the arc.

The topology of the graph allows the use of the Bellman algorithm [14] in order to find the sequence of trajectories with minimum cost.

Due to the presence of obstacles, the path planning submodule later could modify a linking trajectory, leading to a considerable increase in the cost. In this case, all the linking trajectories connecting the same two polishing curves should be, probably, also modified. The costs of the modified linking trajectories replace the initial ones in the graph, and the optimization procedure is executed again.

Path Planning Submodule

The path planning submodule uses a collision map based on an approximate cell decomposition of the configuration space [15] corresponding to the three first joints of the robot and using the I-COLLIDE collision detection library [16] and a modified version of the Fulkerson algorithm [14] for path searching in graphs. It is built for every polishing cell as shown in the following algorithm.

```
Collision-map ()
   1. Partition the configuration space
      into a regular grid.
   2. Verify, for the center of each cell,
      if there is any intersection between
      the objects of the environment and the
      robot, considering the three last
      joints and the grasped part included
      in a sphere.
```

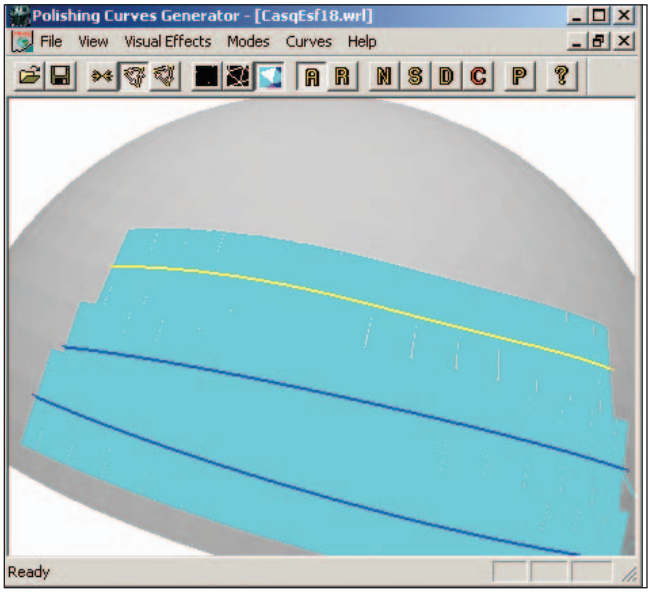


Figure 8. *Polishing strips showing the covered part surface.*

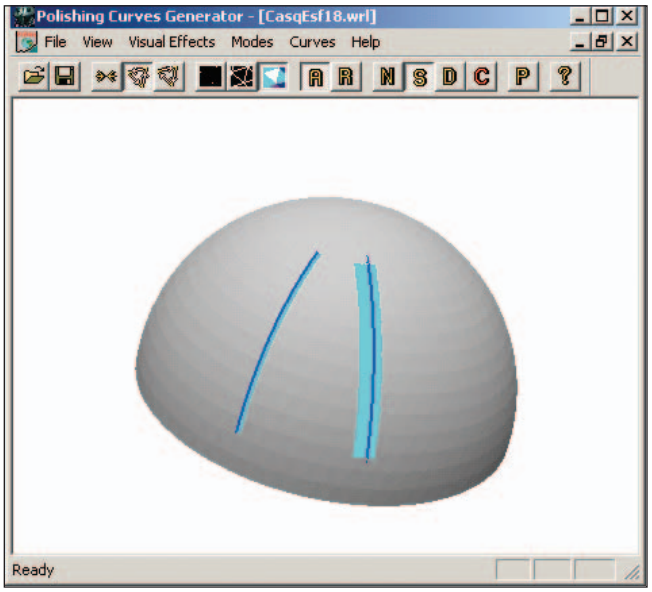


Figure 9. *Polishing strips: the width is dependent on the pressure specified.*

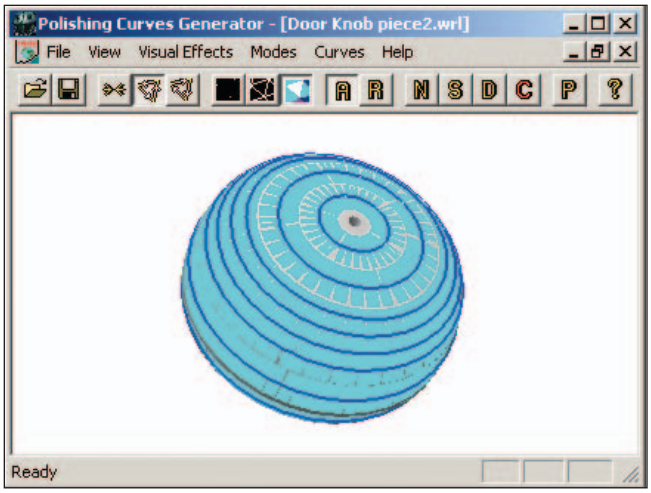


Figure 10. *The polishing curves defined over a door knob.*

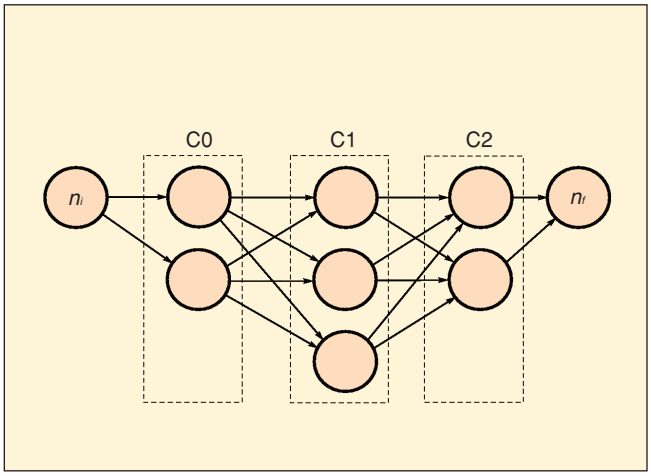


Figure 11. *A graph representing a sequence of three polishing curves, which can be performed by two, three, and two polishing trajectories, respectively.*

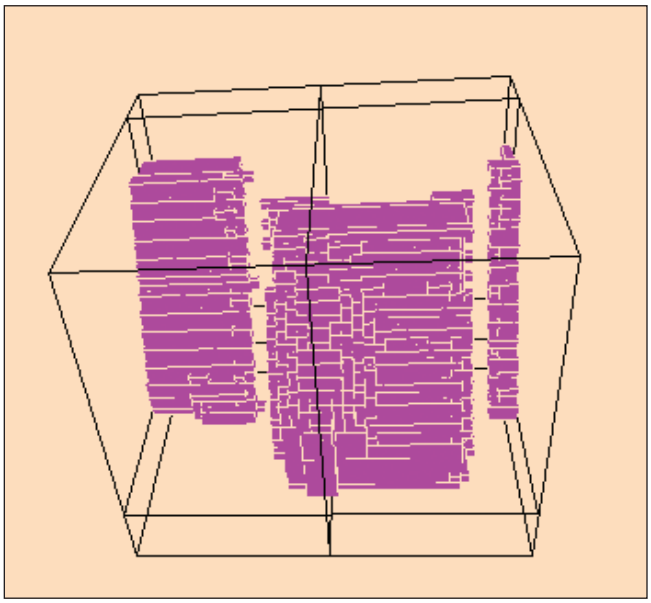


Figure 12. *The configuration space partition.*

3. Mark the cells as free cells if there is no intersection and as collision cells otherwise.
4. Expand the collision space by marking as collision cells those that neighbor any collision cell found in the previous step.
5. Build bigger parallelepiped cells by joining adjacent free cells when possible.
6. Identify free subspaces.
7. Build a graph for each subspace, the nodes being the cells of the partition and the arcs connecting adjacent cells.
8. Find the paths between any two nodes of the graph.

END

As an example, Figure 12 shows a partition of the configuration space with three subspaces. It has been obtained from an initial grid of 64,000 cells.

The *path planning submodule* is devoted to find collision-free paths between the contact configurations corresponding to the end and to the beginning of two consecutive polishing trajectories.

Let c_i and c_f be two such configurations. Let c_i' and c_f' be two configurations located, respectively, at a given distance d from c_i and c_f in the direction of the z-axis of the reference frame of the corresponding polishing location. The distance d is defined by the user.

The path p connecting c_i and c_f is decomposed into:

p_i : rectilinear path in Cartesian space connecting c_i with c_i'.

p_s : a path connecting c_i' with c_f' in joint space.

p_f : a rectilinear path in Cartesian space connecting c_f' with c_f.

These paths will be computed by the path planning algorithm, which uses the following three tools:

1. *Validation tool:* Given a rectilinear path s in joint space, it verifies if s is collision-free.

```
Validate (s)
    Discretize s into a finite set of
    configurations
    FOR each configuration:
       Detect any collision between the
       robot(including the grasped part) and
       the objects of the environment
       IF a collision is detected RETURN
       nonvalid
    END FOR
    RETURN valid
END
```

2. *Smoothing tool:* Given a trajectory p composed of a set of

linear segments, it finds a collision-free smoother trajectory p'.

Smooth (p)
1. Discretize each segment of p into a finite set of configurations
2. Generate a graph with these configurations as nodes, and with the rectilinear paths connecting any two nodes as arcs
3. Apply the Dijkstra algorithm to find the shortest path p' connecting the initial and the final nodes
4. **Validate** (p')
5. IF p' is not valid, eliminate the segments of p' that are not valid and GOTO 3
6. ELSE RETURN p'

END

3. *Search tool:* Given two configurations c_i' and c_f' it finds a collision-free path between them.

Since the collision map is built in a very conservative way, the two configurations c_i' and c_f' will probably not belong to any free cell. However, it is assumed that a free path exists connecting them to a free subspace, since the environment in a polishing cell will not be much cluttered.

Search (c_i', c_f')
1. Find c_i'', c_f'', the two free configurations closest to c_i' and c_f', respectively
2. Obtain the trajectory p' between c_i'' and c_f'' using the **collision map**
3. Obtain the trajectory p by adding to p' the linear segments $\overline{c_i' c_i''}$ and $\overline{c_f'' c_f'}$
4. **Smooth** (p)
5. RETURN p

END

As an example Figure 13 shows the path p searched in the collision map and the corresponding smoothed path p'. Finally, the path planning algorithm is as follows:

Path-Planning (c_i', c_f', d)
Find c_i' and c_f' as the configurations located at a distance d from c_i and c_f, respectively, in the direction of the z-axis of the reference frame of the corresponding polishing location
- p_l: rectilinear path in Cartesian Space between c_i and c_i'
- p_s: rectilinear path in Joint Space between c_i' and c_f'
- p_f: rectilinear path in Cartesian Space between c_f' and c_f

Validate (p_s)
IF p_s is not valid THEN p_s = **Search** (c_i', c_f')
RETURN ($p_i \cup p_s \cup p_f$)
END

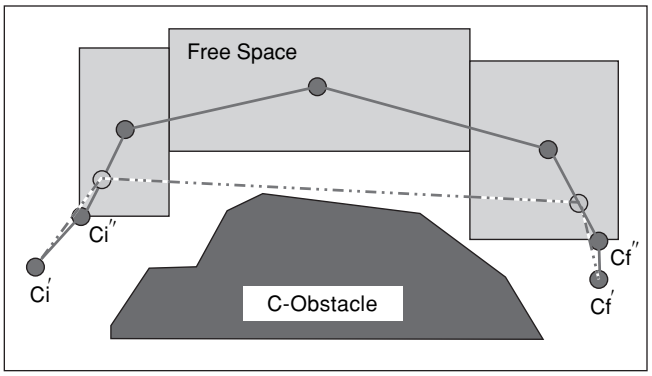


Figure 13. *Trajectory ^{p} obtained from the collision map (continuous line) and trajectory ^{p} (dashed line) obtained by applying the smoothing procedure to ^{p}.*

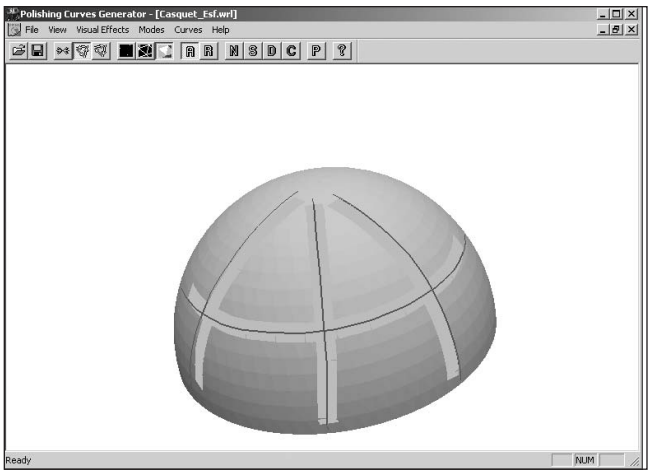


Figure 14. *The polishing curves defined over a semispheric part.*

A Case Study

This section presents the following example:
- The task is performed by a RX-90 Stäubli robot.
- The part to be polished is a semisphere.
- The polishing cell has two polishing stations, each one with only one polishing location, and one obstacle (Figure 1).
- Joint 6 has a finite range.
- The part can only be grasped in one way.

Task Specification Phase

Using the PCG, four polishing curves have been defined over the part (Figure 14).
- The first three, described by ten reference frames, are to be performed at Polishing Station 1.
- The last one is a continuous curve described by 20 reference frames; it is to be performed at Polishing Station 2.
- All of them can be executed in either sense.

The optimization submodule finds the best feasible sequence of polishing trajectories.

Task Planning Phase

Taking into account the inverse kinematics, each of the four polishing curves defined in the specification phase can be performed by 12, 8, 8, and 114 polishing trajectories, respectively.

The program runs in a Silicon Graphics workstation (175 MHz R10000 Indigo2). The two steps of the program synthesis follow.

Optimization Step

The graph is generated in 4.3 s and the optimum polishing trajectory sequence is found in 0.01 s.

Path Planning Step

The collision map is generated with

- an initial partition of 64,000 cells:
  - Joint θ_1: Range divided in 40 intervals (each 8° over a range of 320°).
  - Joint θ_2: Range divided in 40 intervals (each 6.88° over a range of 275°)
  - Joint θ_3: Range divided in 40 intervals (each 7.12° over a range of 285°).
- the radius of the sphere covering the three last joints and the grasped part equals 400 mm.

The collision map is obtained in 43.89 s, distributed as follows:

- The collision detection to determine the free cells is obtained in 40.10 s.
- The connectivity test to generate the subspaces and the grouping algorithm to form bigger free cells is performed in 3.33 s. Three subspaces are obtained, composed of 228, 83, and 55 cells respectively.
- The modified Fulkerson algorithm to find the paths connecting any two cells is performed in 0.44 s, 0.01 s, and 0.01 s, respectively, for each subspace.

The modification of the Fulkerson algorithms allows dealing with rather big graphs, making the grouping algorithm not critical.

The solution path is obtained in 65 s by

- fixing the distance d to 20 mm
- using the collision map to find p_s, which is found to be composed of 20 configurations
- smoothing with a discretization of 100 points per arc
- validating the arcs with a 3° step.

The input and output files of this example, including the simulation, are shown in http://www.ioc.upc.es/usuaris/JanRosell/.

Conclusions

This article presents a graphical task-level robot programming tool for polishing parts held by the robot gripper. The proposed approach allows a user-friendly manner to specify the polishing curves over a CAD model of the part. The user also specifies the width of the abrasive polishing bands to be used and the pressure exerted. The GUI aids the user in verifying that the whole surface is to be correctly polished. Once the task specification is done, a task planning module provides the algorithms to guarantee the time-optimum sequence of robot trajectories to perform the polishing of the curves over the part, avoiding collisions with the obstacles of the cell.

The programming tool has been tested in actual polishing tasks at the industrial company Polits Catalunya using a Stäubli RX-90 robot (Figure 15). An example of the parts used for the test is shown in Figure 16. The proposed approach can be easily extended to other similar tasks like cutting or material dispensing.

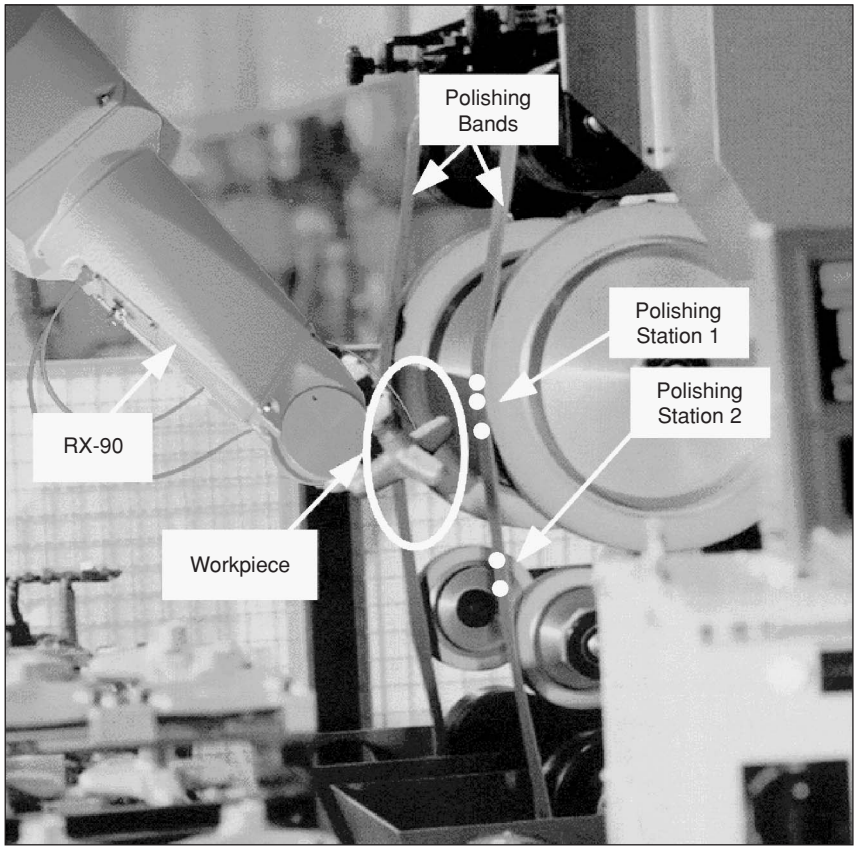


Figure 15. *The RX-90 robot executing a polishing task.*

Acknowledgments

This work was partially supported by the Comisíon Interministerial de Ciencia y Tecnología (CICYT) projects DPI2002-03540 and DPI2004-03104. The authors would like to thank Jaume Gratacòs, Iván Díaz, and Leopold Palomo for the implementation of the programs and Polits Catalunya and Stäubli Robotics for their technical comments during the development of the work.

Keywords

Automatic robot programming, polishing robot system, task planning, trajectory generation.

References

[1] F. Nagata and K. Watanabe, "Teaching system for a polishing robot using a game joystick," in *Proc. 39th SICE Annu. Conf.*, Iizuka, Japan, 2000, pp. 179–184.

[2] Y.T. Wang and C.P. Wang, "Development of a polishing robot system," in *Proc. 7th IEEE Int. Conference Emerging Technol. Factory Automat., ETFA '99*, Barcelona, Spain, 1999, vol. 2, pp. 1161–1166.

[3] Y.T. Wang and Y.J. Jan, "Path planning for robot-assisted grinding processes," in *Proc. IEEE Int. Conf. Robot. Automat.*, Seoul, Korea, 2001, vol. 2, pp. 331–336.

[4] F. Nagata, Y. Kusumoto, K. Watanabe, K. Kiguchi, K. Tsuda, K. Yasyda, K. Yokoyama, M. Umetsu, N. Mori, and M. Omoto, "High precision polishing robot using a learning-based surface following controller," in *Proc. IEEE Int. Symp. Computational Intell. Robot. Automat.*, Fukuoka, Japan, 2003, vol. 1, p. 91–96.

[5] Y. Takeuchi, D. Ge, and N. Asakawa, "Automated polishing process with a human-like dexterous robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, Atlanta, GA, 1993, vol. 3, pp. 950–956.

[6] D. Ge, Y. Takeuchi, and N. Asakawa, "Dexterous polishing of overhanging sculptured surfaces with a 6-axis control robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, Nagoya, Japan, 1995, pp. 2090–2095.

[7] M.C. Lee, S.J. Go, J.Y. Jung, and M.H. Lee, "Development of a user-friendly polishing robot system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Kyongju, Korea, 1999, vol. 3, pp. 1914–1919.

[8] S.J. Go, M.C. Lee, and B.S. Kim, "User-friendly automatic polishing robot system and its remote operation based on network," in *Proc. IEEE Int. Symp. Industrial Electron.*, Pusan, Korea, 2001, vol. 3, pp. 1435–1440.

[9] A. Balijepalli and T. Kesavadas, "A haptic based virtual grinding tool," *Proc. 11th Symp. Haptics Interfaces Virtual Environ. Teleoperator Syst. HAPTICS'03*, Los Angeles, CA, 2003, pp. 390–396.

[10] M. Tsai, J.-L. Chang, and J.-F. Haung, "Development of an automated mold polishing system," in *Proc. IEEE Int. Conf. Robot. Automat.*, Taipei, Taiwan, 2003, pp. 3517–3522.

[11] J. Rosell, L. Basañez, and I. Díaz, "Graphical task-level robot programming for polishing and grinding," in *Proc. 15th IFAC World Congress*, Barcelona, Spain, 2002.

[12] J. Rosell, J. Gratacòs, and L. Basañez, "An automatic programming tool for robotic polishing tasks," in *Proc. IEEE Int. Symp. Assembly Task Planning ISATP'99*, Porto, Portugal, 1999, pp. 250–255.

[13] E.W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[14] B. Roy, *Algèbre moderne et théorie des graphes*. Paris: Dunod, 1970.

[15] J.C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

[16] J. Cohen, M. Lin, D. Manocha, and K. Ponamgi, "I-collide: An interactive and exact collision detection system for large-scaled environments," in *Proc. ACM Symp. 3D Graphics Interactive.*, Monterrey, CA, 1995, pp. 189–196.

Figure 16. *An example of polishing parts used in the programming tool test.*

Luis Basañez received a Ph.D. degree in electrical engineering from the Technical University of Catalonia (UPC), Barcelona, Spain, in 1975. From 1976–1987 he was vice-director of the Institute of Cybernetics (UPC) and director from 1987–1990. Since 1986, he has been a full professor of system engineering and automatic control at the UPC and, since 1990, head of the robotics division of the Institute of Industrial and Control Engineering (UPC). He served as a member of the executive committee of the International Federation of Robotics (IFR) from 1987–1992, and now he is the Spanish delegate at the IFR. In 2005 he was appointed a fellow of International Federation of Automatic Control (IFAC). His present research interests includes task planning and the coordination of multirobot systems, sensor integration, and active perception.

Jan Rosell received his B.S. in telecomunication engineering and Ph.D. in advanced automation and robotics from the Technical University of Catalonia, Barcelona, Spain, in 1989 and 1998, respectively. In 1992, he joined the Institute of Industrial and Control Engineering, where he has developed research activities in robotics. He has been involved in teaching activities in automatic control as an assistant professor since 1996 and as an associate professor since 2001. His current technical areas include automatic programming, robotic assembly, and manufacturing automation.

Address for Correspondence: Luis Basañez, IOC-UPC, Avda. Diagonal 647, planta 11, 08028 Barcelona, Spain. Phone: +34 934017161. Fax: +34 934016605. E-mail: luis.basanez @upc.edu.