From Graphical Task Specification to Automatic Programming in Robotic Polishing Systems

The use of robots to automate some tasks involving sensors and motion planning strategies is not yet widely extended due to the difficulty of their programming. Therefore, there is the need of systems that allow the user to easily specify a high level description of the task (i.e. what is to be done) and that allow to automatically program the robot motions from this description (i.e. how will it be done).

Automatic polishing systems require the generation of collision-free trajectories and the use of compliant control. Most of the approaches have addressed this problem for polishing robots working on fixed parts. Nagata and Watanabe [1] propose a joystick controlled teaching system that allows a polishing robot with impedance control mode to perform polishing tasks on an object with unknown shape. Active force control is also used in Wang and Wang [2], that 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 planning of collision-free paths is cluttered environments is tackled in Takeuchi et al. [4,5], where an automatic programming system is developed for a robot equipped with a polishing tool. The system allows the generation collision-free paths for the polishing of workpieces with complicated shapes. An automatic teaching system is proposed in [6] for a three-axis machining centre and a two degrees of freedom robot. The system is driven by a user-friendly program based on a CAM software to generate 5 d.o.f. NC data. The program also includes a graphic simulator and a teaching mode. The system is extended in [7] with a monitoring program that allows to operate the polishing robot from a remote site.

This paper addresses the problem of the automatic programming of robotic polishing tasks from a graphical high-level description of these tasks. We consider polishing tasks of small parts, which are held by the robot gripper. Compliance is assumed on the polishing station.

The paper presents a formal analysis of the problem and the solution proposed, which is decomposed in two parts: the task specification module and the task planning module. The task specification module is a graphical user interface that allows the user to easily specify the polishing curves over a 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:

- Polishing curve: 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 a 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: Set of locations over a polishing band which allows the same surface finishing. Each location is described by a reference frame.
- Trajectory: Set of ordered robot configurations, a configuration being defined by the joint variables. They can be either a polishing trajectory, that allows to follow a polishing curve at a given polishing location, or a linking trajectory, that allows the robot to connect two polishing trajectories through a collision-free path.
- Polishing motion sequence: The sequence of trajectories which allows to follow all the polishing curves of a part in minimum time.

The aim of this 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, the three following topics must be tackled:

- **Task Specification**: Determination of the polishing curves over the CAD model of the part in a user-friendly way.
- **Optimization**: Finding the best 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 like the different locations of the selected polishing station, the different solutions of the inverse kinematics or the two senses in which many curves can be followed.
- **Path planning**: 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 graphics user interface 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 the specification 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.

- Set of grasps each one 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 description of 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).

- Set of polishing stations.

- The description of the polishing curves over the CAD model of the part resulting from the specification module.

The output files of the planning module are the following:

- 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, with force references to be used by an active compliant polishing station.

- Simulation file: A VRML file which contains the robot motions for the simulation of the task.

Task specification module

The task planning module was introduced in [8]. The graphics user interface built as task specification module is called 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, able to be used by an operator with few computer knowledge. It works under Windows and it is programmed in C using the openGL graphics library:

Main Features

- **Visualization:** The model of the part to be polished is represented by a triangular mesh specified as an input VRML 1.0 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 curves parameters:** Before entering the points of a curve, a dialog box appears in order to select the following parameters of the curve (Figure 3):
    - Type of surface finishing: Identifier of the type of surface finishing.
    - Velocity: 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: Force exerted by the polishing band at the contact point.
    - Width: Width of the polishing band.

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

- **Specification of the curves geometry:** The selection of the points of a curve is done with the mouse over the CAD model of the part. The middle points of the triangles of the triangular mesh are represented as nodes of a graph. Each arc of the graph is composed of two straight segments over the triangles connecting the corresponding nodes through the middle point of the border edge. A curve is specified as a set of subcurves, which are the segments connecting two consecutive points introduced by the user. When the user enters the two points of a subcurve, they are included as nodes of the graph and the subcurve connecting them is generated using the Dijkstra algorithm [9] to search the minimum-distance path. The nodes of the solution path are called internal points of the subcurve.

  The edit menu allows to add or to delete the final subcurve of any curve. Figure 4 shows a curve composed of two subcurves. The current subcurve (i.e. the one being just defined by the user) is interactively computed as the user drags its end point to the desired final position.

- **Smoothing of the curves:** A smoothing algorithm is applied to each subcurve, by moving the points on the edges (Figure 5). Let P be one of these points over a given edge e, and let $\vec{v}$ and $\vec{w}$ be the unitary vectors on the subcurve with origin at P. Then P is moved along e in the sense specified by the projection of $\vec{v}+\vec{w}$ on e an amount $\left(|\vec{v}|+|\vec{w}|\right)/2$. The procedure is iteratively applied until its convergence. Figure 6 shows the smoothing of the previous defined polishing curves of Figure 4.

- **Polishing strip:** The strip polished by a given band depends on the part material, the specified force and the velocity, and it is limited by the width of the band. The strip is visualized over the part as the user specifies the geometry of the curve, which allows the definition of the minimum number of curves to cover all the part surface, as shown in Figure 7. Figure 8 shows the strips polished for two polishing curves that are defined to be polished at different pressure.

An Example

There are different types of pieces that need a polishing process. Among them, bath tabs and door knobs are some of the more usually found in robotized polishing workshops. Figure 9 shows the polishing curves defined over a door knob. There are eleven polishing curves with a total of 825 points that totally cover the knob surface. The curves have been defined in less than five minutes by a trained user.

Task planning module

The task planning module was introduced in [10]. Its main characteristics are the following. The module synthesizes the robot program in two steps:

- First, collision-free polishing trajectories for each polishing curve are obtained. This is done considering the polishing locations of the polishing stations that are consistent with the type of surface finishing described for each polishing curve, and applying the robot inverse kinematics[1]. 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 module 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 10), where:

- Each node represent 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.

[1] Collision-free condition is verified, for each trajectory, by checking a sample of the its configurations.

- The arcs represent linking trajectories.

The cost of a trajectory represents the time needed for the execution of the corresponding motions. It is computed as follows. Let Δq_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 $t_i = \Delta q_i / v_i^{\max}$. Then, the cost C of a trajectory is:

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

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

The cost of the arcs 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 Bellmann algorithm [11] in order to find the sequence of trajectories with minimum cost.

Due to the presence of an obstacle, the path planning module could modify a linking trajectory with 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 in the graph to the initial ones, and the optimization procedure is executed again.

Path planning submodule

The path planning module uses a collision map based on an approximate cell decomposition of the Configuration Space [12] corresponding to the three first links of the robot. 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. Use I-COLLIDE [13] to verify if for the centre of each cell 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. [] Mark the cells as free cells if there is no
       intersection, and as collision cells otherwise.
    3. Expand the collision space by marking as collision cells those
       that are neighbour to any collision cell found in the previous
       step.
    4. Built bigger parallelepiped cells by joining adjacent free cells
       when possible.
    5. Identify free subspaces.
    6. Built a graph for each subspace, the nodes being the cells of the
       partition and the arcs connecting adjacent cells.
    7. Find the paths between any two nodes of the graph (using a
       modified version of the Fulkerson algorithm [11])
END
```

As an example Figure 11 shows a partition of the Configuration Space with three subspaces. It has been obtained from an initial grid of 64,000 cells (Section 4.3).

The path planning module 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 of 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.
    each segment of p into a finite set of configurations
    FOR each configuration:
        Use I-COLLIDE [13] to detect any collision between the robot
        (including the grasped part) and the objects of the environment.
        IF a collision is detected RETURN non-valid
    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 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 very cluttered.

```
Search($c'_i$, $c'_f$)
    1. Find two configurations $c''_i$, $c''_f$ a free subspace, which are,
       respectively, the closest configurations to $c'_i$ and $c'_f$.
    2. $p' =$ trajectory between $c''_i$ and $c''_f$ obtained from the collision
       map
    3. $p = p' \cup \overline{c'_i c''_i} \cup \overline{c''_f c'_f}$
    4. Smooth($p$)
    5. RETURN $p$
END
```

As an example Figure 12 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.
            $p_i$: 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
```

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).
- Four polishing curves have been described over the part (Figure 13):
    - o The first three, described by 10 reference frames, are to be performed at polishing station number 1.
    - o The last one is a continuous curve described by 20 reference frames; it is to be performed at polishing station number 2.
    - o All of them can be executed in either sense.
- Joint 6 has a finite range.
- The part can only be grasped in a way.

As a result of these initial conditions, and taking into account the inverse kinematics, each polishing curve can be performed by 12, 8, 8 and 114 polishing trajectories, respectively.
The program runs in a Silicon Graphics workstation (175 MHz R10000 Indigo^2). The two phases of the program synthesis are the following.

Otimization phase:
The graph is generated in 4.3 seconds and the optimum polishing trajectory sequence is found in 0.01 seconds.

Path planning phase:

The collision map is generated with:

- An initial partition of 64,000 cells:
    - o Joint q_1: Range divided in 40 intervals (each 8º over a range of 320º).
    - o Joint q_2: Range divided in 40 intervals (each 6.88º over a range of 275º).
    - o Joint q_3: Range divided in 40 intervals (each 7.12º over a range of 285º).
- A radius of the sphere covering the three last joints and the grasped part equal to 400 mm.

The collision map is obtained in 43.89 seconds distributed as follows:

- The collision detection to determine the free cells is obtained in 40.10 seconds.
- The connectivity test to generate the subspaces and the grouping algorithm to form bigger free cells is performed in 3.33 seconds. 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, 0.01 and 0.01, respectively, for each subspace.

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

The solution path is obtained in 65 seconds 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 step less than 3 degrees.

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

Conclusions

This paper 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 graphical user interface 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 proposed approach can be easily extended to other similar tasks like cutting or material dispensing.

References

[1] F. Nagata and K. Watanabe, “Teaching system for a polishing robot using a game joystick," in *Proc. of the 39th SICE Annual Conference*, pp.179 -184, 2000.

[2] Y. T. Wang and C. P. Wang, “Development of a polishing robot system," in *Proc. of the 7th IEEE Int. Conference on Emerging Technologies and Factory Automation, ETFA '99*, vol. 2, pp. 1161 -1166, 1999.

[3] Y. T. Wang and Y. J. Jan, “Path planning for robot-assisted grinding processes," in *Proc. of the EEE Int. Conference on Robotics and Automation*, vol. 2, pp. 331-336, 2001.

[4] Y. Takeuchi, D. Ge, and N. Asakawa, “Automated polishing process with a human-like dexterous robot," in *Proc. of the IEEE Int. Conference Robotics and Automation*, vol. 3, pp. 950 -956, 1993.

[5] D. Ge, Y. Takeuchi, and N. Asakawa, “Dexterous polishing of overhanging sculptured surfaces with a 6-axis control robot," in *Proc. of the IEEE Int. Conference Robotics and Automation*, pp. 2090 -2095, 1995.

[6] M. C. Lee, S. J. Go, J. Y. Jung, and M. H. Lee, “Development of a user-friendly polishing robot system," in *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, vol. 3, pp. 1914 -1919, 1999.

[7] 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. of the IEEE Int. Symp. on Industrial Electronics*, vol. 3, pp. 1435 -1440, 2001.

[8] J. Rosell, L. Basañez, and I. Díaz, “Graphical task-level robot programming for polishing and grinding," in *Accepted to the 15th IFAC World Congress*, 2002.

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

[10] J. Rosell, J. Gratacòs, and L. Basañez, “An automatic programming tool for robotic polishing tasks," in *Proc. of the IEEE Int. Symp. On Assembly and Task Planning ISATP'99*, pp. 250-255, 1999.

[12] B. Roy, *Algèbre moderne et théorie des graphes*, Dunod, 1970.

[12] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic Press, 1991.

[13] J. Cohen, M. Lin, D. Manocha, and K. Ponamgi, “I-collide: An interactive and exact collision detection system for large-scaled environments," *Proc. of the ACM Int. 3D Graphics Conference*, pp. 189-196, 1995.

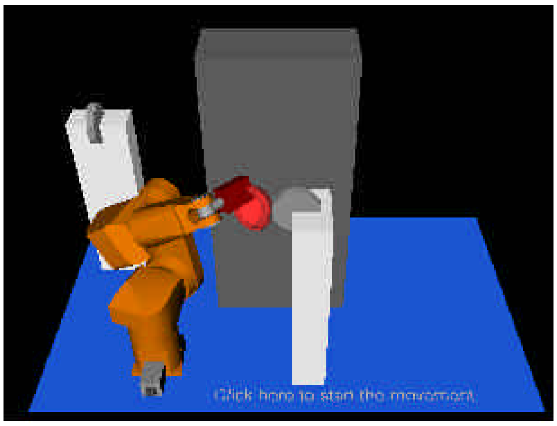


Figure 1. Polishing cell

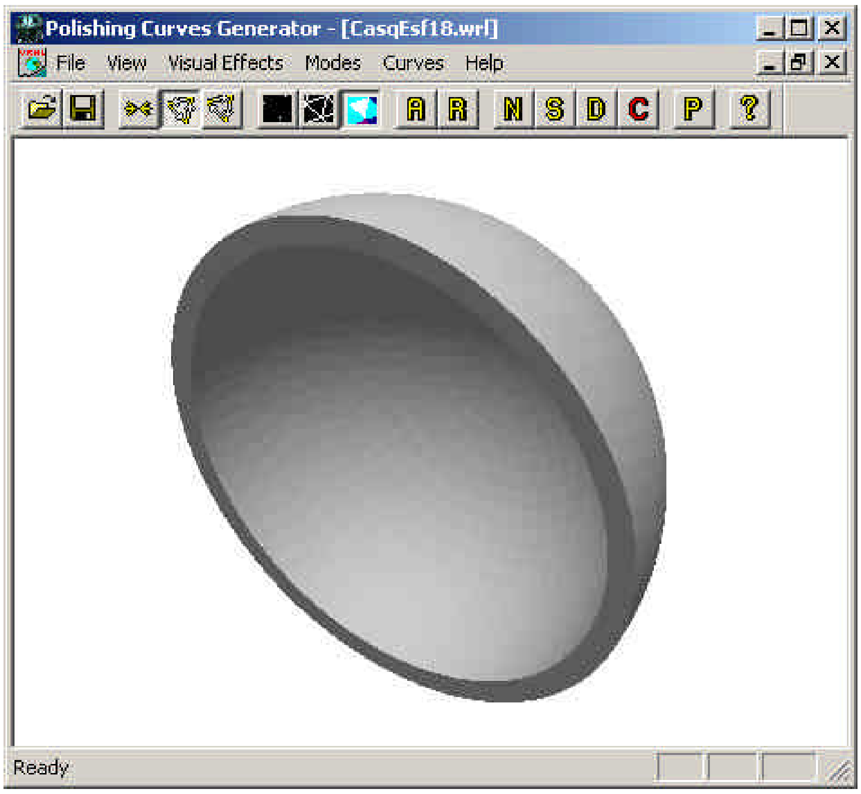
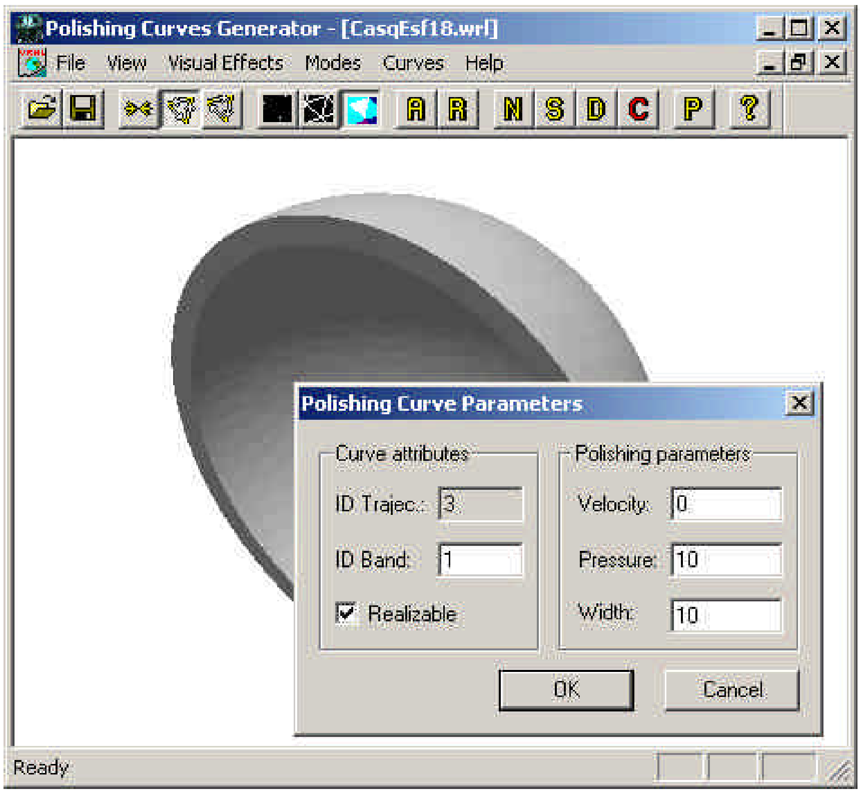


Figure 2. Graphics user interface


Polishing Curves Generator - [CasqEsf18.wrl]
File View Visual Effects Modes Curves Help
Polishing Curve Parameters
OK
Cancel
Ready

Curve attributes

ID Trajec.: 3

ID Band: 1

[✓] Realizable

Polishing parameters

Velocity: 0

Pressure: 10

Width: 10

Figure 3. Specification of polishing curves parameters.

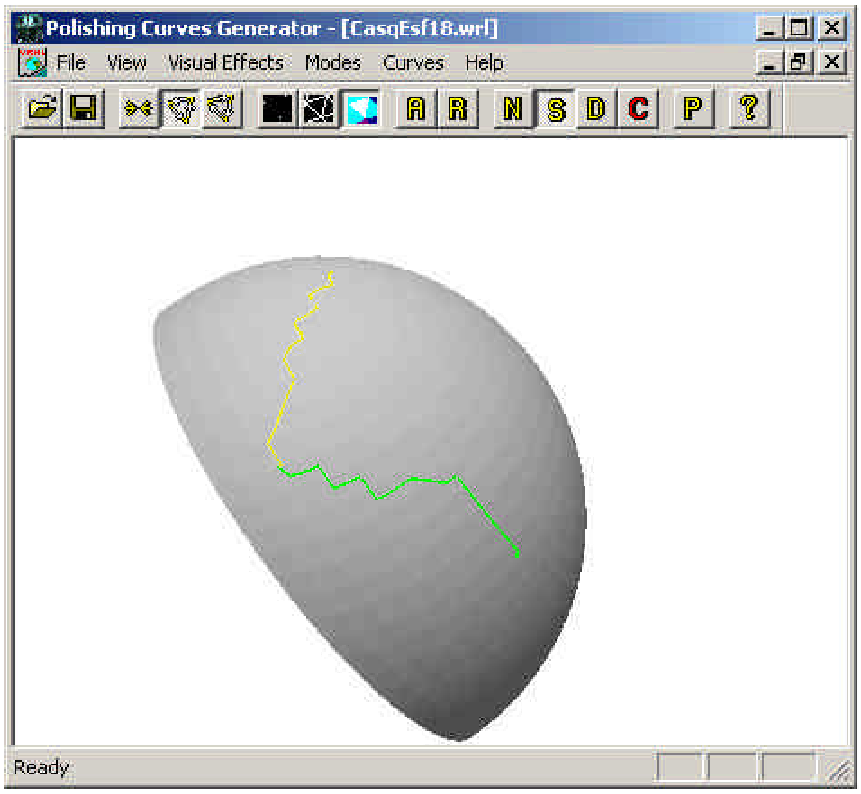

Polishing Curves Generator - [CasqEsf18.wrl]
File View Visual Effects Modes Curves Help
Ready

Figure 4. Specification of polishing curves geometry.

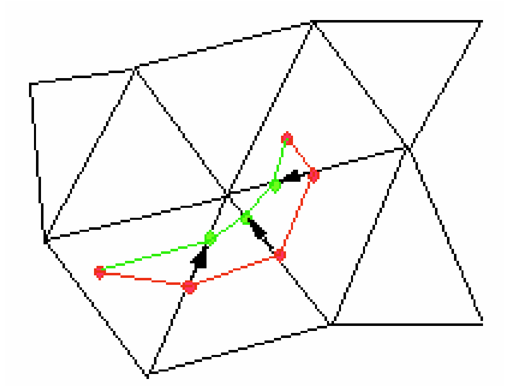


Figure 5. Curve smoothing procedure

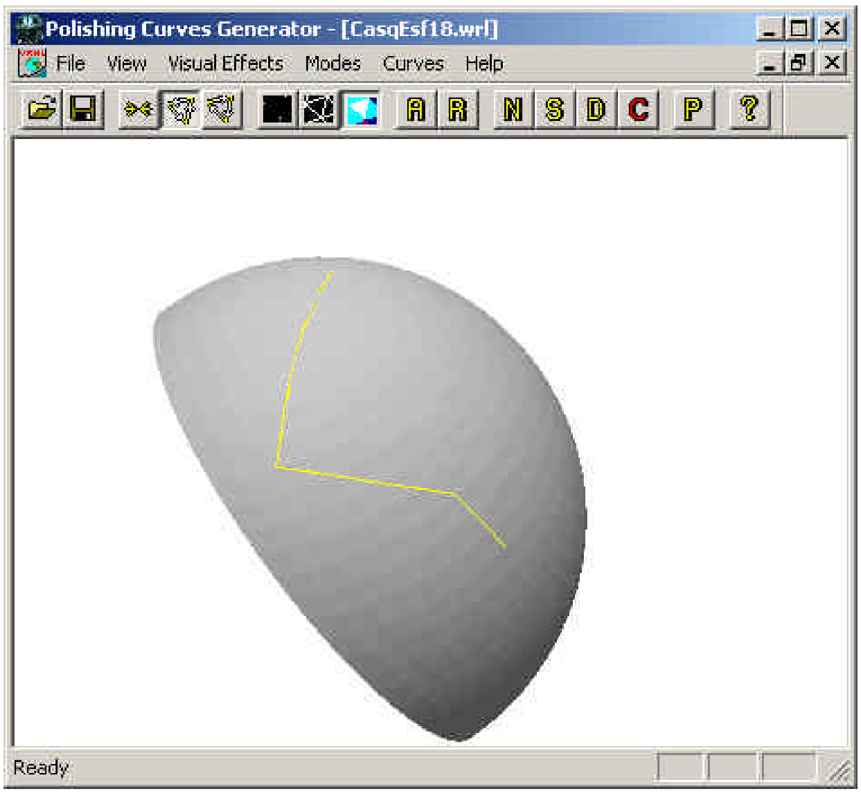


Figure 6. Curve smoothing results.

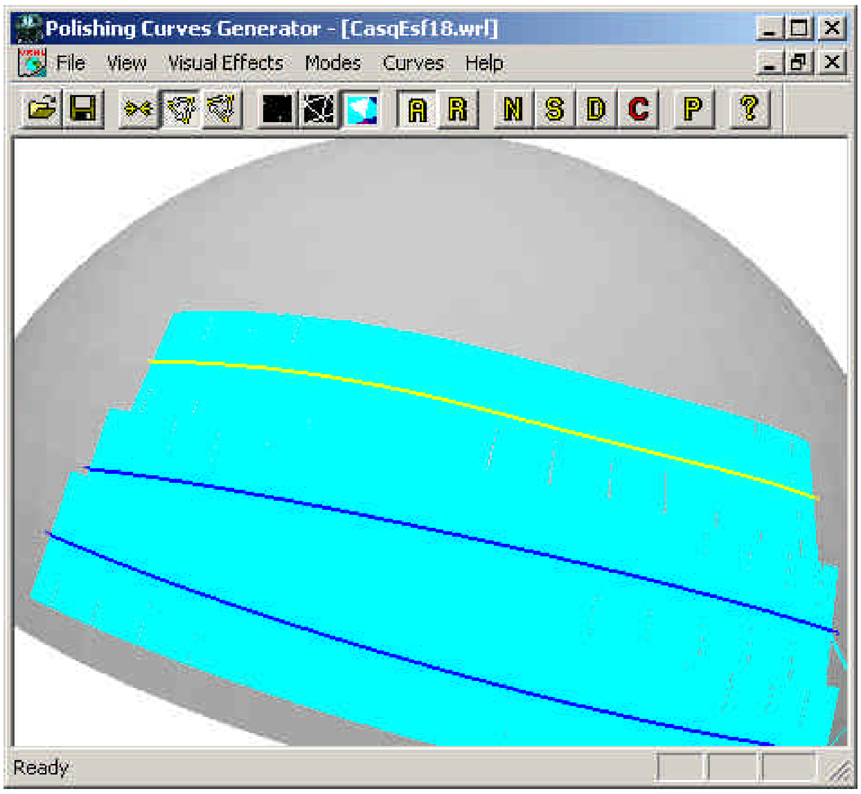

Polishing Curves Generator - [CasqEsf18.wrl]
File View Visual Effects Modes Curves Help
A R N S D C P ?
Ready

Figure 7. Polishing strips showing the part surface being covered.

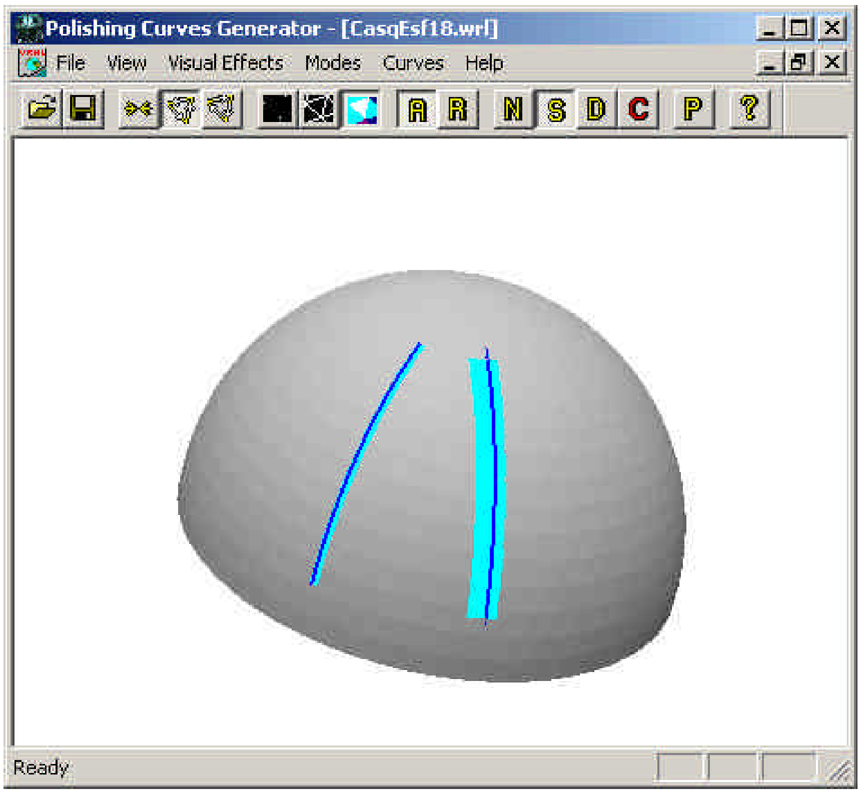

Polishing Curves Generator - [CasqEsf18.wrl]
File View Visual Effects Modes Curves Help
A R N S D C P ?
Ready

Figure 8. Polishing strips: the width is dependant on the pressure specified.

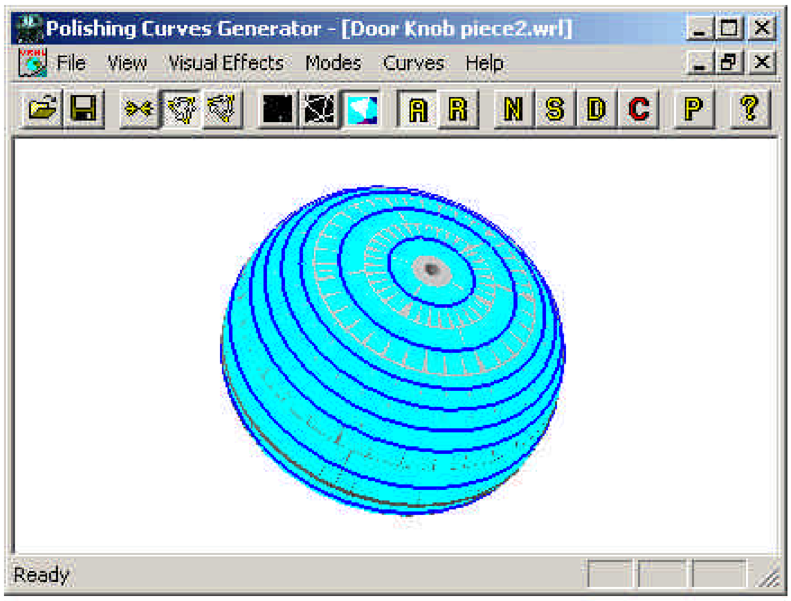


Figure 9. Polishing curves defined over a door knob.

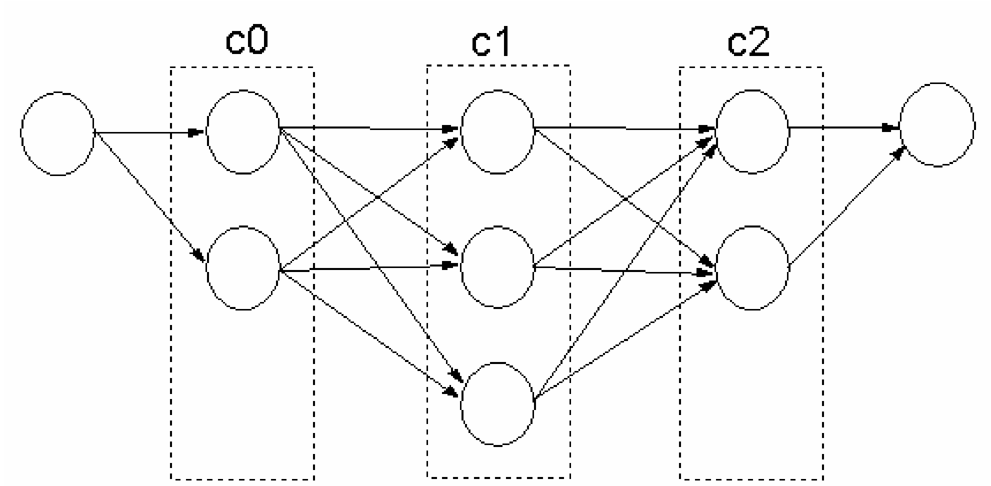


Figure 10. Graph representing a sequence of three polishing curves, which can be performed by two, three and two polishing trajectories, respectively.

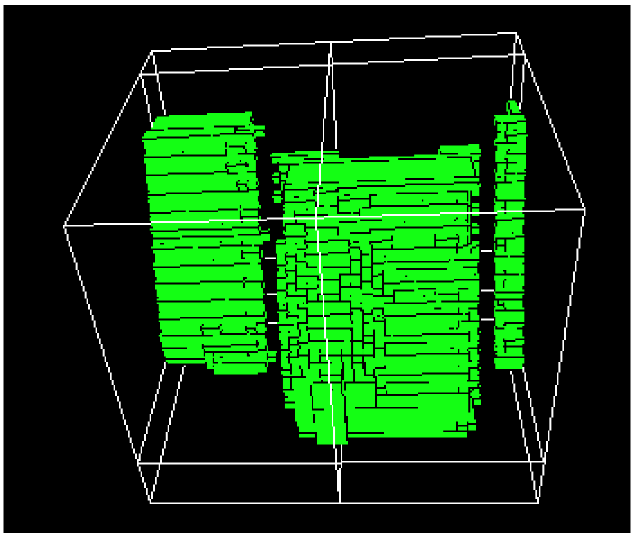


Figure 11. Configuration space partition.

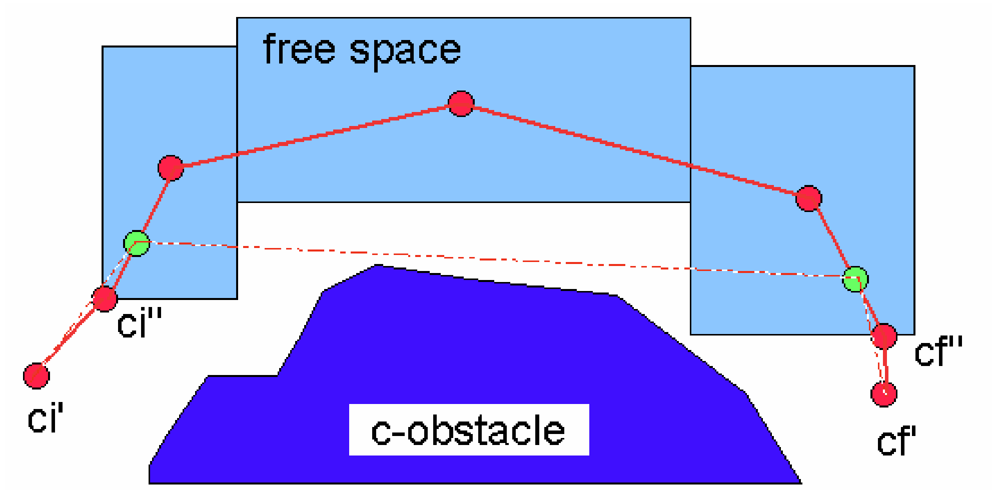


Figure 12. Trajectory p obtained from the collision map (continuous line) and trajectory p' (dashed line) obtained by applying the smoothing procedure to p .

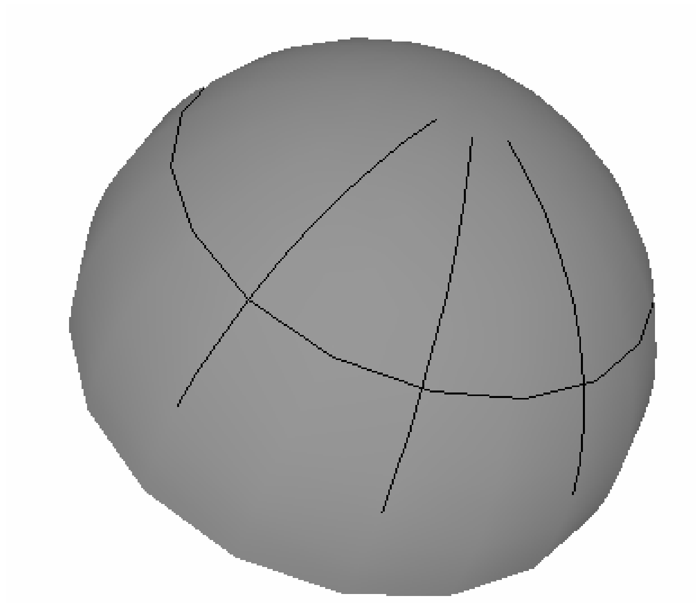


Figure 13. Polishing curves defined over a semispherical part.