4

Advances in CAD/CAM Technologies

Mircea Viorel Drăgoi
Transilvania University of Braşov
Romania

1. Introduction

The product development, as a stage of product life cycle, has to meet the requirements enforced by the technological component on one hand, and by the economical one, on the other hand.
The technological component enforces restrictions referring to materials, cutting tools, machine-tools foreseen for processing, and mainly to the possibility to work a certain part by the means available.
The economical component comes with restrictions referring to the costs of developing the product. It is clear that the two types of restrictions work contrary: in the most cases, the more a material, a tool, or a technology is effective, the more expensive it is. On the other hand, the cheap resources cannot always meet all the technical / technological specific requirements of a certain product. Consequently, a permanent optimisation is necessary. Such an optimisation means finding the best compromise between the technological and the economical criteria. This compromise must lead to producing a product that fully meets all the customer's technical specifications, and which is attractive in terms of price, as well.
Optimisation has to be applied in all the stages of product development, consequently in the simultaneous design – constructive and technological – of the product, and its integrated conception with the production.
It is known that the CAD/CAM software is very expensive. Many SME's that do not use sophisticated hardware resources, and do not use to produce very complicated parts are interested to benefit from low-priced pieces of software that can give answers to the most common problems of designing and manufacturing. Some specific tools that ease the designer's job in terms of finding the best solution to technological problems, or automating some specific laborious tasks are presented in this chapter. Note that the solutions here provided do not intend to replace the well-known specific software, but rather to be an alternative to it, there where those are not a real necessity.
Some samples of problems approached here are automatically checking if a certain part can be machined by up to three axes CNC milling machines, optimizing the approximation of the curves in order to engender automatically CNC files, automatically engendering the tool-path for CNC milling in vertical planes (a new method to compute ball nose milling cutter radius compensation in Z axis for CNC), transferring geometrical data from CAD systems to CAM system (in order to engender CNC files) as a mean of integration of CAD and CAM systems.
AutoCAD and AutoLISP were chosen as the support of CAD/CAM systems.

2. 3D modeling of the product

3D modeling is the necessary stage the geometry of the part is designed in. In the modern product engineering, describing in terms of geometry of a part cannot be completely separated from taking into account the technological criteria. Further more, the quality of the outcome of the conception process is strongly influenced by the level of integration of the constructive and technological aspects of the stage of designing. In these terms, at least two aspects should be taken into account:

- using the constructive-technological entities, which integrate geometrical features (dimensions, shape) and technological features (limits or deviations, surface quality), as well;
- the machinability of the shape.

2.1 Using the constructive-technological entities in designing

The constructive-technological entity (Chicoş & Ivan, 2004; Chicoş et al., 2009) has the advantage that the use of data with technological specific feature creates the basic prerequisite for automatically/computer aided engendering the process plan of the surface under debate, as a necessary stage in elaborating the technology of the part.

Binding technological specific features to a geometric entity (i. e. a line, that in shop-floor drawing may represent a surface of the part) can be done in AutoCAD by means of specific user-defined functions.

Such user-defined functions present dialog boxes by means of which the features (either geometrical or technological) are input. The geometrical data are used to define the entity, calling from the function the appropriate AutoCAD command. The technological data are bound to the new entity by means of its associated list, in the area of extended data.

A sample of user-defined function, named EXDATROUG1 (EXtended DATa ROUGhness), that binds to an existing entity the roughness, as extended data, is presented bellow. The entity the extended data is bound to must be selected by user.

Program line	Effect
(defun C: EXDATROUG1 ()	
(setq addrug (entget (car (entsel))))	Gathers the associated list of the entity (the user selects) to which extended data must be attached
(setq RR (getreal "Input roughness:"))	Gathers from the user the value of the roughness, to be attached as extended data
(regapp "Rug")	Creates an application named *Rug* in order to be attach the extended data
(setq data_extinsa_rug	
(list (list -3 (list "Rug" (cons 1040 rr)))))	Builds the list that defines the extended data
(setq addrug (append addrug data_extinsa_rug))	Appends to the associated list of the initial entity the extended data
(entmod addrug))	Attaches the extended data to the associated list of the entity

Table 1. EXDATROUG1 user-defined function. Explanations

In a similar manner new entity having extended data attached can be created by means of a user-defined function. Building the associated list that defines the attached extended data is done as in the sample above (Table 1). All the dotted pairs that define the geometric data and the AutoCAD properties (e. g. layer, thickness, color, linetype, a.s.o.), and which constitute the associated list of the new entity must be composed as well. Assembling the new entity and drawing it is performed using the standard AutoLISP function *ENTMAKE*. Such a function, EXDATROUG2 (EXtended DATa ROUGhness), is described bellow:

Program line	Effect
(defun C: EXDATROUG2 ()	
(setq RR (getreal " Input roughness:"))	Gathers from the user the value of the roughness, to be attached as extended data
(regapp "Rug")	Creates an application named *Rug* in order to be attach the extended data
(setq data_extinsa_rug	
(list (list -3 (list "Rug" (cons 1040 rr)))))	Builds the list that defines the extended data
(setq P1 (getpoint "Input start point:")	Gathers interactively from the user the first point of the line
P2 (getpoint "Input end point:"))	Gathers the second point of the line
(setq l1 (list (cons 0 "LINE"))	Defines the type of the new created entity (*LINE*)
l2 (list (cons 8 "0"))	Selects the layer where the entity to be placed
l3 (list (cons 10 p1))	Defines the start point of the new created line
l4 (list (cons 11 p2)))	Defines the endpoint of the new created line and ends creating the associated list
(setq addrug (append l1 l2 l3 l4 data_extinsa_rug))	Attaches the extended data to the associated list of the new entity
(entmake addrug))	Creates the new entity (line) with extended data

Table 2. EXDATROUG2 user-defined function. Explanations

Remarks:

- if try to attach new extended data to an entity that already has some one, the old extended data is overwritten. **Sample:** if the entity described by the associated list
  *((0 . "LINE") (8 . "0") (10 5.0 5.0 0.0) (11 5.0 15.0 0.0) (-3 ("Rug" (1040 . **12.5**))))*
  is processed by **EXDATROUG1** user-defined function, binding the value of 6.3 to the roughness, the associated list of the entity becomes
  *((0 . "LINE") (8 . "0") (10 5.0 5.0 0.0) (11 5.0 15.0 0.0) (-3 ("Rug" (1040 . **6.3**))))*,
  and not
  ((0 . "LINE") (8 . "0") (10 5.0 5.0 0.0) (11 5.0 15.0 0.0) (-3 ("Rug" (1040 . **6.3**))) (-3 ("Rug" (1040 . 3.**2**)))),
  as somebody might think;

- independent of the way the extended data was attached, it can be later gathered by using the standard AutoLISP functions *ENTGET* and *ASSOC*. *ENTGET* must however be called with an additional parameter as against the usual way to use it: this parameter must point to the application that defines the extended data to be gathered. So, the call (ENTGET (entlast)) – the common use – will return the associated list

((0 . "LINE") (8 . "0") (10 5.0 5.0 0.0) (11 5.0 10.0 0.0)),

while the call (ENTGET (entlast) „Rug") will return

((0 . "LINE") (8 . "0") (10 5.0 5.0 0.0) (11 5.0 15.0 0.0) (-3 ("Rug" (1040 . **6.3**))))

One might want to use the extended data associated to an entity from within a piece of software that acts as a technological processor (Ivan 1994; Ivan 1996). This is possible by accessing the associated list of the targeted entity and cutting off from the list the wanted data. To automate this task, another user-defined function was created. It is named *GETROUG* and is presented in the following table:

Program line	Effect
(defun C:getroug ()	
(setq l (entget (car (entsel)) (list "Rug"))	Get the associated list of the selected entity
l (cdr (assoc -3 l))	The extended data in the associated list: (("Rug" (1040 . 3.2)))
l (car l)	The first sub-list in the extended data ("Rug" (1040 . 3.2))
l (cdr l)	The right member of the previous list: ((1040 . 3.2))
l (car l)	The first sub-list from the previous: (1040 . 3.2)
Rug (cdr l))	**The roughness**
)	Ending the function

Table 3. GETROUG user-defined function. Explanations

This function may be used either within a technological processor, or in any other piece of software that has to use the value of the roughness as an extended data.

2.2 Machinability of the surfaces

In this context, ***Machinability*** is understood as the potential of a surface to be obtained by certain (simple) means, at the required precision of its shape and dimensions.

With no doubt, any surface of a part has to meet all the requirements in terms of functional and aspect. However, the technological aspect may not be neglected. Some shape details do not affect the good working of the part, but might have major influence on the machinability of a part. A non-technological shape may lead to unjustified expenses related to tool consumption, to necessity of using sophisticated and expensive resources (technologies, machine-tools, fixtures). An appropriate sample in this acceptance is the presence or not of the filleting radius between two adjacent surfaces, as shown in Fig. 1.

A special attention must be given to the grooves at the junction of the perpendicular surfaces that must be grinded, or at the end of internal threads.

Furthermore, a special importance has the shape of the cast parts or of those made by plastic mass injection, in the context of making the cast models and moulds by CNC technologies.

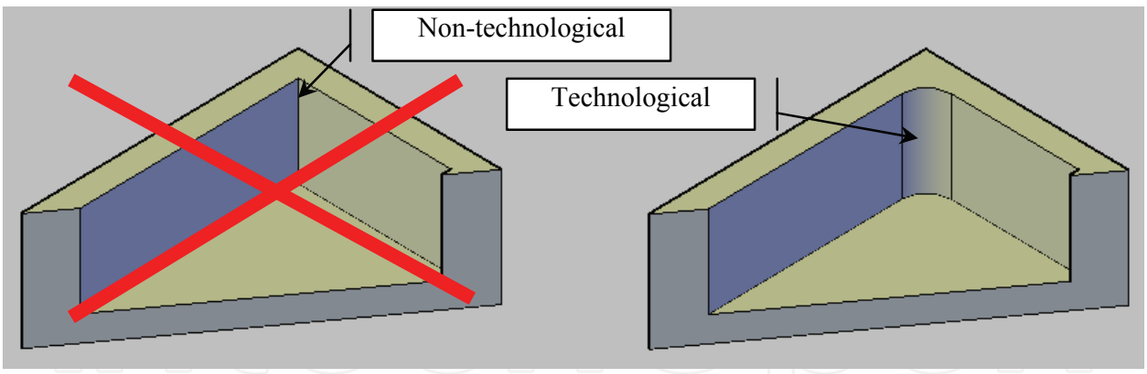


Fig. 1. Filleting radiuses influence the part machinability

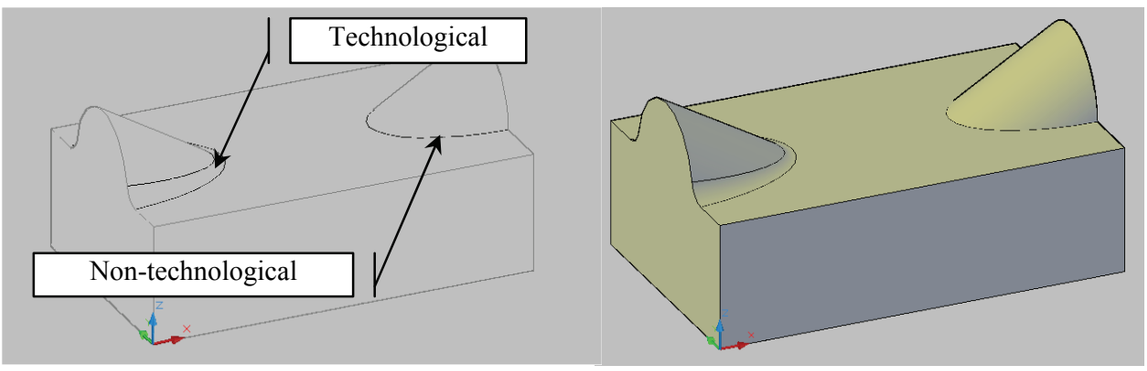


Fig. 2. Filleting radiuses influence the part machinability

2.3 Checking the machinability of the parts in order to be processed by milling on CNC machine-tools

The most CAD/CAM systems suffer from a big deficiency: their machining (CAM) modules engender correctly the tool-paths in terms of collisions, but without reporting if in certain cases the part cannot be completely be worked, that is the shape cannot be completely achieved. For milling processes, this happens if a CNC equipment with no more than three axes is used, and the orientation of a surface is not proper relative to the vertical plane, as can be seen in Fig. 3.

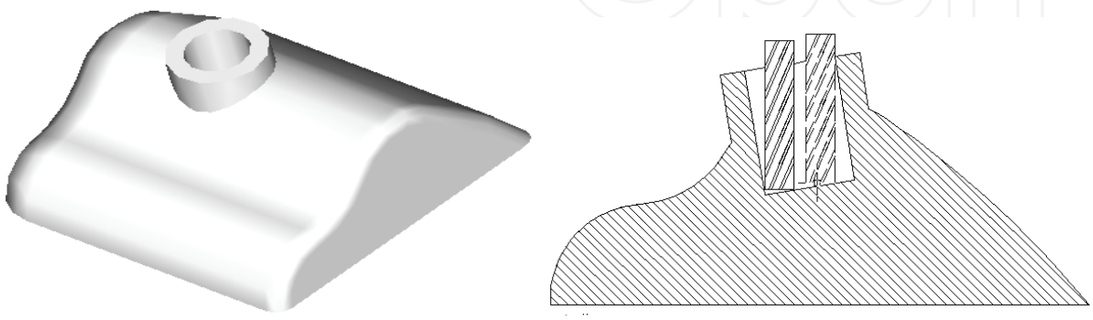


Fig. 3. Part which cannot be processed (the hole because it's inclination) on a 2 ½ or 3 axes CNC machine tool

To be able to be processed (machinable) on a 2 ½ or 3 axes CNC machine tool, the part must meet a specific requirement: none of its surfaces may be bevelled "under the vertical".
The problem that must be solved in such cases is to detect the surface elements (if they exist) that cannot be machined on 2 ½ or 3 axes CNC machine-tools, because of their inclination reported to vertical. Furthermore, if such surfaces exist, must be sought out the possibility to yet process the part by modifying its orientation. If a solution is available, it should be described by the plane and the angle in/with which the part must be rotated to come in a proper position as to be worked.
To identify the surface elements that cannot be worked (in the context of those stated above) may be followed two ways: the study can be performed sectioning the part either by vertical or horizontal planes, each of it having specific advantages and disadvantages.
The solving here proposed assumes the study of the 3D model of the part, either built in AutoCAD, or imported in AutoCAD from other CAD systems, by means of IGES or DXF files.

2.3.1 Scanning in vertical planes

To find the surface elements that cannot be machined for the reason stated above, the 3D model of the part is successively sectioned by equidistant vertical planes, the outcome being a family of regions. Each of the regions is exploded becoming a polyline, whose geometrical features can be investigated by means of data stored in the associated list (Fig. 4). Every pair of points P_i, P_{i+1} is studied, and the occurrence of the situation $X_{i+1}<X_i$ means the existence of a surface element that cannot be machined. We call such a situation *undercut*. In this case the angle the segment P_iP_{i+1} makes with the vertical is computed, taking into account the sign of this angle. The convention of the signs is indicated in Fig. 4.

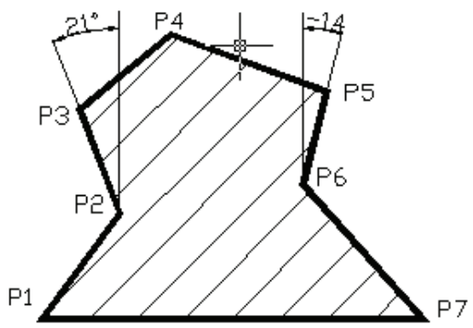


Fig. 4. Contour obtained by sectioning the part with a vertical plane

As more such situations arise, either in the same section, or in different sections, the maximum and the minimum value of the angle are stored. At the end of the scanning process the presence and the type of undercuts are evaluated. Depending on the result of this evaluation, some conclusions are made, as follows:

a. if no undercuts, the part can be machined on a 2½ or 3 axes CNC machine-tool. (final conclusion);
b. if undercuts exist, and they have the same sign, the part must be rotated in vertical plane with a minimum angle equal to the maximum undercut (Fig. 5); Scanning is performed again, until a final conclusion can be made;
c. if at least two undercuts having opposite signs exist, the part cannot be machined on a 2½ or 3 axes CNC machine-tool, even if the part should be rotated (final conclusion).

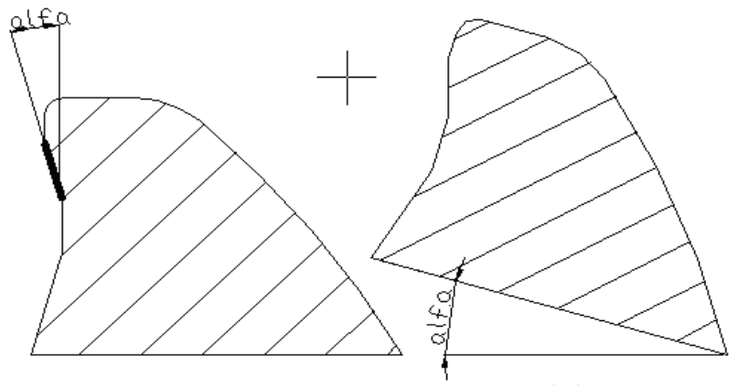


Fig. 5. Rotating the part in order to avoid undercuts

2.3.1 Scanning in horizontal planes

This method is simpler and more reliable than scanning in vertical planes, but doesn't offer solutions in terms of finding an appropriate orientation of the part, as it to be machinable, even when such solutions exist. To describe the algorithm two sample parts are used, those in Fig. 6.

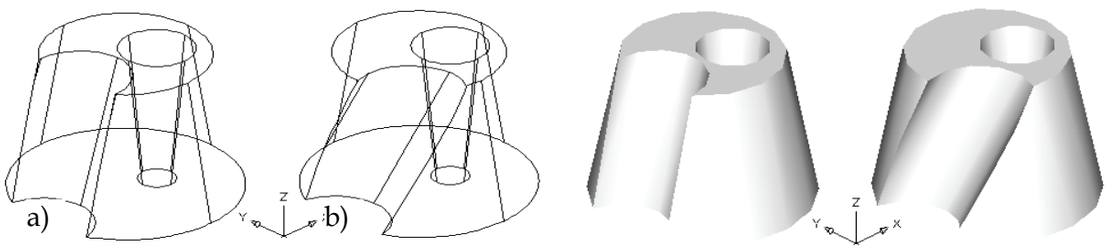


Fig. 6. Parts with similar configuration

The parts are conic trunks from which a cylinder and another conic trunk were extracted. The cylinder was previously rotated 10° about Y axis (Fig. 6a) and 15°and 10° about X and respectively Y axis (Fig. 6b). The rotation of the cylinder 15° about X axis makes machining impossible on a 2½ or 3 axes CNC machine-tool (if reorient the part by rotating it -15° about X axis an undercut occurs on the internal conical surface.

The working principle of the method consists in sectioning successively the 3D model of the part with equidistant horizontal planes, beginning from the bottom (Fig. 7).

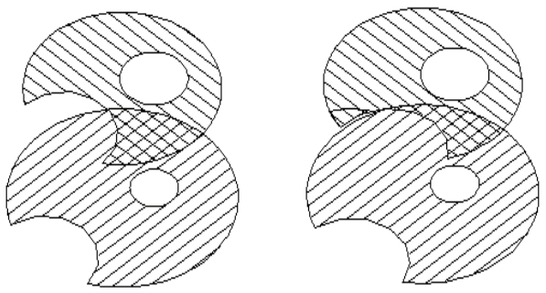


Fig. 7. Horizontal sections through the two sample parts

The outcome of the sectioning is a region. Pairs of regions are studied in the following way: two consecutive regions are copied in the same horizontal plane (the upper region is moved downward to the plane of the lower one), without being moved in horizontal direction. The two objects in the same plane are intersected. If the output is congruent with the region that initially was upper, on the lateral surfaces between the planes of the regions analyzed do not occur undercuts. The mentioned congruence is equivalent with that in a projection on a horizontal plane the upper section is entirely included in the lower one. Such a congruence is emphasized in Fig. 8a), where the smaller region is entirely covered by the larger one. In Fig. 8b) the regions do not overlap integrally, a sector of the smaller region (filled with colour) is not covered by the larger region.

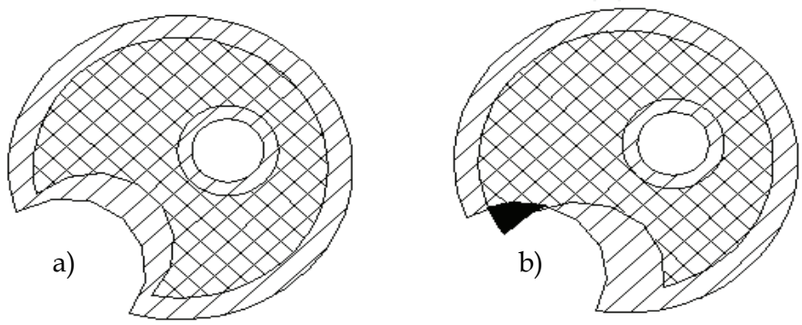


Fig. 8. Intersection (overlaping) of the regions

3. Tools for computer aided engendering of CNC files

3.1 Finding the size of solids in AutoCAD

It is usual to section a 3D model with vertical (YZ or ZX) planes in order to reduce the 3D problem of determining the CNC tool-paths to a set of 2D much easier to be solved problems. "YZ" or "ZX" are not singular planes, but families of parallel planes. To indicate a particular plane of a family, a point the plan goes through must be specified. To get a reasonable effect from sectioning a solid with a plane, of course, the plan must intersect the solid. That is, the sectioning plane must go by a point inside the bounding box of the solid. At a glance, it seems very easy to indicate a point inside a box – that if work interactively. If the job is to be performed from within an AutoLISP program, is not so simple, all the more it has to be done repeatedly exactly from an end of the box to another. So, knowing the data that describe the bounding box of a 3D model is a necessity to handle the solid correctly. The single mean to get the bounding box of a solid is the command MASSPROP. This command returns the wanted data either displayed on the screen, or saved in a file, but doesn't store it in system variables as to be ready to hand for the programmer. A piece of software that gives the user the coordinates of the ends of the bounding box would be very useful. A user-defined function was built, it is named 3DSIZE, and its' return are the minimum and maximum values of X, Y, and Z coordinates of a 3D model in AutoCAD. This function is useful, as well, when designing the billet a part will be cut off from.

The function is based on the fact that MASSPROP command returns all the mass properties of a solid (including its bounding box) in a report having the following format:

---------- SOLIDS ----------
Mass: 1003.5348 Volume: 1003.5348 Bounding box: X: -2.1511 -- 113.0500 Y: -166.0205 -- -156.5898 Z: -2.3466 -- -0.1022 Centroid: X: 62.7399 Y: -161.9498 Z: -1.2244 Moments of inertia: X: 26324880.1352 Y: 5074009.6821 Z: 31395132.5908 Products of inertia: XY: -10197522.0729 YZ: 198988.3441 ZX: -77088.8178 Radii of gyration: X: 161.9634 Y: 71.1065 Z: 176.8744 Principal moments and X-Y-Z directions about centroid: I: 2942.6215 along [1.0000 -0.0008 0.0000] J: 1122292.9852 along [0.0008 1.0000 0.0000] K: 1124487.1726 along [0.0000 0.0000 1.0000]

Table 4. The MASSPROP report

If save this report, all the wanted data can be read from that file, following a specific rule. After read the data from file, the text values must be converted in numbers and stored in variables. The first section of the function is presented in Table 5. Only retrieving the values of z_{max} and z_{min} is described, the same algorithm being followed for X and Y co-ordinates.

Remark: the mass properties are computed appertained to the current coordinate system. So, if before calling the function the origin of the coordinate system is placed in the point that is going to become the part datum, the values returned by the function can be used for CNC programming.

Program line	Effect
(defun C:3Dsize ()	
(setq prisma (car (entsel)))	Select the object to be evaluated
(command "massprop" Prisma "" "Y" "c:\\temp.mpr")	Call MASSPROP command and write data to file
(setq fis (open "c:\\temp.mpr" "r"))	Open the data file
(repeat 5 (setq linie (read-line fis)))	Read lines that contain irrelevant data
(setq linx (read-line fis)) ;X	Read the line with data about solid extent along X axis
(setq liny (read-line fis)) ;Y	Read the line with data about solid extent along Y axis
(setq linz (read-line fis)) ;Z	Read the line with data about

	solid extent along Z axis
(setq fis (close fis)	Close file
c %	Binding values to some variables
cifre (list "1" "2" "3" "4" "5" "6" "7" "8" "9" "0" "." "-")	
s linz	
)	
(while (not (member (substr s 1 1) cifre))	Processing the line with data about the height of the solid. Get the data referring the extent of the solid along the vertical, and computing the height of the object.
(setq s (substr s 2))	
)	
(setq zmins "")	
(while (member (substr s 1 1) cifre)	
(setq zmins (strcat zmins (substr s 1 1))	
s (substr s 2)	
)	
)	
(setq s (substr s 5))	
(while (not (member (substr s 1 1) cifre))	
(setq s (substr s 2))	
)	
(setq zmaxs "")	
(while (member (substr s 1 1) cifre)	
(setq zmaxs (strcat zmaxs (substr s 1 1))	
s (substr s 2)	
)	
)	
(setq zmin (atof zmins)	
zmax (atof zmaxs)	
lz (- zmax zmin)	
.....	Program sections related to the extents of the solid along X and Y axis. They are similar to Z
.....	
)	End the function

Table 5. 3DSIZE user-defined function. Explanations

3.2 Gathering polyline data from DXF files in order to engender CNC files

The DXF files are often used because can be handled by any CAD/CAM system. The function described bellow gathers the data describing polylines from DXF files and use it for automatically engendering the CNC files. Since the polylines in discussion are 2D objects, the application can be used for outlining processes (milling, cutting by laser or water jet). In the DXF files data are always presented in groups of two lines. The value on the first line of the group is always a numeric one, and represents a code that indicates the meaning of the value on the second line. The value on the second line may be a numeric or a text one. The data bundles that describe polylines in DXF files are grouped in four or six lines, as they represent an end of a line or of an arc. The first group of four lines describe the start point of the polyline. The next groups describe the points the polyline passes along. The code 42,

named also *Bulge*, indicates that the next point is reached along an arc. The value of *Bulge* allows computing data that describe the arc. The sign of *Bulge* gives the sense of travel through the arc (positive – counter clockwise, negative – clockwise). The value that follows the code *10* represents the X coordinate of the point, and that follows the code *20* is the Y coordinate of the point. The value following the code *42*, named *Bulge*, describes the curvature of the arc. In fact, *Bulge* represents the tangent of the quarter of the arc's included angle. An arc on the polyline is perfectly defined by three data: startpoint, endpoint, and *Bulge*. Anyway, this minimum set of data is not enough to draw an arc, or to program it in a CNC file. Therefore, *Bulge* must be converted in an angle, as then compute the radius of the arc.
Equations (1) .. (5) are used to compute the data that describe the arc as it to be drawn or programmed in CNC files. The meanings of the notation are according to Fig. 9, as follows: U=the included angle of the arc; d=the length of the arc chord P_1P_2; R= the arc radius.

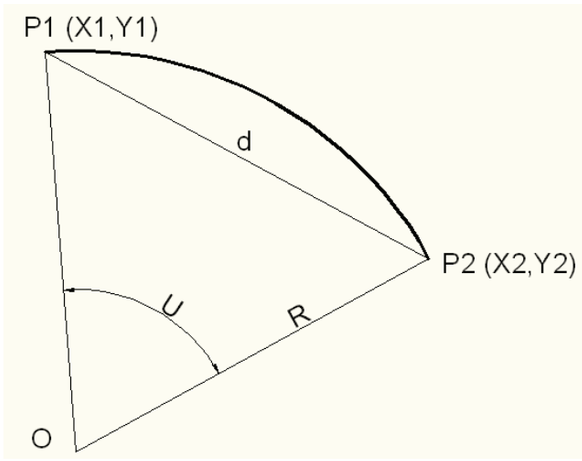


Fig. 9. Arc elements

The included angle of the arc is computed wit equation (1), since bulge is available. The length of the chord, d, is computed with equation (2). Applying in triangle OP_1P_2 the generalized theorem of Pythagoras - equation (3), and furthermore transforming, the value of the radius is found.

$$U=4*\arctan(\text{Bulge}) \tag{1}$$

$$d=\sqrt{(x1-x2)^2+(y1-y2)^2} \tag{2}$$

$$R^2+R^2-2*R*R*\cos(U)=d^2 \tag{3}$$

$$2*R^2*(1-\cos(U))=d^2 \tag{4}$$

$$R=\frac{d}{\sqrt{2*(1-\cos(U))}} \tag{5}$$

The DXF_CNC function opens the DXF file, finds in it data that describe the polyline, and gathers the required pieces of information. For the arc, it computes the included angle and radius. As calculi are made, the segments and the arcs are drawn on the display and the G codes are written to the CNC file. The function is described bellow.

Program line	Effect
(defun C: DXF_CNC ()	
(command "UNDO" "BE")	
.....	Prepare input data (names of files)
(write-line "TEST" CNC)	Writes the first lines to CNC file
(write-line "G28 Z30" CNC)	
(write-line "M06 T01" CNC)	
(write-line (strcat "M03 S" (itoa (fix s))) CNC)	
(setq linie "" inceput T)	
(while (not (= linie "LWPOLYLINE"))	Ignores in the DXF files the irrelevant data
(setq linie (read-line fis))	
)	
(while (not (= linie "ENDSEC"))	
(setq linie (read-line fis))	
(if (= linie "LWPOLYLINE") (setq inceput T bulge 0))	
(cond	
((= (atof linie) 10.0)	Gathers the coordinates of the point
(setq x (atof (read-line fis)))	
)	
((= (atof linie) 20.0)	
(setq y (atof (read-line fis)))	
(if inceput (progn	If it is the start point of the polyline, programs the tool start position and entering into the billet
(setq p1 (list x y) inceput nil)	
(write-line (strcat "G00 Z-" (rtos ad)))	
(write-line (strcat "G01 X" (rtos x) " Y" (rtos y) " F" (itoa (fix Avans))) CNC))	
(progn	If it is an arc, computes the radius - equations (1) .. (5), and programs the arc, clockwise, or counter clockwise, as the sign of *Bulge* impose
(setq p2 (list x y))	
(if (/= bulge 0)	
.....	
)	
(Progn (command "LINE" p1 p2 "")	Programs linear interpolation when need
(setq p1 p2)	
(write-line (strcat "G01 X" (rtos x) " Y" (rtos y)) CNC)	
)))))	
((= (atof linie) 42.0)	Gets the value of Bulge, computes the included angle in radians, and transforms it in degrees
(setq bulge (atof (read-line fis))	
u (* 4 (atan bulge))	
ugrad (/ (* u 180) pi)))))	
(setq fis (close fis))	Close DXF file
(write-line "G28 Z30" CNC)	Writes the ending sequence to CNC file
(write-line "M02" CNC)	
(write-line "M30" CNC)	
(setq CNC (close CNC))	Close CNC file
(setvar "OSMODE" os)	
(command "UNDO" "E")	
)	

Table 6. DXF_CNC user-defined function. Explanations

3.3 Gathering polyline data from the associated list of AutoCAD entities in order to engender CNC files

In table 6 a simplified form of the function was presented. The function is equipped with specific mechanisms that allow it to know if the polyline is open or not. If the polyline is closed, its first point is stored, as it can be used as the last point of the polyline, as well, even this is not explicitly given by the DXF file.

Through two points, two arcs having the same radius, but different included angle (a smaller than 180˚ and a larger than 180˚ - see Fig. 10) can be drawn. To distinguish them, a rule has to be followed: the arc larger than 180˚ has to be programmed in the CNC file as having a negative radius. This aspect is also solved by the function.

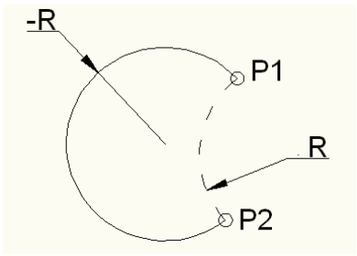


Fig. 10. Two arcs having the same radius and going through the same two points, P_1 and P_2

The function described above was adapted to gather data from the associated list of the polyline in an AutoCAD drawing. The operating and computing algorithm is the same, differs only the way data is gathered.

The list bellow is associated to a pentagon having the edge length of 50, and whose corners are filleted with the radius of 10.

((-1 . <Entity name: 7ef083b8>) (0 . "LWPOLYLINE") (330 . <Entity name: 7ef06cf8>) (5 . "1F7") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbPolyline") (90 . 10)
(70 . 1) (43 . 0.0) (38 . 0.0) (39 . 0.0) (10 27.2654 -1.77636e-015) (40 . 0.0) (41 . 0.0) (42 . 0.0)
(10 62.7346 -1.42109e-014) (40 . 0.0) (41 . 0.0) (42 . 0.32492)
(10 72.2451 6.90983) (40 . 0.0) (41 . 0.0) (42 . 0.0)
(10 83.2057 40.643) (40 . 0.0) (41 . 0.0) (42 . 0.32492)
(10 79.573 51.8233) (40 . 0.0) (41 . 0.0) (42 . 0.0)
(10 50.8779 72.6716) (40 . 0.0) (41 . 0.0) (42 . 0.32492)
(10 39.1221 72.6716) (40 . 0.0) (41 . 0.0) (42 . 0.0)
(10 10.427 51.8233) (40 . 0.0) (41 . 0.0) (42 . 0.32492)
(10 6.79429 40.643) (40 . 0.0) (41 . 0.0) (42 . 0.0)
(10 17.7549 6.90983) (40 . 0.0) (41 . 0.0) (42 . 0.32492) (210 0.0 0.0 1.0))

All the dotted pairs of the associated list are successively cut off until the first one to have the code 70 is found. Here is the piece of information that says if the polyline is closed (value 1) or open (value 0). After this data was gathered, the function looks for the first dotted pair which has the code 10. Beginning from this point the geometrical data of the polyline are obtained. They are processed exactly as they were in the **DXF_CNC** function. Optionally, three special program lines may be added at the beginning of the CNC file, to be simulated as it should be run on a 2 ½ axes milling machine from Denford, UK, and having a Fanuc controller. This version of the function is more advantageous than the DXF one because of if in the drawing there are more polylines, one/those wanted to be processed can be selected. The function LAsoc_CNC is described bellow.

Program line	Effect
(defun C:LAsoc_CNC ()	
(command "UNDO" "BE")	
(setq os (getvar "osmode"))	
(setvar "OSMODE" 0)	
.....	Prepare input data
(setq nume_fis (getfiled "Open DXF" "c:\\ DXF" "CNC" 1))	Prepare CNC file to be written
(setq CNC (open nume_fis "W"))	
(write-line "[BILLET X100 Y100 Z10" CNC)	Write the opening sequence to CNC file
(write-line "[EDGEMOVE X0 Y0 Z0" CNC)	
(write-line "[TOOLDEF T01 D30" CNC)	
(write-line "G28 Z30" CNC)	
(write-line "M06 T01" CNC)	
(write-line (strcat "M03 S" (itoa (fix s))) CNC)	
(setq l (entget (car (entsel "Select the polyline:"))))	Gets the associated list of the polyline
(while (/= (caar l) 70) (setq l (cdr l)))	Learn whether the polyline is closed or not
(setq inchis (cdar l))	
(if (= inchis 1) (setq inchis T) (setq inchis nil))	
(while (/= (caar l) 10) (setq l (cdr l)))	Cuts off the irrelevant data
(setq p1 (cdr (nth 0 l)) x1 (car p1) y1 (cadr p1) bulge (cdr (nth 3 l)) xprimul x1 yprimul y1)	Gets the first point and stores it in order to be used as the last one for the closed polylines
(write-line (strcat "G0 X" (rtos x) " Y" (rtos y) " F" (itoa (fix Avans))) CNC)	Programs the tool start position and entering into the billet
(write-line (strcat "G01 Z-" (rtos ad)) CNC)	
(repeat 4 (setq l (cdr l)))	
(while (> (length l) 1)	
(setq p2 (cdr (nth 0 l)) bulge1 (cdr (nth 3 l)) x2 (car p2) y2 (cadr p2) i 0)	Gets the next point
(setq u (* 4 (atan bulge))	Computes the included angle
ugrad (/ (* u 180) pi))	
(if (/= bulge 0)	
(progn (command "ARC" p1 "E" p2 "A" ugrad)	If it is an arc, computes the radius and programs the arc, clockwise, or counter clockwise, as the sign of *Bulge* impose
(if (> bulge 0) (setq g "G03 ") (setq g "G02 "))	
(setq x1 (car p1) y1 (cadr p1) x2 (car p2) y2 (cadr p2)	
d (sqrt (+ (* (- x1 x2) (- x1 x2)) (* (- y1 y2) (- y1 y2))))	
r (sqrt (/ (* d d) 2 (- 1 (cos u))))	
(if (> (abs ugrad) 180) (setq r (- r)))	Changes the sign of R if the included angle of the arc is larger than 180˚
(write-line (strcat g "X" (rtos x2) " Y" (rtos y2) " R" (rtos r)) CNC)	
(setq p1 p2 bulge 0))	
(Progn (command "LINE" p1 p2 "")	Programs linear interpolation when need
(write-line (strcat "G01 X" (rtos x2) " Y" (rtos y2)) CNC)))	
(repeat 4 (setq l (cdr l)))	Removes from the list the sub-lists of the current point
(setq p1 p2 bulge bulge1)	
(write-line "G28 Z30" CNC)	Writes the ending sequence to CNC file
(write-line "M02" CNC)	
(write-line "M30" CNC)	
(setq CNC (close CNC))	Close CNC file
(setvar "OSMODE" os)	
(command "UNDO" "E")	
)	

Table 7. LAsoc_CNC user-defined function. Explanations

Bellow is shown the CNC file as an output of the function, run for the pentagon described by the associated list presented above.

```
[BILLET X100 Y100 Z10
[EDGEMOVE X0 Y0 Z0
[TOOLDEF T01 D30
G28 Z30
M06 T01
M03 S3000
G00 X27.2654 Y0 F125
G01 Z-2
G01 X62.7346 Y0
G03 X72.2451 Y6.9098 R10
G01 X83.2057 Y40.643
G03 X79.573 Y51.8233 R10
G01 X50.8779 Y72.6716
G03 X39.1221 Y72.6716 R10
G01 X10.427 Y51.8233
G03 X6.7943 Y40.643 R10
G01 X17.7549 Y6.9098
G03 X27.2654 Y0 R10
G00 Z10
G28 Z30
M02
M30
```

3.4 Computer aided engendering of CNC files for milling the complex surfaces

The function named VertMill is used to engender the CNC files for finishing by milling following a strategy in vertical planes. To gather the geometrical data in order to process the part, its 3D model is sectioned by vertical planes (YZ or ZX, by the choice of the user). The regions so obtained are exploded, getting first polylines, and then, by a new explosion elementary 2D entities are separated: lines, arcs, ellipse arcs or splines. From these are kept only those are useful for the freeform shapes (thick in Fig. 11), not the sides or the bottom of the outline.

When gathering and processing the geometrical data, in order to define the tool-path, some aspects must be taken into account:

- most of the 2 ½ and 3 axes CNC equipment can perform only linear and circular interpolation;
- the tool-path is obtained most frequent starting from a polyline;
- the polylines cannot include among their elements ellipse arcs or splines.

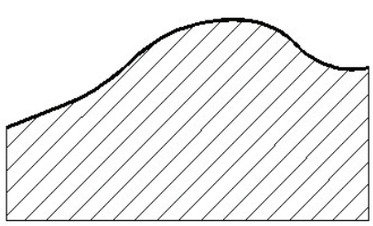


Fig. 11. Section through a freeformed part. Useful elements

Keeping in mind those stated above, the following problems have to be solved:

- approximate the ellipse arcs and the spline curves with sequences of lines and/or arcs;
- assembling these approximations into polylines;
- finally, determining and programming the tool-path.

3.4.1 Approximation the arc of ellipse by lines

A curve, no matter what type it is, can be approximated more or less precisely by a sequence of lines. How many lines should replace a curve? This is a question hard to be answered, because in measuring the quality of the output, two criteria operate: the more segments, the better approximation is, and fewer segments means less tool-path direction changes, that is better dynamic conditions for the machine-tool. Of course, the most important criterion is that of geometrical precision of the machined surface. This must not lead to the impression that a however big number of segments is acceptable, because anyway the computations are easily and fastly performed by the computer. Always a certain admissible deviation from the nominal shape is foreseen. This is the main input that must be taken into account when determining the number and length of the segments that approximate a curve: the maximum distance between the curve and any of the segments (chords of the curve) may not exceed the admitted deviation from the nominal shape.

Determining the set of segments that approximates an arc of circle is simple, because of the curvature of the circle is constant, and consequently all the segments are equal in length. Finding the substitute of an ellipse is more complicated because of the variable curvature of this curve. This means that to maintain the same distance between the curve and any of the segments, these must differ in length (Fig. 12).

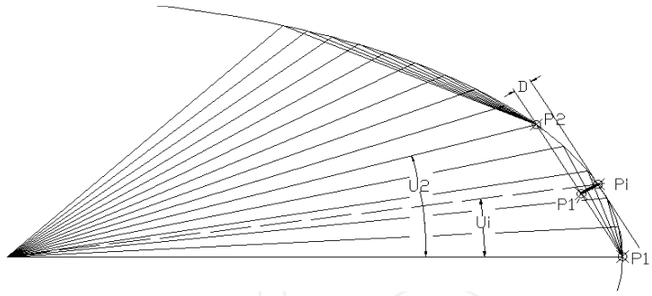


Fig. 12. Data required to compute the distance between the curve and chord

The computing algorithm is as follows: an included angle is such chosen, as it is small enough to ensure that the distance (D) between the ellipse and chord is for sure less that the admitted deviation. This angle is progressively increased towards point P_2 and compute again the distance (D) between the curve and chord. Always the previous point is stored. The procedure is repeated until the distance (D) overcomes the admitted deviation. When this happens the previous point is put in the list of those that describe the approximation. This point becomes P_1 for the next arc. Computing the distance between the curve and the arc is performed as follows: compute the distance in ten intermediary points and store maximum value. This is the distance (D). To perform these calculi the following stages are passed:

- the coordinates of points P_1 and P_2 that end the arc are computed using the parametric coordinates of the ellipse (6);

$$x_i = x_c + a * \cos U_i ; y_i = y_c + b * \sin U_i \tag{6}$$

Where x_i, y_i, x_c, y_c are the coordinates of the current point and of the centre of ellipse, respectively

- write the equation of the chord determined by points P_1 and P_2 (7);

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} * (x - x_1) \tag{7}$$

- to be used later, equation (7) is put in its general form Ax+By+C=0, which has the quotients (8);

$$A = \frac{y_2 - y_1}{x_2 - x_1}; B = -1; C = y_1 - \frac{y_2 - y_1}{x_2 - x_1} x_1 \tag{8}$$

- the intermediary points P_i are determined using equations (6);
- the distance d is computed using equation (9)

$$d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}} \tag{9}$$

Some supplementary calculi have to be performed if the ellipse is rotated (has not the axes parallel to those of the coordinate system).

3.4.2 Approximation the spline curves by lines

Approximation the spline curves by segments is performed with a user defined function – TranSpline – using the data in the associated list of the entity. For that is applied the hypothesis that the spline curves obtained by sectioning solids are smooth and have not big curvature. A spline is generally defined by control points (code 10 in the associated list) and knots (code 11 in the associated list). Spline curves result when sectioning cones, cylinders or thoruses, if the sectioning plane is not perpendicular or parallel to the solid's axis. Such curves have not control points, but knots only. In Fig. 13 a spline is shown and then it's associated list.

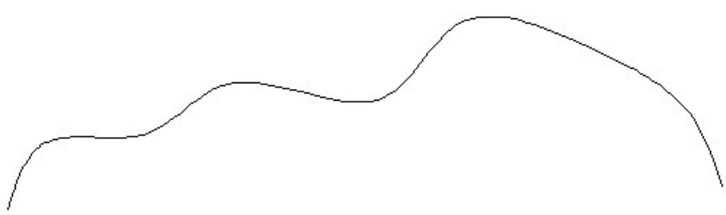


Fig. 13. Spline curve

(44 . 1.0e-010) (40 . 0.0) (40 . 0.0) (40 . 0.0) (40 . 0.0) (40 . 5.14612) (40 . 12.6226) (40 . 18.5784) (40 . 30.3778) (40 . 37.0406) (40 . 47.4409) (40 . 55.131) (40 . 60.5652) (40 . 60.5652) (40 . 60.5652) (40 . 60.5652) (10 45.72 8.55 0.0) (10 46.31 10.42 0.0) (10 47.77 15.01 0.0) (10 55.81 12.13 0.0) (10 61.47 20.31 0.0) (10 72.13 12.74 0.0) (10 77.06 24.35 0.0) (10 86.71 19.79 0.0) (10 93.77 16.32 0.0) (10 95.17 11.98 0.0) (10 95.75 10.18 0.0) (11 45.72 8.55 0.0) (11 48.17 13.08 0.0) (11 55.603 13.89 0.0) (11 60.58 17.15 0.0) (11 72.37 16.52 0.0) (11 76.99 21.32 0.0) (11 87.23 19.51 0.0) (11 93.57 15.16 0.0) (11 95.75 10.18 0.0))

The function for approximation the spline is presented bellow.

Program line	Effect
(defun C:TranSpline ()	
(command "UNDO" "BE")	
(setq osvechi (getvar "OSMODE")	
curba (car (entsel))	Select the spline to be processed
lista (entget curba)	Get the associated list
l lista	
lp1 (list) lp2 (list))	
(while (/= (caar lista) 10)	Remove from the associated list the irrelevant elements for the geometrical data
(setq lista (cdr lista))	
)	
(while (= (caar lista) 10)	Gather the control points of the spline
(setq lp1 (cons (cdr (assoc 10 lista)) lp1)	
lista (cdr lista)))	
)	
)	
(while (= (caar lista) 11)	Gather the knots of the spline
(setq lp2 (cons (cdr (assoc 11 lista)) lp2)	
lista (cdr lista)))	
(setq lp1 (reverse lp1)	
lp2 (reverse lp2)	
i1 (length lp1)	
i2 (length lp2)	
i 0	
)	
(command "line"	Draw a line through the control points (if they exist)
(repeat (1+ i1) (command (nth i lp1))	
(setq i (1+ i))	
)	
(command "")	
)	
(setq i 0)	
(command "color" 5)	
(command "line"	Draw a line through the knots
(repeat (1+ i2) (command (nth i lp2))	
(setq i (1+ i))	
)	
(command "")	
)	
(command "color" "bylayer")	
(command "Erase" curba "")	
(command "Pedit" "l" "Y" "J" "all" "" "")	Join the segment drawn through the knots into a single polyline that approximates the spline
(setvar "OSMODE" osvechi)	
(command "UNDO" "E")	
)	

Table 8. TranSpline user-defined function. Explanations

In Fig. 14 two splines are shown together with their approximations. The curve a) has big inflexions; consequently the polyline drawn through the control points is distant from the curve itself. The polyline drawn through the knots has large deviations from the original object (emphasized in circles), as well. The curve b) obtained by sectioning a 3D object is smooth, has not control points, and the polyline drawn through the knots has not visible deviations from the original curve.

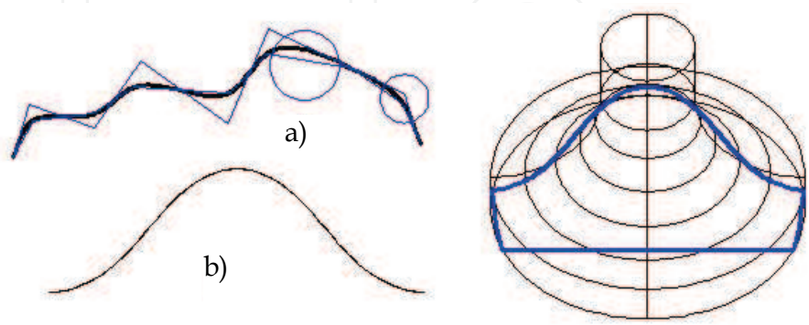


Fig. 13. Aproximation of the curves by polylines a) some spline, b) smooth spline

Assembling the entities that constitute the polyline that approximates the spline is performed using the PEDIT command of AutoCAD. To ensure a correct selection of those entities, the section they come from is placed in a special layer, the single one thaw at the moment of the processing.

3.4.3 Finding the tool-path for milling finishing in vertical planes

Finding the tool-path in vertical planes that is going to be programmed raises some problems if use a ball nosed milling cutter (the most used for finishing). The difficulties occur if the slope of the profile varies, because the contact point of the tool with the surface machined migrates along the tool profile, as shown in Fig. 14. Other approaches are presented in (Bo.& Bangyan, 2006; Kim & Yang, 2007).

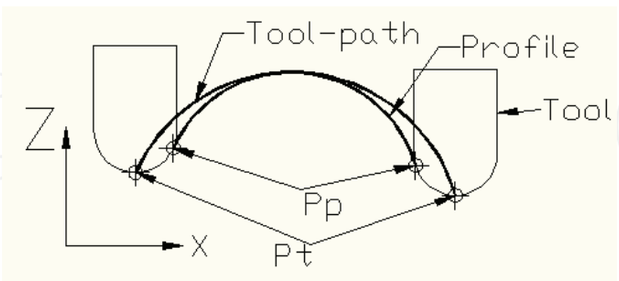


Fig. 14. The contact point Pp migrates along the cutting edge

Because of the tool radius compensation mechanism does not work in a vertical plane, the position of the programmed point must be permanently evaluated. The way the profile slope influences the relative position of the contact point (P_c) between tool and part, on one hand, and the programmed point on another hand is depicted in Fig. 15 (Drăgoi, 2009). Furthermore, in Fig. 15 can be seen the distortion of the tool-path reported to the machined profile.

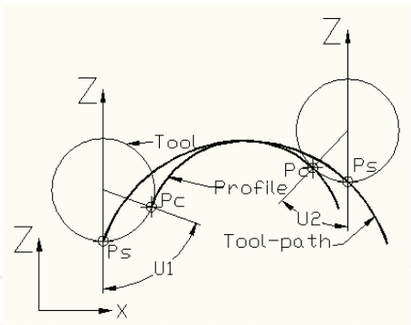


Fig. 15. The relative position of the contact point (P_c) and the programmed one (P_s)

In Fig. 16 are presented the data required to compute the coordinates of the programmed point as a function of the point on the part profile.

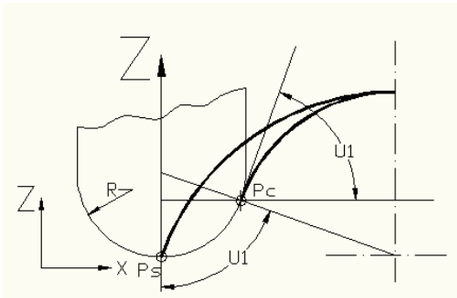


Fig. 16. Computing the position of the programmed point

The coordinates of the programmed point are computed with equations (10) and (11) related the coordinates of the point on the profile, tool radius, and the slope of the profile.

$$X_{Ps}=X_{Pc}-R*\sin(U1); \tag{10}$$

$$Y_{Ps}=Y_{Pc}-R*(1-\cos(U1)); \tag{11}$$

Where R=tool radius, U1=the contact angle, equal to the profile slope, P_c=contact point, and P_s=point to be programmed.

Although computing the coordinates of the programmed point is not difficult, it becomes a problem in the case of curved profiles, where the slope varies continuously along the profile. Because of computing the tool-path point by point cannot be a solution, an AutoLISP user-defined function was designed to solve this problem by means of AutoCAD. The stages of the algorithm are the following:

1. section the 3D model of the part by vertical planes;
2. the region obtained is exploded into the constituent entities;
3. keep only those entities that describe the profile to be machined, not the sides or the bottom of the former region;
4. join the entities retained at the previous stage into a polyline;
5. apply *Offset* command on the polyline, above, at a distance equal to the tool radius;
6. move the output of *Offset* command downward with a displacement equal to the tool radius.

The curve obtained at the stage 6 is the tool-path that must be programmed in the CNC file. The new offset seems to be (and it is) distorted from the part profile. This curve is not an

equidistant to the profile, since a point of it (P1) lies on the profile, and another point (P2) is at a certain distance in any other position. Furthermore, that distance varies permanently, related to the slope of the profile, as can be seen in Fig. 17.

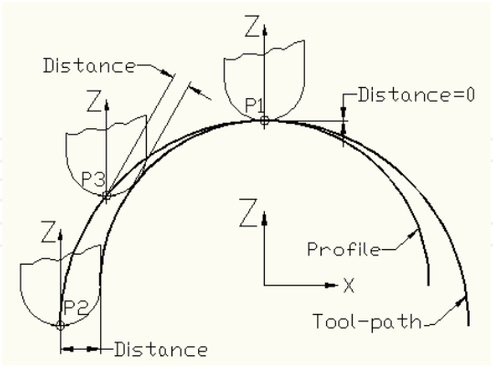


Fig. 17. The tool-path is not equidistant to part profile

This manner to determine the tool-path has another major advantage against computing the coordinates of the programmed point: the equations for computing the coordinates must be modified if the surface to be machined is concave instead of convex. In other words, the equations are different if machining inside or outside the profile.
In practice, problems may occur if some sectors of the profile are curves that cannot be joined to a polyline (i. e. ellipse arcs or splines). In such cases, if every entity is offseted individually the output cannot be used as a tool-path because of the discontinuities (Fig. 18a).

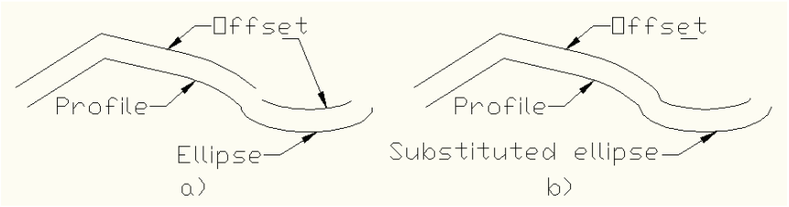


Fig. 18. Offsetting entities individually is not a solution, because of the discontinuity of the output

To overcome this disadvantage, the arcs of ellipse and splines must be substituted – as have been described above in this chapter - by sequences of lines, which can be joined to a polyline. Such a polyline can now be offseted, and the output can act as tool-path (Fig. 18b).
Everything has been presented so far related to tool-path finding for milling in vertical planes is only the theoretical support of the problem. Solving the problem is the output of the AutoLISP user-defined function called VertMill, which steps through the following stages:

1. opens the file where the CNC program will be saved;
2. gets by means of 3Dsize function the sizes of the 3D model;
3. adopts the sizes of the billet and generates the billet;
4. builds an axial section of the milling cutter that will be used later;
5. sections successively the part with vertical planes, YZ or ZX as the user chooses;

For each section the following tasks are performed:

6. align the UCS to the region obtained by sectioning. Aligning is necessary in order to perform later some specific 2D operation that can only be done in the XY plane;
7. explode the region in the current XY plane;

8. every entity obtained by explosion is examined by means of the associated list.
9. the arcs of ellipse and splines are substituted by sequences of lines;
10. the old arcs of ellipse and splines are deleted;
11. all the segments and arcs (if they exist) are joined into a polyline;
12. an offset is drawn above the profile at a distance equal to the tool radius;
13. the offset is moved downwards with a displacement equal to the tool radius – this is the tool-path;
14. gather from the associated list of the offset its' geometrical data;
15. coordinate transformations are applied to the points on the offset (they are in the current XY plane, and must be transferred to the YZ or ZX plane of the part);
16. The new coordinates are written to the CNC file;
17. sweep the section of the tool (obtained in stage 4) along the tool-path;
18. the solid obtained at the previous stage is subtracted from the billet. Stages 16 and 17 constitute the components of the machining simulation module;
19. the vertical plane is moved with an increment equal to the transversal feed-rate;
20. stages 6 to 18 are repeated until the 3D model is entirely processed;
21. close the CNC file.

Only the main stages have been here presented. Many other auxiliary actions are necessary to perform all the transformations: managing the layers and UCSs, moving entities from a layer to another to facilitate the required entities selections, operating with OSNAP modes, saving some system variables at the beginning of the program and restoring it at the end of the program, many computations.

4. CAD for Rapid tooling

Rapid tooling is, in the widest sense, the fast making of tools, in the context of manufacturing engineering. Particularly the *Rapid tooling* syntagma is associated with rapidly producing mould inserts for injection moulding of plastics and casting of metals. As a step towards producing the tools mentioned before, CAD of inserts for injection of plastic mass parts is very important. Some aspects related to the design of inserts, having as data input the 3D model of the part itself are discussed bellow. For illustration, take as sample a mould to obtain the plastic shell for packing a tooth-brush and a mould for plastic injection of the tooth-brush's handle.

4.1 3D modelling of the product

As already has been mentioned, in the context of simultaneous design, even in the CAD stage must be taken into account the technological aspects. That is, the shape of the part has to be adequate to the technology it will be made by. In order to have the input for designing the plastic shell for packing, not the material and inner details are important, but the outer shape of the part. The shape of the proper tooth-brush is obtained by extruding a 2D contour. The handle is built using the LOFT command of AutoCAD applied on a set of elliptical sections aligned to the same axis (Fig. 19).

Taking into account of the shape of the handle (the cross section is elliptical), the plastic shell will not follow exactly the aspect of the product; that is the shell does not mould the handle under it's the horizontal medial plane, noted with P-P in Fig. 20.

Consequently, in order to design the mould, the model of the product must be adjusted. For that, the handle is sectioned by plane P-P and the region obtained is extruded downward (Fig 21).

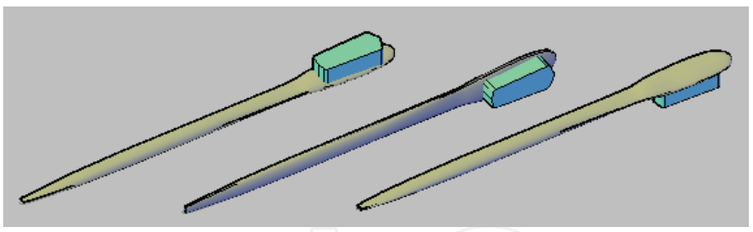


Fig. 19. The 3D model of the product

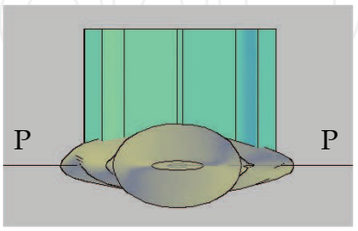


Fig. 20. The 3D model of the product. Detail

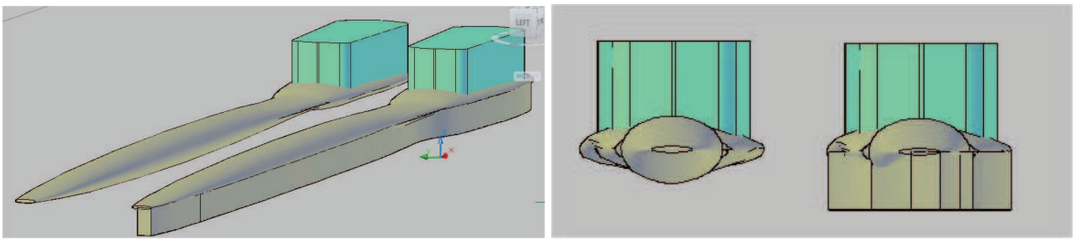


Fig. 21. The 3D model adjusted in order to design the mould

4.2 Design of the mould used to shape the plastic shell

Modelling the mould supposes building the 3D models of the inserts. For this, first of all a model of the shell itself must be created. To do that the command of AutoCAD SOLIDEDIT/Body/Shell is used. To get the shell outside the model of the product, a negative offset distance is used. The lower face of the object is removed. The output is shown in Fig. 22.

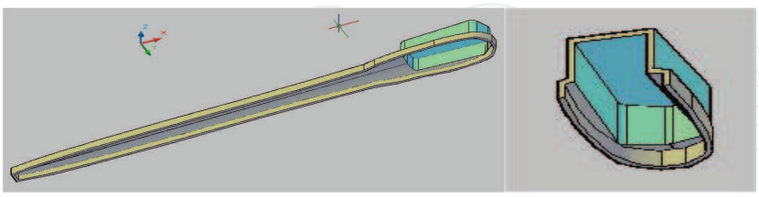


Fig. 22. The model of the plastic shell

Later, this shell is subtracted together with the model of the product from a box-shaped body, to get the upper insert of the mould (Fig.23a). The lower insert is obtained unioning the product 3D model to a box-shaped object (Fig.23b). Of course, in this way, only the active parts of the mould were obtained, forwards all the other details being designed (determination if part retainer or "chase" is required, incorporation of "locks" to prevent shifting of core and cavity, specification of the ejection required, if any, finding the size and location of spruem runners, and gates, determination of the size, number, and location of water lines for temperature control, if required).

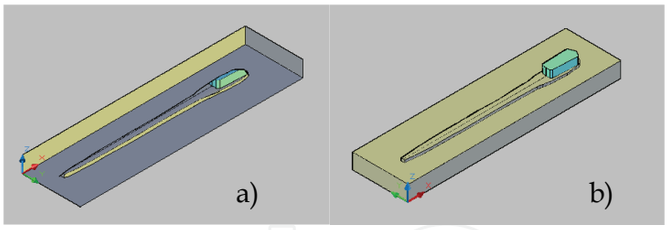


Fig. 23. The inserts. The upper one – a), the lower one – b)

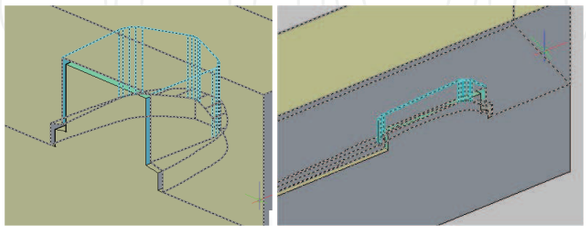


Fig. 24. The two inserts assembled (two sections - details)

The sequence of operations presented was turned into an AutoLISP user-defined function named Mould1. Its important stages are presented in Table 9.

Program line	Effect
(defun C:Mould1 ()	
.....	
(setq produs (car (entsel "\n Select the object to be processed ")))	Select the object to be processed
(setq g (getreal "\The thicknes of the shell "))	Input data
(command "copy" produs "" "0,0,0" "")	Make a copy of the object
(setq cs_prod (entlast))	Give a name to the object
(command "copy" produs "" "0,0,0" "")	Make a new copy of the object
.....	
(command "Solidedit" "B" "S" copie "" (- g) "" "")	Create the shell
(Princ"\nBuild the two inserts: ")	
(setq P1 (getpoint "\nFirst corner of the insert:")	Input the sizes of the inserts
P2 (getpoint "\nOther corner ")	
h1 (getreal "\nUpper insert height:")	
h2 (getreal "\nLower insert height::")	
)	
(command "BOX" P1 P2 h1)	Make the rough inserts
(setq sup (entlast))	
(command "BOX" P1 P2 (- h1))	
(setq inf (entlast))	
(command "-layer" "N" "Produs" "")	Manage the layers
(command "CHPROP" produs "" "LA" "Produs" "")	
(command "-layer" "F" "Produs" "")	
(command "Subtract" Sup "" copie cs_prod "")	Make the active parts of the inserts
(command "Union" inf ci_prod "")	
)	

Table 9. Mould1 user-defined function. Explanations

4.3 Design of the mould used to make the product by plastic injection

Modelling the inserts of the mould supposes making the cavities in the inserts, the plastic mass is going to be injected in. The sample is the tooth-brush handle. The main stages of the process are as follows:

- scaling the model with the thermal dilatation quotient;
- defining the billet;
- subtracting the product 3D model from the billet;
- separating the two inserts by slicing the billet with a horizontal plane at he level of the separating plane

The sequence of operations presented was turned into an AutoLISP user-defined function named Mould2. The main stages are presented in Table 10.

Program line	Effect
(defun C:Mould2 ()	
(command "Undo" "BE")	
(setq os (getvar "osmode"))	
(setvar "OSMODE" 0)	
(setq produs (car (entsel "\ Select the object to be processed ")))	Select the object to be processed
(setq P1 (getpoint "\nFirst corner of the insert:")	Input data to describe the sizes of the inserts
P2 (getpoint "\nOther corner ")	
h1 (getreal "\nUpper insert height:")	
h2 (getreal "\nLower insert height:")	
Sep (getpoint "\nThe level of sepparating plane ")	Input the separating plane
)	
(command "UCS" "O" Sep)	
(setq coef (getreal "/n thermal dilatation quotient:"))	
(command "Scale" produs "" "0,0,0" coef)	
(command "BOX" P1 P2 (+ h1 h2))	Make the rough insert (unique for now), and name it
(setq pastile (entlast))	
(Command "move" pastile "" (list 0 0 (- h2)) "")	
(Command "Subtract" pastile "" produs"")	Make the active part of the insert
(command "Slice" pastile "" "XY" "0,0,0" "B")	Separate the inserts
(command "-layer" "N" "Produs" "")	
(command "CHPROP" produs "" "LA" "Produs" "")	
(command "-layer" "F" "Produs" "")	
(setvar "OSMODE" os)	
(command "Undo" "E")	
)	

Table 10. Mould2 user-defined function. Explanations

After run the function, the ensemble in Fig. 25 is obtained (In Fig. 25 slices of the inserts are presented, as the inner details to be visible).

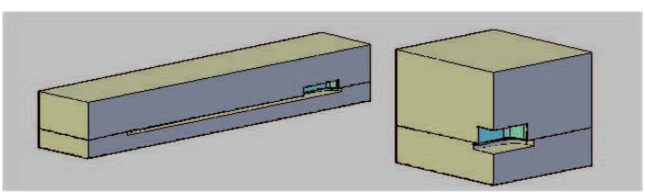


Fig. 25. The mould inserts

4.4 Application. Modelling the mould inserts for plastic mass injection

The application uses the function Mould2 to design the active part of the inserts of a mould for plastic mass injection. In Fig. 26 the part to be produced is presented in two 3D views. In Fig. 27 a slice through the two inserts knocked together, and in Fig. 28 the two inserts individually are depicted.

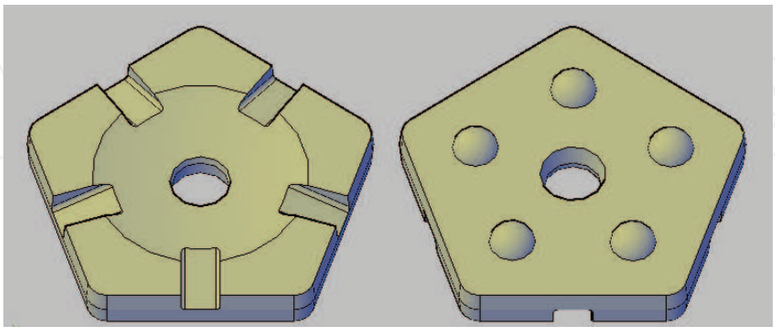


Fig. 26. The part to be processed. Special roundel. Up and down view

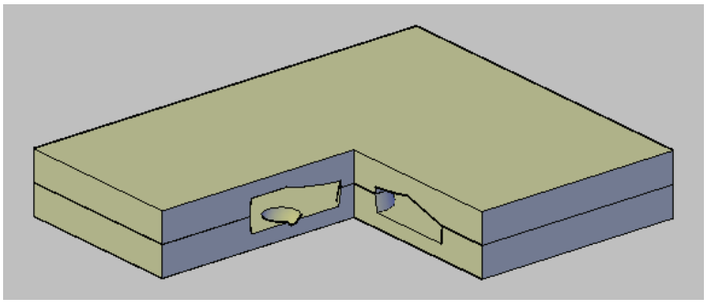


Fig. 27. The mould inserts. Slice to view inside details

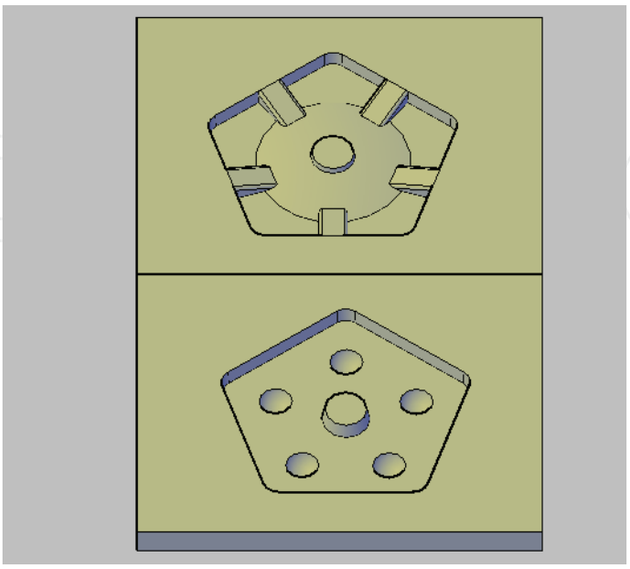


Fig. 28. The pair of inserts

5. Conclusion

The approach of the subject here presented is a unitary one, all the solutions proposed being obtained by means of AutoCAD and AutoLISP. The set of AutoLISP functions cover several specific tasks of the manufacturing engineer. The manufacturing engineer is thought as a specialist that acts both in the constructive and the technological sides of design, being able to integrate them. The integration goes now further, being applied to design and manufacturing by means of CAD/CAM systems. Two new user-defined functions for the constructive-technological design were described (EXDATROUG1 and EXDATROUG2). New techniques of scanning 3D models in horizontal and vertical planes are set up, as means of checking the machinability of the parts by 2 ½ or 3 axes CNC machine-tools.
2D CNC milling processing is targeted by the new user-defined functions DXF_CNC and LAsoc_CNC. Means to aid the engineer in designing the active inserts of some moulds are also presented in this chapter.

5.1 Future research

After presenting the new tools offered to manufacturing engineers, some questions might rise: are these tools good as they are? Are they too specialized and too focused to narrow problems, or are they too general? The answers can be given only by those interested to use it, after they will get benefits from it, or will find certain drawbacks.
Future researches are foreseen to improve the tools as potential users would suggest. Here could be mentioned extending the set of functions consecrated to mould design with new ones that make easier the job of designing it in terms of determination if part retainer is required, incorporation of "locks" to prevent shifting of core and cavity, specification of the ejection required, if any, dimensioning and placing the water lines for temperature control, if they are necessary, and others.
New useful tools are intended to be provided in the area of reverse engineering, namely reconstituting the 3D model of a part, having as input the CNC file used to produce the object.

6. Acknowledgement

Some achievements presented in this chapter (2.1, 3.2, 3.3, 4.1, 4.2, and 4.3) were obtained within the research project Sisteme Expert de Optimizare a Proceselor Tehnologice (Expert System for Optimisation of Technological Processes) – ESOP, PN II, 2007-2010; nr. 71-133/18.09.2007 funded by the Romanian Ministry of Education and Research.

7. References

Bo, Z. & Bangyan, Y. (2006). Study on 3D tool compensation algorithm for NC milling, Proceedings of *International Technology and Innovation Conference 2006 (ITIC 2006)*, pp. 290 -294, ISBN: 0 86341 696 9, Hangzhou, China, 6-7 November 2006

Chicoş, L.A. & Ivan N.V.,(2004), Programmes Package Regarding Concurrent Engineering for Product Development - New Applications, *Academic Journal of Manufacturing Engineering (AJME)*, Vol.2, Nr.2, (June 2004), pg.31-37, ISSN: 1583-7904,

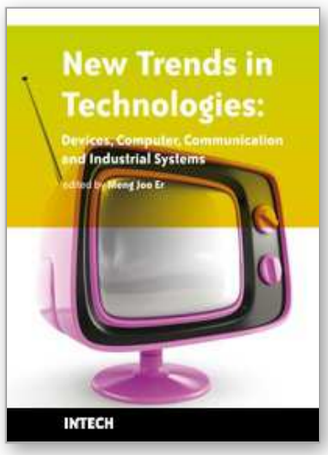
Chicoş, L.A., et al., (2009)., Simultaneous approach of CAD and CAM technologies using constructive-technological entities, *Annals of DAAAM for 2009*, Volume 20, No.1 (November 2009), pp.377-378, ISSN 1726-9679.

Drăgoi, M. V., (2009,) Ball Nose Milling Cutter Radius Compensation in Z Axis for CNC.. Proceedings of *the 8th WSEAS International Conference on Recent Advances in Software Engineering and Parallel and Distributed Systems (SEPADS '09),* pp. 57-60, ISBN 978-960-474-052-9 Cambridge, UK, February 21-23 2009.

Ivan, N. V. (1994), Prozessoren für geometrische und technologische Beschreibung der Werkstücke. *Proceedings of 5th Internationales DAAAM Sympozium*, pp. 173-174, ISBN 86-435-0084-4, Maribor, Slovenia, October 1994, University of Maribor, Maribor

Ivan, N. V., et al., (1996) Technological Processor Structure. *Buletin of Transilvania University of Brasov, New Series*, A, Vol 3 No 38, (October 1997) p.129-134. ISSN 1223-9631

Kim, S.-J. & Yang, M.-Y.(2007), Incomplete mesh offset for NC machining", *Journal of Materials Processing Technology* Vol. 194 No. 1-3 (November 2007), pp. 110–120, ISSN: 0924-0136



New Trends in Technologies: Devices, Computer, Communication and Industrial Systems
Edited by Meng Joo Er

ISBN 978-953-307-212-8
Hard cover, 444 pages
Publisher Sciyo
Published online 02, November, 2010
Published in print edition November, 2010

The grandest accomplishments of engineering took place in the twentieth century. The widespread development and distribution of electricity and clean water, automobiles and airplanes, radio and television, spacecraft and lasers, antibiotics and medical imaging, computers and the Internet are just some of the highlights from a century in which engineering revolutionized and improved virtually every aspect of human life. In this book, the authors provide a glimpse of new trends in technologies pertaining to devices, computers, communications and industrial systems.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Mircea Viorel Dragoi (2010). Advances in CAD/CAM Technologies, New Trends in Technologies: Devices, Computer, Communication and Industrial Systems, Meng Joo Er (Ed.), ISBN: 978-953-307-212-8, InTech, Available from: http://www.intechopen.com/books/new-trends-in-technologies--devices--computer--communication-and-industrial-systems/advances-in-cad-cam-technologies

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen