AUTOMATIC GENERATION OF 3D MACHINING SURFACES WITH TOOL COMPENSATION FROM GRAYLEVEL IMAGE MODELS

Theodor Borangiu, Anamaria Dogar, Alexandru Dumitrache

University Politehnica of Bucharest
Dept. of Automation and Applied Informatics

Abstract: The paper presents a theoretical approach and implementing issue for automatic CNC toolpath generation using morphological image processing operators applied to depth map images. This method allows computing cutter compensation for various end mill shapes. Roughing is made using zigzag cuts with a flat end mill, and finishing is done with isoparametric toolpaths using various rounded end mills. An AI-based algorithm and image processing software create CNC machining instructions from depth map image models. The paper also describes a graphical user interface and includes experimental results. *Copyright © 2007 IFAC*

Keywords: image processing, computer aided design, CAD/CAM models, CNC, machining.

1. INTRODUCTION

This work presents an isoparametric toolpath generation method with cutter compensation for 3-axis CNC milling machines, using a 2.5D point-based surface representation.

A manufacturing process planning includes machining, assembly and inspection (Park, 2005). In part machining process, a phase-based approach is very common to shape a work piece into its final form. The sequence therefore typically follows the template: roughing, semi-finishing and finishing. The most important objectives for the finishing phase are: meeting the tolerances and surface finish. These objectives are opposite to the objectives of the roughing stage. Rough machining consists in fast and efficient material removal. The tool path generation for the roughing operations is almost always done in 2.5D mode. This means that the geometry of the volume to be machined (called the 'delta volume') is sliced by a set of parallel planes and that in each plane a 2D toolpath is calculated (contour-offset, zig, zig-zag, etc.). The intermediate step to move between two successive parallel planes (.5 dimension) makes that the tool paths in these planes are not connected which results in the generation of the 'stairs' shape (Lefebvre and Lauwers, 2005).

The automated generation of machining tool path is a challenging issue. There are four general types of toolpaths: point-to-point, profiling, pocketing and roughing, and surface machining (Elber, 1995).

The type of object representation determines what geometric information is readily available and what must be computed. It also determines what information can realistically be computed. For example: some representations support estimation of tool load by giving an estimate of the width and depth of cut for a given tool move. Other representations support the generation of partial or cleanup paths that send the cutter into only the areas of the model containing material not removed. The choice of object representation therefore determines the types of path generation algorithms that are possible. There are four general types of object representation: point-based, curve-based, surface-based and solid-based.

Point-based representations include both pixel based representations and formulations based on non-uniform point sampling. *Curve-based* representations involve planar profile geometry for pocketing and roughing. *Surface-based* representations include common surface schemes such as tensor product, B-spline surfaces, trimmed NURBS surfaces as well as surface approximations such as piecewise linear triangular mesh approximations. Solid representations include Constructive Solid Geometry (CSG) and B-Rep solids octree models, and even tensor product solids (Dragomatz and Mann, 1997).

2. SURFACE REPRESENTATION FROM DEPTH MAP IMAGES

Surfaces that can be described by an arbitrary explicit function $z = f(x, y)$ may be stored as greyscale images, where the pixel's coordinates in the image map to the real variables x and y, and the grey level encodes the z value. Usually the black colour maps to the lowest z level and the white colour maps to the highest. Because pixel coordinates are integers, a pixel-to-mm ratio must be used.

Most computer graphics software use 8 bit greyscale images and high end ones can use 16 bit greyscale images. For 16 bit images, the z level can have 65536 discrete values, and this means, for a 100 millimetre tall work piece, a maximum precision of 0.0015 mm. The precision in XY plane is given by the image resolution. In order to get a precision of 0.01 mm for a 100x100 mm part, a 10000x10000 pixels image is needed. This makes the pixel model suitable only for low-precision applications, such as free-form artwork pieces.

The depth map images can be obtained through passive acquisition techniques like laser scanners, and active acquisition techniques like structured light or generated with 3D modelling software packages.

In this work, the 3D objects were modelled using the freeware software POV-Ray (Fig. 1). To create a depth map image, an orthographic camera is used, the texture and light sources are removed, and a gradient pigment is applied. The gradient is along the camera orientation axis. The gradient pigment is scaled and translated such that the farthest visible point from the camera maps to pure black and the closest point maps to pure white (Fig. 2).

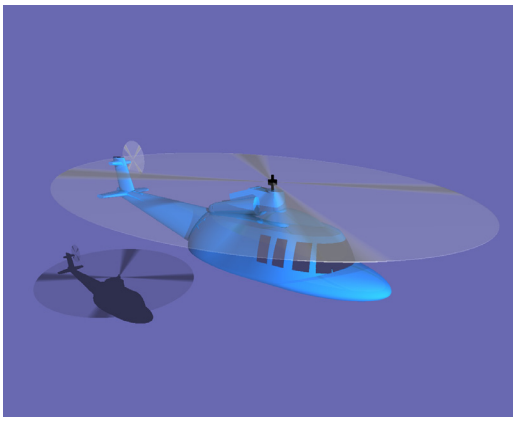

Fig. 1. POV-Ray helicopter 3D model.

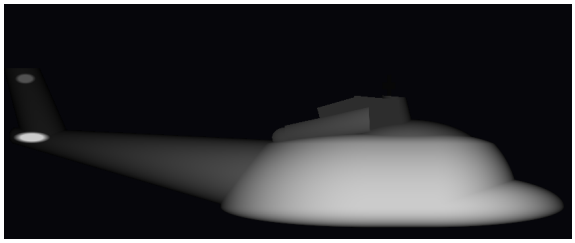

Fig. 2. Depth map image of a helicopter.

3. TOOL PATH GENERATION

A 2D profile can be stored as a binary image, where white pixels are portions of material to be kept, and black pixels are the portions of material to be removed. Using this model, it is possible to generate gouge-free toolpaths for 2D profile milling by morphological dilation with a round-shaped structural element.

After dilation, the contour extraction algorithm (Moore-Neighbour) is used and the toolpath is generated. By eroding the dilated image with the same structural element, one gets the actual machined part, where the corners that cannot be milled with the given tool are rounded (Fig. 3).

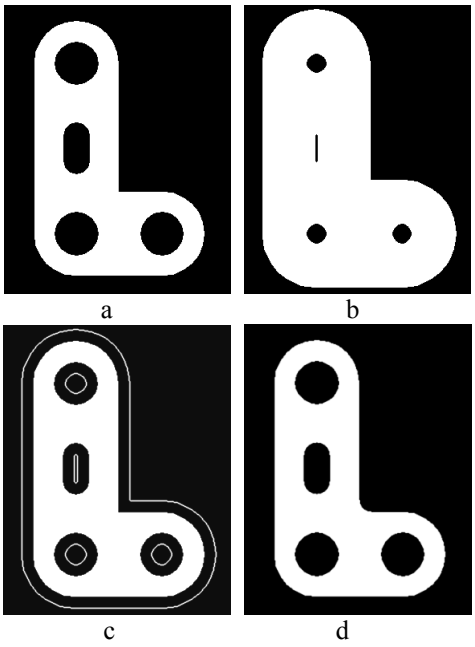

Fig. 3. Generating 2D toolpaths by morphological erosion: a) the workpiece model; b) dilated image; c) tool path; d) workpiece with rounded corners.

A speed increase for the morphological dilation is possible by processing only contour pixels of the original workpiece. The advantage of this approach is that it can be used for any free-form 2D profile shapes, and it is not necessary to compute intersections between geometrical primitives. The method can test whether a given tool radius is appropriate for machining the 2D profile, by comparing the original image with the eroded one. However, the precision of the method is limited to the image resolution, and the processing speed is much lower than geometrical-based approaches.

The method can be extended for milling 2.5D surfaces stored as greyscale height-map images. It allows one to compute cutter compensation for different cutter shapes, including but not limited to flat end mills, ball end mills, and flat end mills with corner radius, known as bull end mills.

The algorithm is based on the following idea: at every location in the XY plane (i.e. any pixel in the image) one has to compute the depth which should be

reached by the milling cutter, in order to be tangent to the surface. The shape of the milling cutter was modelled as a greyscale image, using the same scale factors as for the surface to be milled. The principle of discrete cutter compensation based on gradient computing in the 2D greyscale cutter shape image is given in Fig. 4:

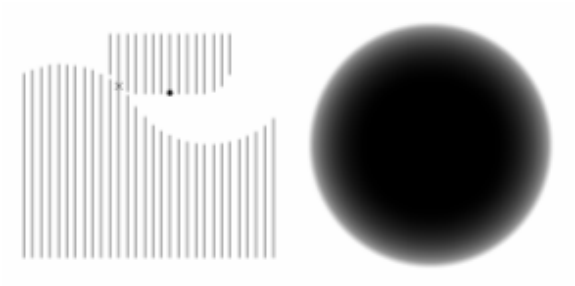

Fig. 4. Discrete cutter compensation (section). Greyscale representation of the cutter shape.

Suppose the workpiece image is stored in a matrix of $M \times N$ pixels, named A, and the cutter image is a matrix of $m \times n$ pixels, named B. We also use a $m \times n$ Boolean mask array which is true at the positions where the cutter image model contains valid values (i.e. the inside of the circle). The surface on which the tip of the milling cutter can move tangent to the workpiece is computed in the image matrix C, which is also $M \times N$ pixels. The algorithm is:

```
for ia = 1 to M
  for ja = 1 to N
   h = +inf
   for ib = 1 to m
    for jb = 1 to n
     if mask[ib,jb] = true
      hc = A[ia+ib-int(m/2),
           ja+jb-int(n/2)] - B[ib,jb];
      h = min(h, hc)
     end
    end
   end
   C[ia,ja] = h;
  end
end
```

Of course, in a real implementation, one must take care of image boundaries, but for simplicity, these tests are omitted here.

The algorithm has the complexity of $O(M \cdot N \cdot m \cdot n)$, or roughly $O(N^4)$. This significantly increases the computational time; however for some types of isoparametric toolpaths there is no need to compute the compensation for the whole surface. The computation process is equivalent to a greyscale morphological dilation, by using a non-flat structural element which is the negative of the greyscale cutter shape image.

Once the offset surface has been computed, isoparametric toolpaths can be generated easily by basic image manipulation operations. A toolpath having the X parameter constant can be obtained by extracting a column from the image. In the same way, a toolpath with the Y parameter constant is obtained by extracting a line from the image.

Toolpaths having the Z parameter constant (iso-level curves) can be computed by thresholding the image at a value corresponding to the desired Z level, followed by applying the contour detection algorithm on the binary image. Toolpaths that follow a constant direction in XY plane can be obtained by first computing the points of the 2D line along that direction using the Bresenham algorithm, and reading the grey values (heights) from these points (Bresenham, 1965).

The toolpaths generated in this way can be used for finishing the workpiece, only after the roughing cycle has been executed. If we want to generate roughing toolpaths, we must consider the following points:

- the roughing cutter should not touch the model's surface; instead, it should keep a small constant distance from the model, let's say 1 mm. This is because roughing cutters are less precise than finishing cutters, and we want to make sure roughing cuts won't be visible on the finished product.
- the roughing cutter must not plunge into the material more than a maximum allowed depth. For example, if we want to mill a pocket having a 15 mm depth and the maximum allowed depth is 5 mm, we should first plunge the cutters at 5 millimetres, then at 10 mm, and at last, at 15 mm.

For roughing cuts we will use only toolpaths at constant Z level, and a flat end mill. This allows one to use the 2D toolpath compensation algorithm, which is much faster. First we get a binary image by thresholding the original greyscale surface at the desired Z level. To achieve a small offset between the cutter and the ideal model, we compute the offsets using a larger cutter radius, i.e. we add the offset value to the actual cutter radius. That is, if we use a cutter having a radius of 10 mm, we will compute the cutter compensation with a radius of 11 mm. Also, we will subtract the offset from the plunging depth, i.e. if we thresholded the image at $Z = -5$ mm, we will plunge the cutter at $Z = -4$ mm. Actually, this will lead to a distance in the interval [*offset* ... *offset* · 1.41], depending on the local surface derivative, but this is not a problem.

For precise offsetting, the tool profile should be modified using an algorithm similar to 2D cutter compensation, and we should use the 3D compensation algorithm which is slower. The strategy for generating roughing cuts, in pseudocode, is given below:

```
for Z = -dz to -MZ step -dz
   B = threshold(model, Z)
   se = round_structural_element
                (tool_radius + offset)
   Boff = dilate(B, se)
   make_roughing_cut(Boff, Z + offset)
end
```

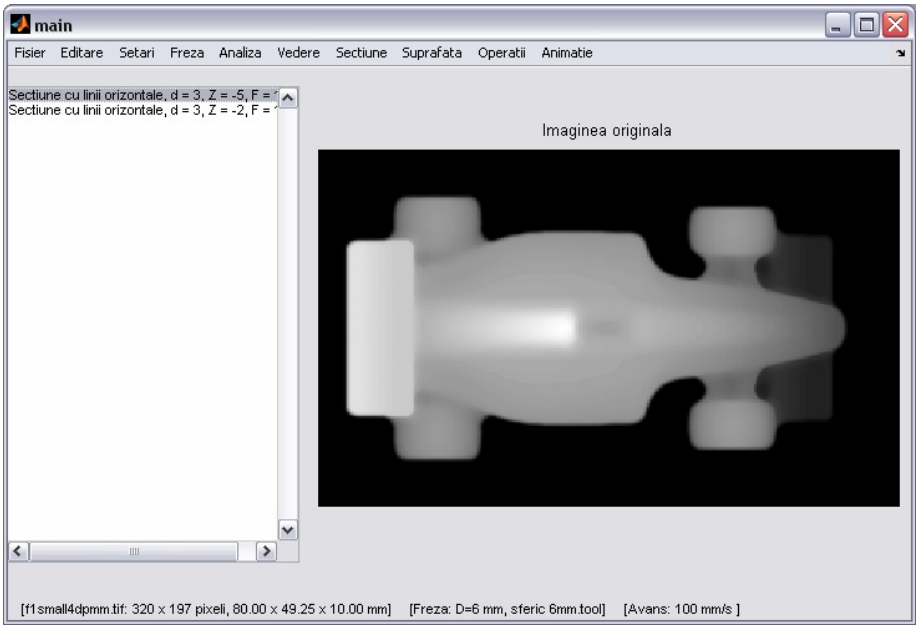

Fig. 5. Graphical user interface main window (sample: F1 car).

The computation of a roughing toolpath at a given *Z* level is done using the thresholded and dilated image, which is black where the milling tool's tip should move, and white where it shouldn't. This image is intersected with a set of straight lines parallel with one of the axis (either *OX* or *OY*). From these lines, only the segments where the image is black are retained, and after this process there will be a set of segments on which the milling tool should move. To create the actual toolpath, it is necessary to link these segments to form a continuous path. The segments should be reordered such as the extra movement of the machine (i.e. when moving from segment to segment and the milling tool is not cutting) is minimized. Because this is practically the travel salesman's problem, which is NP-complete, it may be better to use a simple greedy approach that finds a suboptimal solution, for example, starting with an arbitrary segment and, at each step, adding to the path the closest segment.

4. GRAPHICAL USER INTERFACE DESCRIPTION. EXPERIMENTAL RESULTS

A graphical user interface for tool paths generation using depth map images has been developed. The program allows defining the milling tool shape and performs cutter compensation. It is possible to view the part in different stages of the machining process. A simulation of the milling operations can also be displayed.

After loading a depth map image, the dimensions of the final product can be set by specifying the pixel to mm ratio and the depth of cut (Fig. 5). The user can define the milling tool shape (Fig. 6). Based on that, the program computes the surface on which the tool tip is moving, such as the tool is always tangent to the model.

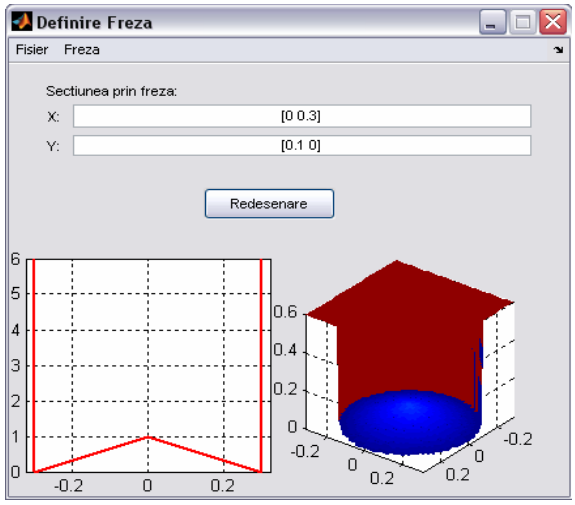

Fig. 6. Defining a milling tool shape.

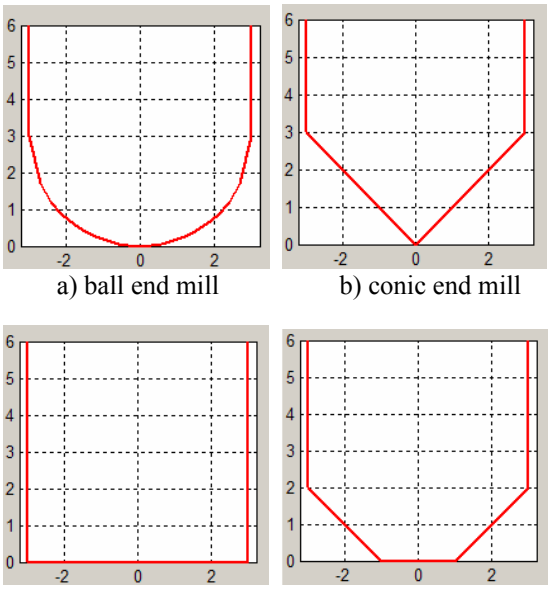

Fig. 7. Available milling tool types.

The set of milling tools includes three predefined types (ball end mill, flat end mill or conic end mill) or any shape defined by the user, by means of two arrays *X* and *Y* that represents the milling tool section. The user defined milling tool can be saved in a `.tool` extension type file.

The surface on which the controlled end point is moving so that the milling tool is tangent to the model is computed using the morphological operation for grey level images. Having a milling tool defined for the machining of the F1 car, the user can see [observe = obey] the error, i.e. the material not removed due to the dimension of current tool being too high (Fig. 8).

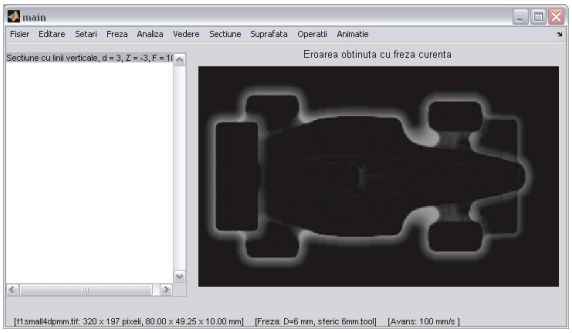

Fig. 8. Machining error due to current tool dimension

By analyzing error values, a new suitable tool may be loaded and the new instruction code is generated based on the error image.

Fig. 9 and 10 show the part model and the surface on which the tip of a 6 mm ball end mill will move.

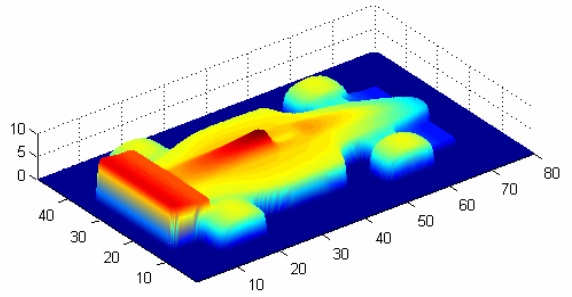

Fig. 9. Part model.

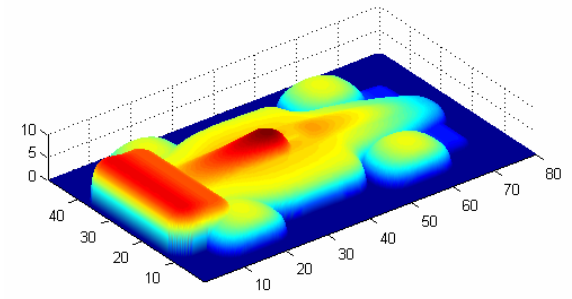

Fig. 10. Surface on which the tool tip is moving.

The part is executed in two stages: roughing and finishing. The rough machining is realised by using parallel trajectories with the *OX* or *OY* axis as shown in the graphical window in Fig. 11 and 12.

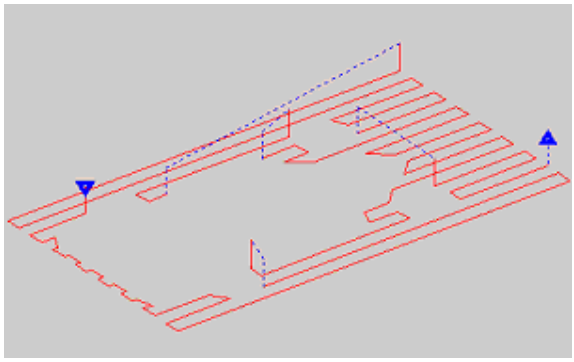

Fig. 11. Roughing stage. Horizontal lines section.

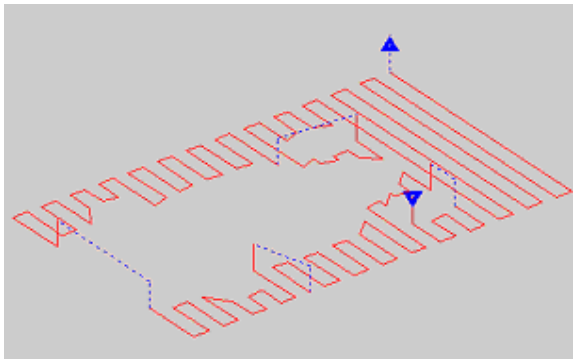

Fig. 12. Roughing process. Vertical lines section.

The finishing can be done by three methods: paths in *XZ* plane, paths in *YZ* plane and iso-level paths (Fig. 13, 14 and 15). The distance between the isoparametric toolpaths can be user defined, based on the type of the surface to be machined. Analyzing the part view after the execution of defined operations, the user can redefine the finishing method or the distance between isoparametric toolpaths.

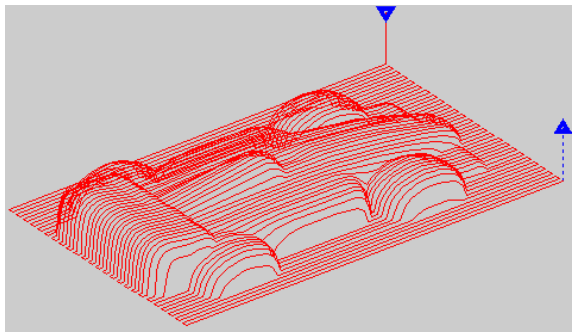

Fig. 13. The surface with horizontal paths.

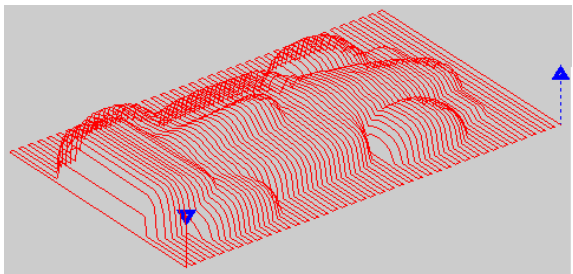

Fig. 14. The surface with vertical paths.

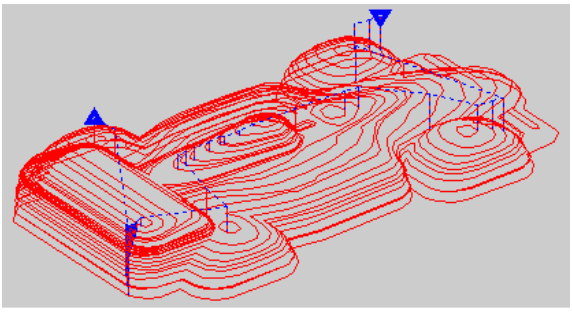

Fig. 15. The surface with iso-level paths.

The toolpaths are straight lines (G1) or circular arcs (G2/G3). The effect of the finishing type is simulated. For example, if one selects a finishing mode using paths in *XZ* and *YZ* plane, the part's granulation can be observed (Fig. 16 and 17).

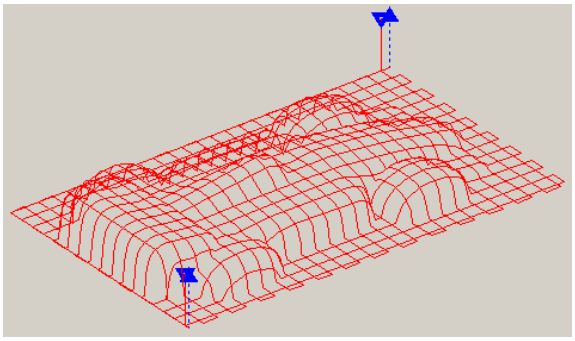

Fig. 16. Finishing toolpaths.

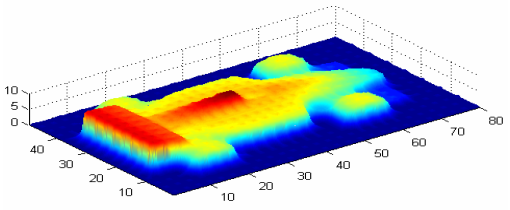

Fig. 17. Part view after the defined operations.

After defining the operations that will be executed, the 'G-code' is generated and the program can be downloaded to the machine. For the F1 car surface complete machining presented in this section, the generated machining program has 1744 instruction lines. The part is 50 x 80 x 10 millimetres, the roughing cuts were made at $Z = -5$ mm and at $Z = -10$ mm with a 5 mm flat end mill, and finishing was done with a 6 mm ball end mill using 30 non equally spaced levels for iso-level toolpaths.

5. CONCLUSIONS AND FUTURE WORK

This work has been done in the Robotics and AI Laboratory of the Faculty of Automatic Control and Computers. A vertical 3-axis EMCO F1 CNC milling machine was used for experiments. Currently, this machine is integrated in a networked production cell including machine vision stations.

The machine will be tended by a vertical automated robot equipped with matrix camera.

The communication between the robot controller and the controller of the machine is realized by input/output signals and the command programs are downloaded through the serial interface of the machine.

Fig. 18. Roughed machined Plexiglas F1 car.

The pixel model for surfaces, which was used for toolpath generation, allows computing of cutter compensation for arbitrary end mill shapes, but the algorithms are too slow for industrial usage. There are methods for speeding up the algorithms, such as parallel implementation on a multiprocessor computer, but the speed gain still does not allow achieving a high precision. However, for free-form artwork parts, the pixel model is reasonably fast and gives good results.

Future works include toolpath optimization for minimizing machining time and using better algorithms for polyline simplification, with curve fitting techniques.

REFERENCES

Bresenham, J. (1965). Algorithm for computer control of a digital plotter. *IBM Systems Journal*, Vol. **4**, no. 1, 25-30.

Dragomatz, D. and Mann, S. (1997). A Classified Bibliography of Literature on NC Tool Path Generation. *Computer-Aided Design*, Vol. **29**, no**.** *3*.

Elber, G. (1995). Freeform Surface Region Optimization for Three- and Five Axis Milling. *Computer-Aided Design*, Vol. **27**, no. *6*, 465-470.

Lefebvre, P. and Lauwers, B. (2005). 3D Morphing for Generating Intermediate Roughing Levels in Multi-Axis Machining. *Computer-Aided Design & Applications*, Vol. **2**, no. *1-4*, 115-123.

Park, S. (2005). Associating a Design Model and its Machining Process Plan. *Computer-Aided Design & Applications*, Vol. **2**, no**.** *5*, 665-673.