Quality Festival 2023
ISBN 978-86-6335-104-2

Predrag Mitić [1]
Marija Zahar Đorđević
Vuk Petronijević
Nebojša Abadić
Aleksandar Đorđević

AUTOMATIC TOOL PATH GENERATION IN CONTOUR MILLING USING GENETIC ALGORITHM

Abstract: *The purpose of this paper is to present new approach in automatic tool path generation in contour milling based on genetic algorithm and bitmap representation of work piece and additional resources. It deals with the problem of tool path generation and optimization in contour milling which is the most common case in metalworking industry. The starting point is assumption that the geometry of initial working piece, machined part and clamping fixtures is represented as bitmaps, then the mathematical model is presented and genetic algorithm is used to generate and optimize tool path. Proposed approach greatly reduced the costs of part production through improved machining efficiency, realized through independent software solution implemented in object-oriented language Delphi and can be used as starting point for fully autonomous NC code generation.*

Keywords: *Genetic algorithm, optimization, milling tool path, bitmap representation*

1. Introduction

The problem of tool path generation and optimization in chip removal machining processes has attracted the attention of a large number of researchers for several past decades. Machining in general and milling in particular is one of the main production processes used to manufacture durable goods. The cost optimization of production processes remains one of the major focus points of machine builders world-wide. (Khatiwada, Nepali, Raj and Bhattarai, 2020). The conventional ways of selecting the tool path or programming the NC code used data from machining handbooks and the knowledge of programmer for optimal processing (Bharath and Natraj, 2018). However, the conventional or traditional NC programming when compared to the modern CNC machining has many disadvantages for instance increasing time and cost production and, decreasing accuracy and quality of the work piece. In the literature, there is a growing trend of development and application of models for the automatic generation and optimization of toolpaths. The most common and most researched approach is the application of artificial intelligence and metaheuristics algorithms in tool path optimization and milling processing parameter’s selection, as well as the application of various geometry recognition techniques starting from a 3D model or from a 2D representation of the part. is reflected in the model setting and the modeling of goals and constraints.

Bharath and Natraj (2018) stated that determining the optimal machine tool path has been proven to lead to high productivity and minimal production costs. Furthermore, they provide statistical data related to the methods used by various authors in researches, with the conclusion that Genetic

Corresponding author: Predrag Mitić
Email: predrag2904@gmail.com

Algorithms (GA) and Particle Swarm Optimization (PSO) method are largely used to optimize machining efficiency and when compared with other methods, the GA has been successfully applied for many optimization problems with various parameters related to tool path and effective in improving the robustness of feature selection over a range of problems.

Kumar and Khatak (2018) define the tool path as a sequence of positions within the surface to be machined and divide the surface into a grid with a finite number of squares, where each square represents one element of the tool path and the square is viewed as one point. The goal of optimization is minimization of processing time, minimization of the time required for tool change and positioning in high-speed motion, and minimization of the effect of jerk on the machined surface. The optimization is performed by applying the GA. Makhanov (1999) presents a mathematical model of a variable grid of points as a basis for path optimization in five-axis milling and provides an algorithm that generates a tool path consisting of different segments that are adapted to the type of tool movement and increases the processing accuracy. Jacso, Matiasi and Szalay (2019), presented the model of tool path generation in contour milling, which is not based on the discretization of the surface being processed, but on equations from analytical geometry and vector calculus, which assumes that the angle of engagement of the tool is always constant. The generated tool path follows the shape of the contour, that is, it enables the processing of any shape of the contour, but the model does not foresee path optimization.

The model that integrates toolpath generation and optimization is provided in the research of Barclay, Dhokia, Nassehi (2015) The model is based on the discretization or pixelization of the surface, which is processed by levels, that is, the tool path is generated and optimized for the value of the axial depth of cut. The authors state that the problem of path generation in milling can be viewed as a TSP problem (Traveling Salesman Problem) that can be efficiently solved by applying GA. Like Kumar and Khatak (2018) the surface to be processed is modelled with a mesh with a finite number of squares. where each square represents a tool positioning point, i.e. a city through which a traveling salesman must pass, establishing an analogy with the TSP problem. They define the number of squares in the grid as finite, limited only by the resolution of the CNC machine. Car, Veza and Mikac (2004) and Essink, Nassehi, and Newman (2014) have an almost identical approach to problem modeling, with the only difference being in the structure of the GA (genetic operators, representation of genes and chromosomes, etc.). Oysu and Bingul (2009) presented an approach for milling tool path generation based on optimization of sequences of predetermined contours. This approach gives good results if there is a clear logic for the previous selection of contours,

From the previous brief literature review, it can be concluded that the presented models of automatic tool path generation and optimization are based to a lesser extent on a purely geometric approach or to a greater extent on model discretization and the application of metaheuristic methods for solving the problem of tool path generation and optimization. The lacks of the presented models are almost identical - they are applicable mainly to rough milling processes, they depend on the shape of the contour being processed and on the degree of discretization of the surface.

The goal of the research presented in this paper is to define and demonstrate through experimental results the concept of automatic generation and optimization of the tool path in contour 3-axis milling which is independent of the shape of the contour and applicable for both rough and finish milling as a starting point for fully autonomous NC code generation.

2. The problem definition and formulation

The basis for the automatic generation and optimization of the tool path is the technical documentation (technical drawing, 3D model of the part being processed, etc.) and also the initial shape of the raw material from which the part is made. The information contained in the technical documentation must be recognized in a form suitable for tool path generation. Figure 1 show a typical industrially inspired part in 3D model representation.

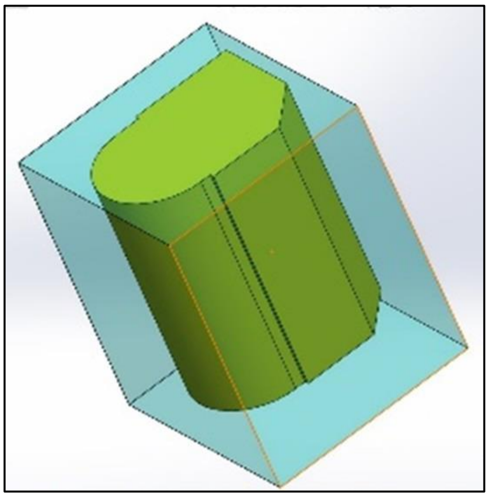


Figure 1. 3D model of work piece

Figure 2 shows a plane view of the same part, where the hatched surface represents the material that must be removed in order to obtain the shape marked in red, i.e. the final contour of the part. The shape and dimensions of the initail workpiece (marked blu in Figure 1) is predetermined and as such, represents one of the input data in the model.

In order to determine the path of the tool, it is necessary to find a measure of the discretization of the surface being processed (Barclay, Dhokia, and Nassehi 2015) i.e. to determine the size of the set of points and the position of the points that represent the positioning of the center of the tool, having in mind that the tool must be positioned at each point of that set, but also that its movement leads to processing the contour to the final size, that is, to enable the final processing. In order for the tool path to be generated, it is necessary first to determine the tool used to perform the processing and to determine the milling parameters-speeds, feeds and cutting depth. The process od automatic tool path

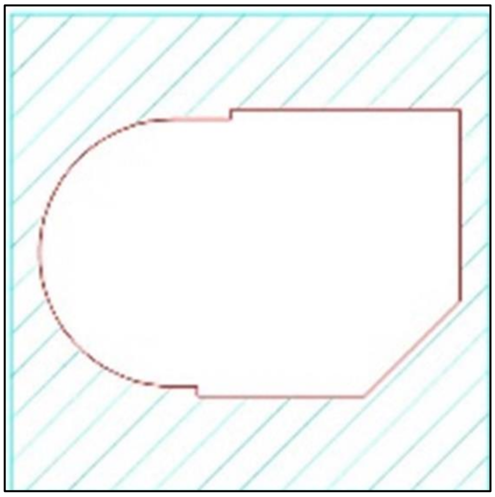


Figure 2. Planar representation of the work piece contour

generation in contour milling can be divided in three steps:

- determination of tools and milling parameters (speed’s and feed’s)
- discretization of the surface being processed
- tool path generation

2.1. Tools and milling parameters

The task of optimizing the contour milling machining is defined at the following way: select the milling parameters speed and feed that meet the quality limits surfaces and optimization criteria (Tanović, Petrakov 2007). The traditional methods for milling parameters calculation are widely known and are well described in the literature. Although the mentioned methods are applicable in everyday practice, nowadays, especially in the conditions of individual and small-batch production, data from the tool manufacturer's catalogue are most often used to determine the tool and milling parameters for the selected cutter. These catalogues are available in electronic form, so determining the tool and parameters using this data is very (ae) easy and fast. Based on the geometry of the surface and the required quality of processing, a tool of the appropriate diameter is chosen in such a way that all segments of the contour can be processed with that tool, but very often, due to the shape of the contour segments, several different tools must be selected. For the purposes research presented in this paper, it is considered that the milling process of the work piece is performed with one tool, because the principle of path generation for each subsequent tool is the same. The manufacturer of the tool according to the type of engagement (Figure 3) and according to the recommended values of the axial (a_p) and radial (a_e) depth of cut gives the recommended values of the milling parameters.

The radial (a_e) and axial depth of cut (a_p) are always given as a function of the diameter and material of the tool, the cutting speed Vc (m/min) is given depending on the material of the work piece, and the step per tooth fz (mm/tooth) is given depending on the diameter of the tool, the type of engagement and radial and axial cutting depths. For the purpose of this paper, the speeds and feeds will be considered as a constant value, while the radial cutting depth can exceed the value determined by the catalogue data, which depends on the contact surface of the tool and the work piece, which will be discussed later. The axial cutting depth is also constant value, but it depends on the depth of the contour, i.e. the height of the work piece, and in case the length of the cutting part of the tool corresponding to the selected sizes is smaller than the height of the work piece, the processing is performed in several passes according to the height of the work piece. Therefore, the number of revolutions of the main spindle, the speed of the auxiliary movement and the axial depth of cut are considered as constant value and with radial depth of cut along with diameter of the tool represent set of input data for the model presented in this paper.

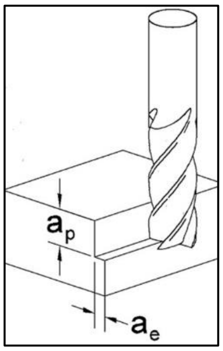


Figure 3. Tool engagement

2.2 A bitmap based discretization model of the machined surface

In contour milling, the tool path represents an ordered set of points where the tool is positioned, moving in the direction of the auxiliary motion velocity vector. If *n* is the number of points in that set, then theoretically there are *n!* potential toolpaths. If restrictions are taken into account, then that number decreases but is still very large. Discretization can be modelled as pixelization-mapping of the surface being processed with a large number of squares (Kumar and Khatak, 2018) basically cover the surface being processed very well, but then their number, that is, the resolution of the grid of squares goes up to 0.01 mm is the usual resolution of today's CNC machines. Such a data set is very large for processing by metaheuristic methods, especially GA, which can lead to the fact that it is not possible to find a solution that converges to the optimal one in an acceptable time (Barclay, Dhokia and Nassehi, 2015). Other approach is surface discretization by placing up an equidistant grid of points on the surface to be processed, but this method gives good results only in cases of rough contour milling (Nassehi, Essink, and Barclay, 2015).

However, if the surface to be processed is represented as a coloured bitmap image, these drawbacks can be avoided. Figure 4 shows initial bitmap image of working piece where the initial work piece contour is coloured blue and the finished part is coloured yellow. In order to discretize a milling, feature the following algorithm must be carried out on initial bitmap image.

1) Offset all contours by tool radius to determine the boundaries of the tool positions (Figure 5)
2) The contour of the finished part from the previous step offset by the value of allowance for fine milling (Figure 6)
3) Colour the finished part and all additional resources in yellow, the allowance for fine processing in red and the rest of the space inside the initial work piece contour in blue (Figure 7)
4) Pseudo code of point map algorithm for discretization of machined surface is shown below:

```
for i=1, bitmapwidth step $a_e$
 for j=1, bitmapheight step $a_e$
  if PixelColor(i,j)=blue then $P(i,j) \in M$
 next j, next i
```

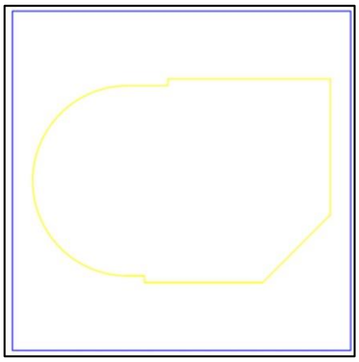


Figure 4. Initial stage of discretization

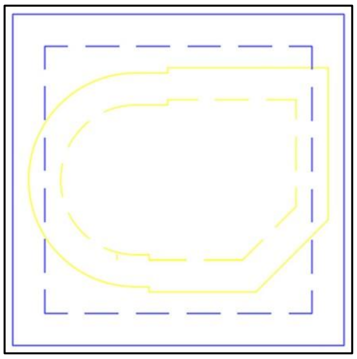


Figure 5. Offsetting feature boundary

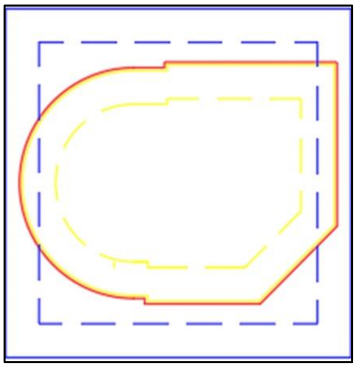


Figure 6. Offsetting for the fine milling

The colour of each pixel that is at a distance equal to the radial depth of cut a_e from neighbouring pixels is checked, and if it is coloured blue then it belongs to the set of points M through which the centre of the tool must pass in order to obtain the machined surface (Figure 8). In this way, all illegal moves of the tool are eliminated because a collision cannot occur between the tool and the work piece or between the tool and the clamping fixture.

Example:

bitmap width: X=2000 px

bitmap height: Y=2000 px

bitmap resolution: R=1000 DPI (px/inch) i.e. 39.37 px/mm

The dimensions of the initial workpiece are:

width: 2000/39.37=50.8 mm

height: 2000/39.37=50.8 mm

tool diameter: d=12

radial cutting depth: a_e=1.2 mm

resolution of the grid points for rough machining: 1,2x39,37=42,44 px/mm.

The grid of points for rough machining is defined as points map M= $[m_{ij}]$ and for above example has 762 points. For rough machining it is more than enough. For fine machining the same algorithm is processed, but with one difference. Every pixel is a point in point map for fine machining (pixels colored in red in Figure 7), and if the allowance for fine machining is 0,2 mm which is common value taken from everyday practice then the number of points in point map for fine machining is 672 points but with resolution of 0,02 mm/px (1/39,37 px/mm) which is enough for fine machining.

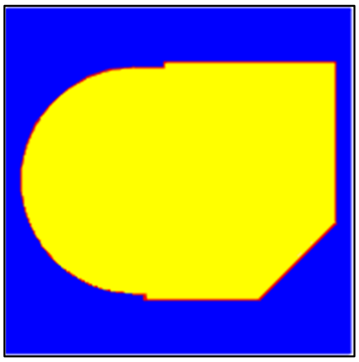


Figure 7. Step 3 of point map algorithm

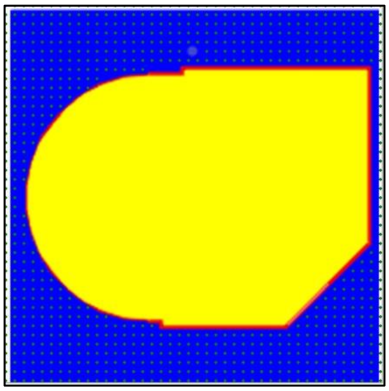


Figure 8. Imposed map of points

The necessary bitmap image (Figure 7) for executing the point map algorithm is very easy to obtain from any CAD software. Radial depth of cut a_e or *stepover* is the percentage of the tool diameter engaged in material (Figure 9).

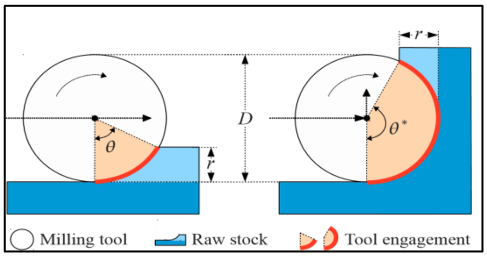


Figure 9. Radial depth of cut and TEA

The stepover determines the material removal rate (MRR) and reflects the cutting forces, but only for straight line motions. A parameter which better reflects the cutting force, regardless of the toolpath shape, is the *tool engagement angle-TEA* (Dimitrache, Borangiu and Dogar, 2010) as the amount of sweep subtended by each cutting edge as it engages and leaves the stock. For straight line motions, there is a direct, nonlinear relationship between tool engagement angle and stepover (Tanović and Petrakov, 2007). The engagement angle reaches its maximum (360°) when plunging the tool vertically into the material. The next maximum value, 180°, is encountered during a slotting operation; this condition may lead to high thermal stress on the tool, since the chips cannot be evacuated properly. Tool engagement is also known to increase at internal corners in toolpath. The engagement angle also has direct influence on the chip shape, therefore keeping TEA constant ensures consistent chip size and shape throughout the milling process (Dimitrache, Borangiu and Dogar, 2010). A large value of the TEA would result in a large amount of material removed, which certainly increases productivity, but then the cutting resistance is high and the wear of the tool is intense. As there are advantages and disadvantages of a larger or smaller milling angle, in the model considered in this paper, the milling angle Θ_{max} represents the input value in terms of the target value. Therefore, it is necessary to know TEA at every moment, that is, at every point of the tool path, and by comparing that value with the target, make a decision whether that segment is acceptable or not. In their research Wang, Jang and Stori (2005) presented in detail the process of calculation TEA by determining the overlap of the pixels of the section of the circumference of the tool and the current geometry of the work piece. This way, which is common in literature, involves the discretization of the tool and surface being processed into bit matrices of 0 and 1 and is highly dependent on resolution, which can be very time-consuming and even lead to the impossibility of real-time execution. In the next paragraph, a very simple method for calculating TEA will be presented, which is based on the simulation of the movement of the tool and bitmap representation of work piece.

2.3 Simulation algorithm for tool engagement angle calculation

By moving the center of the tool through the points of the points map M= $[m_{ij}]$, the chip is removed. As shown in Figure 10, the contact between the tool and the surface being processed is determined by the length of the circular arc between the first and last point of contact.

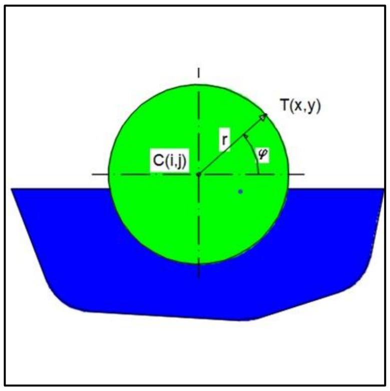


Figure 10. Calculation of TEA

Every point T on tool boundary can be represent by $x = r \cdot cos\varphi$ and $y = r \cdot sin\varphi$. If r=r+1 then all points outside the tool boundary will be checked but not on tool boundary, which is important because we need to number all blue pixels immediately behind the tool boundary, because the pixels are blue only in the points where the tool has not yet been. The pseudo code is shown below:

```
for every $[m_{ij}]$ do
n=1
r=r+1
 while $\varphi \leq 360^{\cdot}$ do
   $x = r \cdot cos\varphi; y = r \cdot sin\varphi$
if PixelColor(x,y)=blue then n=n+1
 $\varphi = \varphi + 0{,}1$
  end
end
$\theta_M = n/0{,}02\frac{mm}{px}$
If $\theta_M \leq \theta$ point C is acceptable
           else point C is illegal;
```

By simulating the movement of the tool as shown, it is very easy to calculate the TEA, and what is even more important, the condition of the work piece is known in any moment i.e. where the tool has passed (white colour) and where it has not (blue colour).

2.4 Mathematical formulation of the proposed model

A 3 axes CNC machine moves in both *x* and *y* directions simultaneously and Euclidean distance function is to be used to calculate the distance between points. In this way a distance matrix matrix D=[d_{ij}] is created.

Let *i* and *j* be two arbitrary points from set M.

Input variables

- geometry described by a bitmap image
- the set of points described by the point map M= [m_{ij}],
- the distance matrix D=[d_{ij}] between the points of the set M
- radial cutting depth a_e in mm
- maximum TEA θ_{max} in degrees
- tool diameter d in mm
- speed of main spindle S(mm/rev)

$$\forall j \in \{0 \dots n\} \sum_{i=0}^{n} Z_{ij} \cdot \psi_{ij} = 0$$

- feed rate F_{rh} (mm/min)
- rapid feed rate F_{bh} (mm/min)

Control variables of the mathematical model

$$X_{ij} = \begin{cases} 1 \; if point\ i\ is\ immediately\ \ by\ j \\ 0 \qquad otherwise \end{cases} \tag{1}$$

$$Y_{ijk} = \begin{cases} 1 \; if\ tool\ travels\ from\ i\ to\ k\ in\ same\ dir. \\ 0 \qquad otherwise \end{cases} \tag{2}$$

$$Z_{ij} = \begin{cases} 1 \; if\ j\ is\ machined \\ 0 \quad otherwise \end{cases} \tag{3}$$

Θ_{ij} *TEA when moving from i to j* (4)

Constraints of mathematical model

1) Constraint that ensure that every point from set M is visited by the tool at least once

$$\forall j \in \{0 \dots N\} \sum_{i=0}^{N} X_{ij} > 0 \tag{5}$$

2) Constraint that ensure that tool leaves each point after visting for another

$$\forall j \in \{0 \dots n\} \sum_{i=0}^{n} X_{ij} = \sum_{k=0}^{n} X_{jk} \tag{6}$$

3) Constraint that ensure that every point is followed by a different point and path moves on

$$\forall j \in \{0 \dots n\}\, X_{ii} = 0 \tag{7}$$

4) From each subset of points the tool must be positioned at least once in another point that is not part of the of that subset of points

$\forall Q \subset \{0 \dots n\}: Q = \emptyset$

$$\sum_{\{i.i \in N\}:\, i \in O.\, i \in N \setminus O} X_{ij} \;+ \sum_{\{i.i \in N\}:\, i \in O.\, i \in N \setminus O} X_{ji} \geq 2 \tag{8}$$

5) Constraint to ensure that the tool engagement angle Θ_{ij} must be smaller or equal than the maximum allowed engagement angle Θ_{max}

$$\forall ji\{0 \dots n\} \sum_{i=0}^{n} X_{ij} \cdot \theta_{ij} \leq \theta_{max} \tag{9}$$

6) Constraint that ensure that during the chip removal process, the movement of the tool is allowed only towards neighbour points.

$$\forall j \in \{0 \dots n\} \sum_{i=0}^{n} Z_{ij} \cdot \psi_{ij} = 0 \tag{9}$$

The objective function

The objective function of the mathematical model represents the criteria of optimisation. In the observed model the goal is to achieve high productivity, thus minimization of processing time, with minimization of the jerk effect with as uniform tool angle of engagement as possible.

Minimization of the processing time, can be written as:

$$F_{c1} = min \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\frac{d_{ij}}{F_{rh}} + \frac{r_{ij}}{F_{bh}} \right) \cdot X_{ij} \quad (10)$$

where d_{ij} are the distance between the points of the path in which the tool moves in a feed rate and r_{ij} the distance between the points of the path in which the tool moves in a rapid rate.

The minimization of the number of changes in the direction of movement of tool, taking into account the control variable defined by expression (2) can be written as:

$$F_{c2} = min \sum_{i=1}^{n-2} \sum_{j=2}^{n-1} \sum_{k=3}^{n} \frac{1}{Y_{ijk}} \quad (11)$$

The third goal of optimization is the smallest possible deviation of the TEA from the target value and can be defined as:

$$F_{c3} = min \sum_{i=1}^{n} \sum_{j=1}^{n} \left| \theta_{ij} - \theta_c \right| \cdot X_{ij} \quad (12)$$

that is, as a minimization of the deviation of the TEA value at each point of the path in relation to the target value.

The problem of milling path optimization is clearly multi objective optimization problem. For the purpose of this research the method of weight coefficients will be applied. Determining the weight coefficients can be a problem (Singiresu S.R., 2009) because the vector of weight coefficients controls the optimal solution. Mathematically, the optimal solution obtained with equal weighting coefficients should lead to the smallest conflict between the optimization goals, but in practice it is often not a satisfactory solution, so when determining the weighting coefficients, it is always necessary to have information in the order of priority of the goals. For the purposes of this paper, the greatest weight will be given to achieving maximum productivity so objective function is defined by following expression

$$F_c = 0{,}5 \cdot F_{c1} + 0{,}25 \cdot F_{c2} + 0{,}25 \cdot F_{c3} \quad (13)$$

3. Proposed algorithm fo milling path generation and optimization

An acceptable milling tool path represents permutations of points while respecting the constraints defined in the mathematical model which is basically an integer programming model. Genetic algorithms, as one of the modern, metaheuristic methods, are very suitable for solving this type of problem because in most cases they can find the global optimum with a very high probability (Singiresu S.R., 2009). It should be noted that the efficiency of the genetic algorithm depends on the applied genetic crossover and mutation operators (Umbarkar & Sheth, 2015). There are many literature sources related to this method, i.e. the GA's mechanism is widely known. Furthermore, only short definitions of GA-related terms and explaining the definitions in the observed model's context are given.

3.1. Definition of GA related terms in the context of observed model

A gene in the observed model is the primary carrier of informations. The point is denoted as M and every point that should be visited by the tool at least once contain the following informations:

- position defined by coordinates
- condition (machined or no)
- speed at which the tool moves towards the point (F_{rh} or F_{bh})
- ordinal number of point in the path

An individual or chromosome is represented by a combinations of genes and it is acceptable solution. In the observed model. this will be accetable tool path.

The population is a set of individuals, and in the observed model, it will be a set of all acceptable tool paths.

The parents are two accetable tool paths that combine to create new tool path.

Fitness is a function that tells us how good each tool path is. In the observed model, the path length is the fitness and it is defined by expression (13) in the mathematical model.

Crossover is the genetic operator defined as combining two individuals, i.e. tool paths to create new tool path.

The mutation is the genetic operator defined as the process in which one individual or one tool path participates, and the goal is to generate a new tool path. A mutation changes the value of one or more genes on a chromosome. The process of implementation of genetic operator's crossover and mutation in the observed model will be explained later.

3.2. Chromosome representation and decoding

In the construction of a genetic algorithm, the first step is to define an appropriate genetic representation or an appropriate coding method. An acceptable representation is the most critical factor influencing all other phases of the GA (Singiresu S.R., 2009). It primarily depends on the model being observed so that the chromosome coding solution with a vector of real components will be given.

As mentioned earlier in this paper the geometry of working piece is already recognized. The coordinates of every point is known so the distance matrix D=[d_{ij}] can be easily determined. In the observed model, the tool can pass through any point of the map of points several times, with the fact that it only passes once to remove chips, and it can pass through the same point again only for the purpose of approaching the cutting zone. Thus, the gene in the chromosome is marked with a natural number and represents the ordinal number of the point of the point map through which the tool passes. The tool can move from one point to another either at feed rate or rapid rate depending on whether the chip is removing or tool positioning is performed. Therefore, it is necessary to know the speed with which the tool moves from the point i and arrives at the point j. The next segment that describes the trajectory of the cutter is the engagement angle θ_{ij} which can be $\theta_{ij} = 0$ if the tool is moving at rapid rate or $\theta_{ij} > 0$ if the tool is moving at feed rate. In addition to knowing the x and y coordinates of each path point, it is also necessary to know the z coordinate, i.e. the plane in which the center of the tool moves and the condition of the point i.e. if is previously machined o no. An example of fully decoded chromosome is given in Table 1.

The decoded chromosome enables the creation of a CL (Cutter Load) file, that is, a file that represents the basis for generating NC programs and is generated by all today's CAM systems. For example, 3-5 record in the previous table shows that the tool moved from point 3(22, 6, 50) to point 5 (26, 6, 0) at rapid rate without removing chips by positioning in the plane z=50 (26, 6 , 50) and rapid move to z=50 (26, 6, 0). In this way, the entire path of the cutter from point

1 to point 6 can be read, with the order of movement of the tool being 1-3-8-5-4-2-10-7-9-8-3-5-6.

Table 1. Decoded chromosome

	1	3	8	5	4	2	10	7	9	8	3	5	6
F (mm/min)	F_{bh}	F_{rh}	F_{rh}	F_{rh}	F_{rh}	F_{rh}	F_{rh}	F_{rh}	F_{rh}	F_{bh}	F_{bh}	F_{rh}	F_{rh}
θ_{ij} (°)	0	40	42	25	30	35	39	41	42	0	0	0	30
x (mm)	20	22	24	26	28	30	32	32	34	24	22	26	26
y (mm)	6	6	6	6	6	6	6	6	6	6	6	6	10
z (mm)	50	0	0	0	0	0	0	0	0	50	50	0	0
Machined	true	false	false	false	false	false	false	false	false	true	true	false	false

3.3. Initial population

Using the distance matrix D=[d_{ij}], it is necessary to form another set of points which is very important for generating the initial population. That set K is the set of the closest points of each point of the point map, which contains all the k closest points to the point where the center of the tool is located. In the observed model, there can be up to 24 closest points to each point of the point map, and the x and y coordinates of the members of that set tell to which point the tool can move from any current point of the path. By forming the set K, a part of unacceptable or illegal paths is eliminated, i.e. the constraint defined by expression (9) is implemented. When all points of set K is visited by the tool then next random point of tool path is chosen. At the initial moment. only the points located in the first and last row and the first and last column of the map of points have the condtion *machined=true* and that point are peripheral points status of peripheral points, all other points have the status *machined=false*, i.e. the status of unprocessed points. It is clear that the first point of any milling path must be one of the points with status *machined=true*. Furthermore, the algorithm for forming the initial population is given below:

<u>Step 1</u>: Randomly select the initial point from the set of points M with machined=true

<u>Step 2</u>: For a randomly selected point from step 1, load the previously determined set of nearest points K to the randomly selected point.

<u>Step 3</u>: Randomly choose one point from the set K in the label j that has the status of an unprocessed point i.e. *machined=false*

<u>Step 4</u>: Calculate the milling angle θ_{ij} when moving from i to j.

<u>Step 5:</u> If $\theta_{ij} \leq \theta_{max}$, point j becomes a tool path point. If $\theta_{ij} \geq \theta_{max}$max then go to step 10.

<u>Step 6:</u> Assign status *machined=true* at point j.

<u>Step 7:</u> Calculate the vector product of vectors and $\vec{\imath} x \vec{\jmath}$ if the vector product is different from zero, increase the control variable t=t+1, if it is equal to zero, the control variable t keeps its previous value.

<u>Step 8:</u> Calculate $\left|\theta_{ij} - \theta_c\right|$.Control variable d=d+$\left|\theta_{ij} - \theta_c\right|$.

<u>Step 9</u>: Check the statuses of the neighboring points of point j. If the statuses are machined=true, then point j becomes a peripheral point, ie. the status of point j machined=true

Step 10: Repeat steps 3-9 for each point from the set K.

Step 11: Repeat steps 1-10 for each point in the set M until all points from the set M have status machined=true

Step 12: Repeat steps 1-11 for each individual from the initial population

By forming the initial population whose size is the input parameter of the genetic algorithm, the initial set of possible tool paths is formed. By choosing the parents, that is, two tool paths, and applying the genetic operators of crossover and mutation, and calculating goodness for each individual, that is, for each tool path, a path that converges to the optimal path is reached.

3.4. Fitness calculation

Fitness is a function that defines how good is the created tool path. This implements the goal function given by expression (13) in the mathematical model. The fitness calculation is done with each tool path in the population. The algorithm for fitness caluclation is simple and in fact it is the calculation of the variables ProcessingTime, DirectionChange and Deviation.

The ProcessingTime variable is determined as the time it takes the tool to cover the distance between each two adjacent points from the previously formed tool path, taking into account whether the tool moves between the points at a rapied rate or a feed rate. The ProcessingTime is calculated for each path from the population.

The variable DirectionChange is determined as the number of changes in the control variable t (defined in step 7 of the algorithm for creating the initial population) in the tool path. The number of changes in the control variable t is calculated for each tool path in the population.

The Deviation variable is determined as the sum of the differences between the TEA and the target value of the TEA. The sum of the differences is defined by the control variable d (step 8 of the algorithm for creating the initial population). The deviation variable is calculated for each tool path in the population.

When these variables are determined, the fittness of the tool path, is calculated. The calculation method is given by the following pseudo code:

```
Set the TopPath variable to 0
for I=1 do Population size in steps of 1
Assign the ith toolpath to the variable C1 C1=Path(I)
Assign the toolpath to the variable C2 with the index TopPath C2=Path(TopPath)
If
0.5*ProcessingTime(C1)+0.25*DirectionChange(C1)+0.25*Deviation(C1))<
((0.5*Processing Time(C1)+0.25*DirectionChange(C2))+0.25*Deviation(C2 )+ 0.25*Deviation(C2))
Assign the value of the i-th index to the variable TopPath, that is, save the value of the best fitness in the variable TopPath.
 Go to the next passage
```

3.5. The parents

The parents represent two toolpaths that combine to create new toolpaths. The selection of parents or paths is done by comparing their fitness. It is clear that the parent should be the path that has the shortest length (the greatest fitness), and when this is determined by comparing with the fitness of other paths, then the logical variable *parent* is assigned the value of truth, ie that path is chosen in the set of paths for crossover. The comparison is repeated until the number of iterations defined by the size of the set of parents, which represents the input size of the genetic algorithm, is fulfilled.

3.6. Crossover operator

The crossover is a process in which two individuals are combined to obtain new individuals, ie the selection process selects parents and the newly created individuals are children. The genetic material of a child is a combination of the genetic material of both parents. With the fact that in the problem of path optimization for milling processing, the same points can be included more than once in the same path, so specific crossover operators are used to implement crossing, so specific crossover operators are used to implement the crossing. The reviews of the most common crossover operators used in GA are given in (Umbarkar, Sheth 2015) and in (Padmavathi, Yadlapalli 2017). The Order-base Crossover (OX) operator, was chosen for the realization of crossover operation in this paper. It constructs an offspring by choosing a substring of one parent and preserving the relative order of the elements of the other parent. For example, the following two parent strings: (1 2 3 4 5 6 7 8) and (2 4 6 8 7 5 3 l), and suppose that a first cut point between the second and the third bit and a second one between the fifth and the sixth bit is selected. Hence, (1 2 - 3 4 5 - 6 7 8) and (2 4 - 6 8 7 - 5 3 1). The offspring are created in the following way. Firstly, the string segments between the cut point are copied into the offspring, which give (* * 3 4 5 * * *) and (* * 6 8 7 * * *). Next, starting from the second cut point of one parent, the rest of the elements are copied in the order in which they appear in the other parent, also starting from the second cut point and omitting the elements that are already present. When the end of the parent string is reached, we continue from its first position. In those example, this gives the following children: (8 7 - 3 4 5 - 1 2 6) and (4 5 - 6 8 7 - 1 2 3).

3.7. Mutation operator

Mutation is a process in which one individual or one tool path participates, and the goal is to generate a new tool path. A mutation changes the value of one or more genes in a chromosome. As the gene in the optimization model is a point with coordinates that describe it, which cannot be discarded, and the chromosome represents the path of the tool, there is a specificity of the mutation process in the observed model. Namely, in the classic TSP model, at the point where the mutation is performed, 0 turns into 1, and vice versa. The goodness value must be calculated for the newly created individuals. Thus, a new population is formed, with the same number of individuals as at the beginning. However, in the model for determining the tool path in the the contour milling, it is not possible to remove the gene, because the gene is a point, and the path must include all points at least once. For this, the mutation is done by the process of inversion, that is, by replacing the places of two points in the path with a correspondingly low probability, which represents the input data of the GA. After changing the points in the path, the acceptability of the newly created path is checked, that is, the angle of engagement is checked. Points are chosen randomly until points satisfying the previous condition are found.

Pseudo code of proposed GA

Input parameters for GA shown in this paper are: Set of point M, TEA target value, diameter of the tool, the population size, the number of parents, mutation rate and the number of generations.

```
Start
Enter the input parameters
Create Points map, Distant matrix and
Matrix of neighbors K  of each point
```

```
Create Initial population
Calculate the fitness of individuals of the
first generation
    Generation = 1
        Repeat
            Selection of parents
            Children = 0
            Repeat
        Pick of two parents for a crossover
        (Parent1, Parent2)
    Crossover OX (Parent1, Parent2)
    Mutation
    Children = Children + 1
    Until Children = population size-total
    number of parents
    Generation = Generation + 1
    Calculate the fitness of individuals for
    the current generation
Until generation = total number of
generations
End.
```

4. Presentation of the independent software solution, experimental results and discussion

The application in which the previously presented solution is implemented is written in the object-oriented programming language Delphi. Bitmap representation of the image is created base on industrially inspired part in Autocad. Figure 11. shows the initial form of the application. The application consists of several parts:

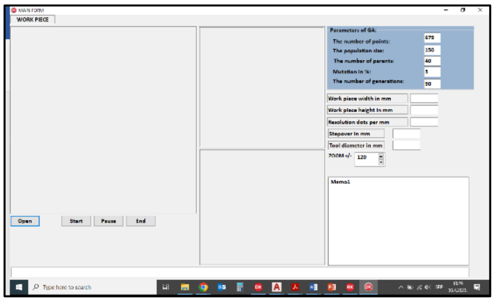

Parameters of GA:
The number of points: 679
The population size: 150
The number of parents: 40
Mutation in %: 5
The number of generations: 50
Work piece width in mm
Work piece height in mm
Resolution dots per mm
Stepover in mm
Tool diameter in mm
ZOOM +/- 120

Figure 11. The initial form of the application

In the left is a window where bitmap image is displayed, in upper right corner is the part for entering the genetic algorithm's input parameters, and in the middle there are panels that monitor GA executions, in which the graphic representation of the current tool path is simultaneously displayed. After loading the application, user must choose bitmap by pressing the Open button. By pressing the Start button execution of GA begins with creation of points map, as shown in Figure 12.

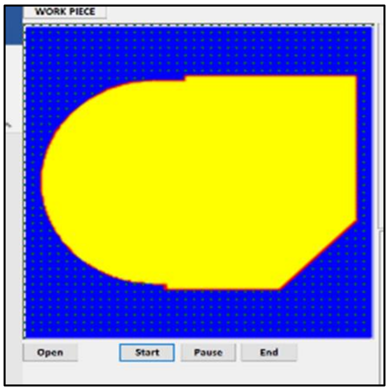


Figure 12. Creation of point map

The middle panel showing current data and graphical display during GA execution and the lower right part of the input screen represents the space in which, during and after the genetic algorithm's execution, the GA execution results are displayed. After the proces of generating and optimizing the tool path) the data necessary for NC code creation are written in a sort of CL (Cutter Load) file prepared for the processing in the appropriate post processors of CNC machines.

4.1. Experimental testing of proposed GA

Unlike many examples that can be found in the literature, where the results of GA execution are shown by varying the degree of mutation, population size, number of parents, etc., in this research, the proposed GA was previously tested by varying the previously mentioned parameters in order to find the optimal parameters. The tests were done with a constant value of TEA=40 degrees. Then GA was tested by varying TEA and radial depth of cut (stepover) according to Table 2 (Dimitrache, Borangiu and Dogar, 2010).

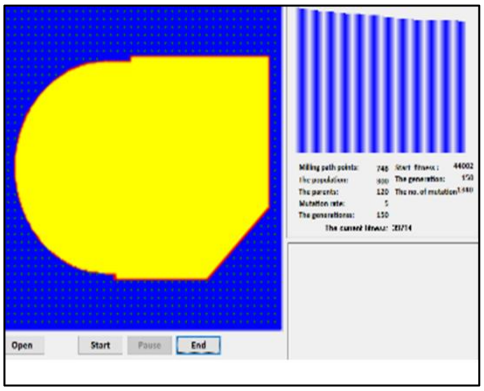
Table 2. Relation between a_e and TEA

a_e	10%	25%	50%	75%	85%	95%
TEA	37°	60°	90°	120°	135°	155°

During the extensive testing of GA, it has been shown to give the best results for:

- population size=300
- the number of parents=120
- mutation=5 %
- the number of generations=150

Fitness value was 39714 (Figure 13). With these parameters GA was tested for TEA value given in Table 2.



```
Milling path points:   748   Start fitness:        44002
The population:        300   The generation:         150
The parents:           120   The no. of mutation   1480
Mutation rate:           5
The generations:       150
          The current fitness: 39714
```

Figure 13. Results of GA testing

The results of GA testing for different TEA values are given in Table 3. It is clear that fitness or the path of the tool that converges to the optimal one, is highly dependent on TEA, because in the formula for calculating the fitness, the highest weight coefficient is given to the objective related to the length of the path.

Table 3. Relation between TEA and fitness of the proposed GA

Fitness	39714	20584	16249	11435	10024
TEA	40°	60°	90°	120 °	150 °

From the results shown in Table 3 it is clear that it is necessary to find the optimal ratio of the diameter of the tool, the tool angle of engagement (TEA) and the radial depth of the cut to have a tool path which converges to optimal one. The results of the previous experiment were needed to carry out the experiment in the real production conditions using the coordinates of the points obtained by executing the proposed GA.

4.2. Experimental testing of proposed GA in real production conditions

The conditions of the experiment are given in Table 4.

Table 4. Conditions of the experiment

CNC machine	CNC milling machine IBARMIA ZVX 2000
Control unit	HEIDENHAIN TNC407
The machine working space	2500x500x600 mm
Tool	Carbide millin cutter: HK16822161 Ø16x63x125 mm, 30° TiAlN
Initial work piece dimension	50x50x70 mm
The dimension of finished work piece	40x30x59,9 mm
Materijal	1.4301
Speed of the main spindle	S=1600 o/min
Feed rate	F_{rh}=100 mm/min
Rapid rate	F_{bh}=4000 mm/min
Radial depth of cut	a=2 mm
TEA max value	θ_{max}=40°
The clamping of the piece	Shown on Figure 14
The operation	Contour milling

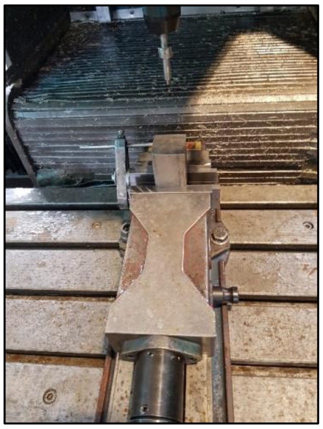


Figure 14. Clamping of work piece

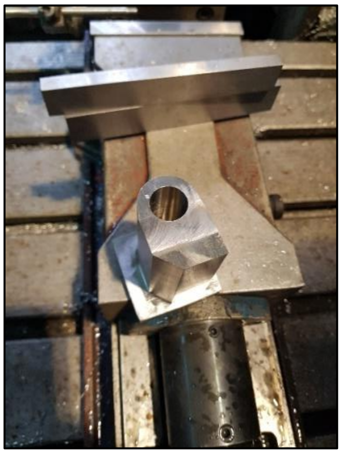


Figure 15. Finished part

By executing the NC program where the path of the tool was obtained by executing a previously defined algorithm implemented through several programming codes of the Delphi language, the machined contour was obtained, which is shown in Figure 15. It should be noted that the central hole was not the subject of this experiment, but was done due to the test of the drilling tool. By measuring the overall dimensions of the processed piece, it was determined that all dimensions except the width of the piece of 30 mm are within the tolerances of the free measurements, and that the measures of 30 mm deviates by +0.2 from the tolerance of the free measures, which may be a consequence of inadequately entered correction of the cutter diameter. Figure 16 shows (circled in red) the deviation from the contour defined by the drawing in the transition area (the contour is shown in Figure 13) from a straight line to a curvilinear contour, which is a consequence of the irregular arrangement of points in the map of points in this zone, so that the algorithm for placing the map points should be revised and eventually corected.

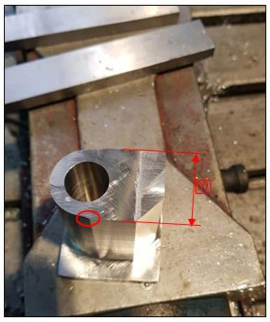


Figure 16. Deviations from measures and shapes

The surface roughness of the machined surface was not measured because it was not part of the experiment. The purpose of the experiment is to determine deviations in the dimensions and shape of the finished piece during processing with the NC program, basically generated by the application of the algorithm described in point 2 of this paper. The total machining time of the piece was 6 min and 15 s. It was concluded that it is necessary to perform another experiment with a corrected algorithm for setting the point map and measuring the surface roughness, comparing the measured values with those required on the drawing of the piece, in order to obtain more complete results of the application of the proposed tool path generation model in contour milling.

5. Conclusion

For the classical method of creating NC programs, qualified operators, complex and often expensive software CAM packages are needed, the time of creating the program is relatively long, and there is no guarantee that the time of creating the part will be the least, that is, that the production costs will be reduced to a minimum, because the effectiveness NC programs largely depend on the knowledge or experience of the programmer or operator. One of the alternative approaches, presented in this paper, is the complete autonomy of the NC code generation process during contour milling based on the existing or recognized work piece geometry.

The paper provides a detailed description of the algorithm for mapping the surface to be processed, which, based on experimental results, must undergo minor correction in the part when the length of the contour segment is less than the recommended radial depth of cut. Also, the cutting zone and the method of calculating the parameters are defined, before all angles of engagement in the cutting zone, and a detailed description of the genetic algorithm used to solve the optimization problem is given. Complete automation of the process is possible as a subject of further research, but it requires very complex computer programming, especially in the part related to the transformation of the coordinates of the optimized path into NC code, taking into account the variety of control units that can be found on modern CNC machines. Also, further research is needed in order to evaluate presented GA with other crossover and mutation operators to improve the speed of execution. The proposed GA was tested only for rough machining and having in mind that the presented algorithm for fine contour milling is the same, the further research will refer to testing and possible corrections for fine contour milling,

References:

Barclay, J., Dhokia, V., & Nassehi, A. (2015). Generating milling tool paths for prismatic parts using genetic programming. *Procedia CIRP, 33,* 490-495.

Bharath, S., & Natraj, K.R. (2018). Application of Artificial Intelligence Methods of Tool Path Optimization in CNC Machines. *International journal of engineering research and technology*.

Car, Z., Mikac T., Veza, I. (2004). Utilization of GA for optimization of tool path on 2D surface, Department of industrial engineering and management, *Faculty of Engineering, Rijeka University Vukovarska 58,* 51000 Rijeka, Croatia

Dumitrache, A., Borangiu, T., & Dogar, A. (2010). Automatic Generation of Milling Toolpaths with Tool Engagement Control for Complex Part Geometry. IFAC Proceedings Volumes, 43, 252-257.

Essink W. P., Nassehi, A., Newman, S.T. (2014). Toolpath Generation for CNC Milled Parts Using Genetic Algorithms, *Enabling Manufacturing Competitiveness and Economic Sustainability (pp.189-193),* doi:10.1007/978-3-319-02054-9_32

Jacso, A., Matyasi, G. & Szalay, T. The fast constant engagement offsetting method for generating milling tool paths. *International Journal of Advanced Manufacturing Technology 103*, 4293–4305 (2019). https://doi.org/10.1007/s00170-019-03834-8

Kumar, M., Khatak, P. (2018). Implementation of Genetic Algorithm in Multi-objective Optimization of Milling Toolpath, *International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162*, Vol.5, Issue 5, page no.1147-1155/, Available :http://www.jetir.org/papers/JETIR1805933.pdf

Kumar, S.P. (2017). Automation of tool path generation in multi-process micromachine tool for micromachining of prismatic and rotational parts. *Internatioanl Journal od Computer Integrated Manufacturing 31*, 49-70.

Makhanov, S. (1999). An application of variational grid generation techniques to the tool-path optimization of industrial milling robots, *Zh. Vychisl. Mat. Mat. Fiz*., Volume 39, Number 9, 1589–1600

Nassehi, A., Essink, W.P., & Barclay, J. (2015). Evolutionary algorithms for generation and optimization of tool paths. *Cirp Annals-manufacturing Technology, 64*, 455-458.

Oysu, C., Bingul, Z. (2009). Application of heuristic and hybrid-GASA algorithms to tool-path optimization problem for minimizing airtime during machining, *Engineering Applications of Artificial Intelligence*, Volume 22, Issue 3, https://doi.org/10.1016/j.engappai.2008.10.005.

Padmavathi, K., Yadlapalli, P., (2017), Crossover Operators in Genetic Algorithms: A Review, *International Journal of Computer Applications*, DOI: 10.5120/ijca2017913370

Singiresu S. R., (2009) Engineering optimisation-theory and practice, Fourth edition, *John Wiley & Sons, Inc.*, Hoboken, New Jersey

Tanović Lj., Petrakov, Y. (2007) Teorija i simulacija procesa obrade, *Mašinski fakultet*, Beograd

Umbarkar, A.J., & Sheth, P.D., (2015) Crossover operators in genetic algorithms: A review, Department of Information Technology, Walchand College of Engineering, India, *ICTACT Journal on Soft Computing, Volume: 06, Issue: 01*

Wang, H., Jang, P., & Stori, J.A. (2005). A Metric-Based Approach to Two-Dimensional (2D) Tool-Path Optimization for High-Speed Machining. *Journal of Manufacturing Science and Engineering-transactions of The Asme, 127*, 33-48.

Predrag Mitić
University of Kragujevac
Faculty of Engineering,
Serbia
predrag2904@gmail.com

Marija Zahar Đorđević
Faculty of Engineering, University of Kragujevac
Kragujevac, Serbia
maja_199@yahoo.com

Vuk Petronijević
Police Department in Kragujevac,
Kragujevac, Serbia
vukpetronijevic@ymail.com

Nebojša Abadić
Metropoliten University,
Belgrade,
Serbia
nebojsa.abadic@metropolitan.ac.rs

Aleksandar Đorđević
Faculty of Engineering, University of Kragujevac
Kragujevac,
Serbia
adjordjevic@kg.ac.rs