THE GENERATION OF ROBOT EFFECTOR TRAJECTORY AVOIDING OBSTACLES

MARTIN KOMAK, MARIAN KRALIK, VLADIMIR JERZ

Slovak University of Technology in Bratislava,
Faculty of Mechanical Engineering,
Institute of manufacturing systems, environmental technology and quality management,
Bratislava, Slovak Republic
DOI: 10.17973/MMSJ.2018_06_201764

e-mail: martin.komak@stuba.sk

This paper discusses the problem of trajectory planning for robotic manipulators at minimal cost. The paper deals with problems of obstacle avoidance during the manipulation and technological activities of the industrial robot. A system has been designed to find a trajectory that is both collision-free and of minimal distance. The goal is to reduce workplace costs with the robot and thus increase the efficiency of the robotized production cell. The work is mainly focused on the optimization of trajectory by planning the shortest path between robot targets. Also, when a trajectory is generated, the dynamic effects on the industrial robot are taken into account. Task solving is universal and is designed for several types of robot kinematic structures. The task comes under off-line robot programming.

KEYWORDS
obstacle avoiding, trajectory planning, industrial robot, optimization, robot effector

1 INTRODUCTION

The appropriate application of robots to industrial processes is essential, and currently firms are under pressure to make production more efficient. Two problems must be solved when trajectories are planned. Firstly, a path must be found between targets. This requires a geometric solution resulting from the geometry of the workspace. The result will be a description of the trajectory in space - the robot effector path. There are many approaches and methods to plan the shortest paths in a space with obstacles. The second level of solution is to control the robot along this path. In this case, it is about generating a trajectory in terms of the dynamic effects on the robot. If optimal trajectory solutions from the point of view of minimizing electricity consumption are sought, it is necessary to plan the trajectory so that both levels of solution are taken into account simultaneously [Carbone 2015]

The methodology of optimizing robot trajectories began to emerge in the 1970s, but the greatest increase in solutions has been in the past 20 years. The main task remains to find an optimal and non-colliding trajectory from the start position to the target position of the robot effector. Another task in the case of multiple goals is to effectively determine their sequence. It is clear from the findings that the planning of a trajectory focuses on three optimization criteria: the minimum duration of the operation, minimal energy consumption and minimization of undesirable dynamic effects on the robot construction. [Ata 2007] These criteria are interconnected. For example, by decreasing the acceleration of its movement on a path, the dynamic effects on the construction of the industrial robot will be reduced; but it is also assumed that the time required to complete the planned path will increase. Movement efficiency can be increased by using good process models, which are always associated with the research.

To find the optimal trajectory it is necessary to create a path in the space in which the robot effector will move. A number of path-planning methods in 2D and 3D space can be found in specialist literature. Many of the methods that have been used spring from the development of computer games. These methods seek not only to find the desired path between points in the space but also to determine the path so that it is optimal in terms of distance. Path representation can be achieved in several ways. One of the simplest algorithms that solves this problem is called a *Bug Algorithm*. The principle is that the robot avoids the obstacle in a clockwise direction until no more obstruction is encountered. Another approach is the Road Map method, which maps collision-free interconnection of a space based on a one-dimensional curve system in the C-space, without evaluating these paths. [Carbone 2015] Another kind of planning algorithm is based on Voronoi diagrams, which are defined as a way to divide space into sections that have specified characteristics. [Voronoiov 2015] The most common method is the creation of a graph where the nodes represent the achieved positions in the space [Saha 2006, Henrich 1998]. One example is the Visibility Chart, which generates the edges between obstacle peaks and target positions; these allow the use, for example, of the Dijkstrom algorithm to obtain the minimum distance. The workspace may also be divided into a grid by using the Grid Lattice method and working from that. Other algorithms that can be used to solve the path search are Gradient Descent Path Optimizer, Shortcutting, Probabilistic road map – RRT and Randomly Exploring Randomized Trees, which has several variants: RRT*, RRTConnect, T-RRT, Constrained RRT, Kinodynamic RRT, Discrete RRT, DARRT and others. [Klingensmith 2013]

The projection of a non-colliding path is becoming the basis for determining the trajectory for the

robot effector. Several approaches for planning the trajectory can be found in specialist literature, and these play a major role in the overall design of a robotized workstation, as well as in the application of the robot. In one approach, a *Genetic algorithm (GA)* was created, which was used to find the optimal solution [Zha 2002, Tian 2004, Toyoda 2004]. This optimization method is based on a model based on functional analysis and dynamic planning. The trajectory is approximated and the whole trajectory represents polynomials. The "*linear time-invariant*" (LTI) system can be used to optimize energy consumption [Wang 2012]. The aim is to optimize the "*Point-to-Point*" (PTP) movement from the energy point of view, taking into account the movement time. In literature [Leng 1997], the problem is formulated as a *non-linear mathematical programmable model*. This method takes into account the dynamics of the robot, the singular configuration and the constraints on the non-colliding path. Innovative trajectory planning models also include the generation of trajectories using *neural networks* [Koleda 2012]. Many researchers use so-called *spline curves* [D.Luca 1991], which enable dynamic effects on the robot construction to be taken into account. We can also use the so-called "cost function" in order to determine the trajectory, which evaluates the generated trajectory [Chettibi 2004] according to monitored input parameters. The approaches that take into account dynamic limitations of robots can be found in the relevant literature [Fang 2012, Saramago 2000, Vaz 2004, Saramago 1998, Valero 2006, Saramago 1999].

When we addressed the minimization of the time that the robot needs to perform a programmed operation, we focused on two ways. The aim was to find the shortest path between the objectives of the robot and design it in such a way that it can move at maximum speed.

2 PROBLEM STATEMENT

Let us consider a *n*-axes industrial robot. Each axis has its articulated movement restrictions, limitation of speed, torque, acceleration and jerk, which is the extent of acceleration in time [Tian 2004]. Optimization consists of decreasing or increasing one or more of these parameters. At the same time, the trajectory that the robot passes through must be free of obstacles. To find a non-colliding trajectory, we must first define in some way an obstacle in the robot's workspace. Our solution is based on avoiding the obstacles that are in STL format (STereoLithography). STL files only describe the surface geometry of a three-dimensional object without any representation of the color, texture, or other attributes common to CAD models. They describe an unstructured surface consisting of a triangle network described by normal lines and cusps using a three-dimensional Cartesian coordinate system. [Wikipedia 2017] The system designed by us loads this triangle network and continues to work with it. The user includes the input data into the system through the dialog window. This window contains input values for the start and end positions of the robot in the space - they are represented by three coordinates (X,Y and Z) in the 3D space. In this manner we generate the first targets for the robot control system. Next, a security zone is defined to ensure that the trajectory is located at a set distance (offset) from the obstacle ; a trajectory smoothing zone is also defined. Finally, the parameters of the output images - height, width, scaling factor and the angle of rotation of the cross-sections are entered. From the set values a configuration file is generated with which the system continues to work.

3 ALGORITHM SOLUTION

In the first step of the solution, we load an STL file into the memory. This file describes an obstacle in the robot's workspace. From this file we will use mainly the coordinates of particular triangles. After loading the entire file, a model of the avoided object – obstacle is obtained. This model is constructed from a network of triangles - Fig. 5. In the next step, the system retrieves the starting and target positions of the robot from the configuration file. Between these positions we draw a straight line, thus obtaining the shortest path between these positions. An algorithm was created to determine the intersection between the triangles and the straight line to see if the straight line intersects at least one triangle from the loaded model of the object - obstacle. If no intersection occurs, this straight line is found to be the shortest path between positions. On the other hand, the shortest possible non-colliding path for the robot to avoid the model of the obstacle will be found. If the axis representing the shortest path between the starting and the target position passes through the object, it is called a cross-sectional axis. With the help of this cross-sectional axis, cross-sectional planes can be defined. These are planes that will intersect the object and rotate around the cross-sectional axis as shown in Fig. 1.

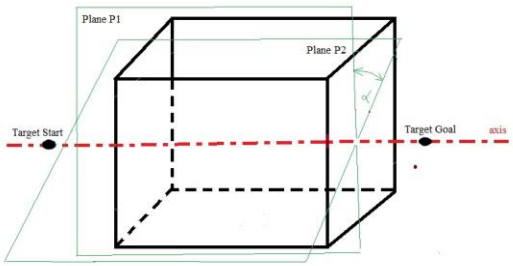

Figure 1. Imaging of cross-sectional planes

The angle of tilting of the cross-sectional planes α is entered by the user and determines the angle between the cross-sectional planes. These cross-sectional planes are generated from 0° to 359°. The generation of cross-sectional planes is as follows. It is known that each plane is defined by at least 3 points. All cross-sectional planes will contain the starting and ending positions. We transform the starting point by the T1 transformation to the beginning of the coordinate system and shift the target position according to this transformation. A third point in the space will be chosen. The first cross-sectional plane is given by the cross-sectional axis and the third point in the space. The third point is rotated according to the rotation matrix around the cross-section axis by the angle α and thus we obtain the other cross-sectional planes. The points generated by rotation of the third point lie on an imaginary circle whose center is on the cross-sectional axis, and the vector passing from the center of the circle through that point is perpendicular to the cross-sectional axis. The points are then converted back to the space by the inverse transformation to T1. In this way we obtain n cross-section planes (n = 359°/α), which are needed to generate trajectories.

First of all, the normal line is calculated from these three points determining the cross-sectional plane, and a general formulation of the cross-sectional plane is obtained. After that, all sides of the triangles of the obstacle model are replaced by lines and we express them parametrically of three equations (x, y, z). These expressions will be inserted into the general cross-sectional equation to obtain the t parameter by which the line crosses the plane. Now we have one equation of one unknown. We also get t parameters for the marginal points of abscissas representing the sides of the triangles. If the parameter of the intersection between the line and the plane lies between the parameters of the marginal points, it follows that the point of intersection is in the triangle area and we have the exact point of intersection. Otherwise, the triangle is outside the cross-sectional plane. The generated points of intersection represent the corner points of an object's contour in a given cross-sectional plane. Trajectories can be generated around this contour. Two trajectories can be created from each cross-sectional plane as shown below.

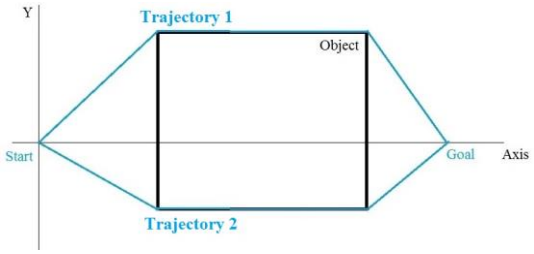

Figure 2. Trajectories with object in the cross-sectional plane

The figure shows the cross-section of the object and the two generated trajectories around the object in the cross-sectional plane. In order to obtain these trajectories, we proceed as follows: First, we express the cross-sectional plane by the general equation of the plane. By inserting the lines representing the sides of the triangles into this equation, we will obtain the positions in which the intersection occurs. After verifying all straight lines from all triangles from the network, we will gain the intersection positions in the given cross-sectional plane. By connecting these points, we will obtain a curve that represents the boundary of an object in a given cross-sectional plane. Now we want to determine how to avoid these boundaries of the object.

We can generate the straight lines $s1$, $s2$, ..., sn that pass through the starting position and all the points of the intersection - Fig. 3-1. All straight lines will be expressed parametrically. Thus we will obtain the straight line with the highest parameter to generate the upper path $s1$, or with the lowest parameter to generate the lower path $s4$. This straight line, as far as the point of intersection, represents the first part of the track. The next procedure will be the same, only at the generating of straight lines we will start from the point of intersection in which the track has ended - Fig. 3-2. Thus we will go around the whole object clockwise or counterclockwise in the cross-sectional plane - Fig. 3-3.

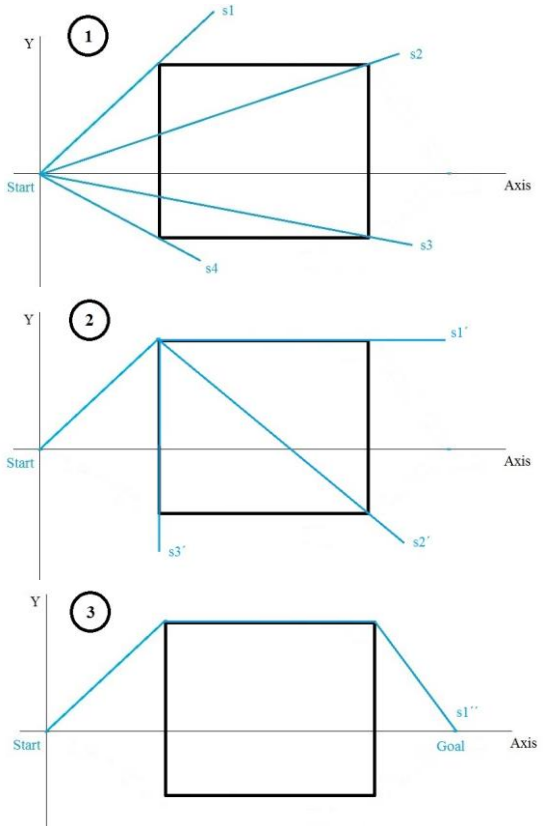

Figure 3. The sequence of steps in path generation

This procedure is applied in all cross-sectional planes. The paths generated from cross-section planes can then be ordered by their length to obtain the shortest possible path with the transition points that represent the calculated targets. These targets are the output data for generating a collision-free trajectory.

4 THE GENERATION OF "SMOTHED" PATH

To make the robot move more smoothly on the track, we can insert into the program the so-called zones. The zone is an imaginary ball around the generated target. After the entry of TCP (Tool Center Point) effector into the interior of the zone, the robot control system evaluates that the robot has reached the target and will move along the arc path to the next target of the programmed trajectory. As a result, the robot does not have to come in and stop at a precise position, but it is enough to get the robot to come near the target position at a certain speed. Thus the robot can continue to the next position without having to stop and perform jerky motions in slowing and starting. In order to avoid the obstacle in the area with the zones, the generated targets cannot match the object and thus have to be deflected by a particular vector according to Fig. 4.

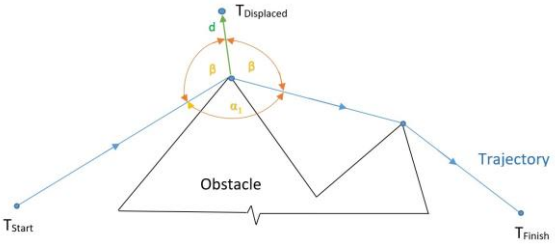

Figure 4. Deflection the transition target

The length and direction of the deflection vector *d* are calculated from the following relationship:

$$d = z - k_\beta + z_{Safety} \quad (1)$$

where
z are zone data – the size of the generated corner path,
z_{Safety} – a safety zone, can replace the offset of the trajectory with the real size of the effector,
k_β is the distance coefficient which can be calculated as follows:

$$k_\beta = z - \frac{z}{135} * (\beta - 45) \quad (2)$$

where β is the angle between the vector and the trajectory, and can be calculated as:

$$\beta = \frac{360 - \alpha_1}{2} \quad (3)$$

α_1 – is the angle between two consecutive abscissa of the trajectory.

It follows from the relations that if α_1 approaches the 180° angle, the trajectory segments are parallel and the length of the deflection vector is minimal. Otherwise, if the angle is minimal, the part of the object that is avoided by the trajectory is very sharp and therefore the length of the deflection vector adjusts according to the size of the angles and the zone. In the case of a defined zone, we will move all the transition targets to get all the targets from the resulting trajectory.

5 EXPERIMENTAL RESULTS

This software solution was verified by an experiment. As an obstacle, we used a water turbine - Fig. 5. Initial coordinates were (-80, 0, 30) and the target coordinates (80, 5, 40). The angle between the cross-section planes was 5°, so 71 cross-sectional planes were generated. From each plane, we obtained one trajectory. We have sorted them in ascending order according to the length. The value of the security zone was set to zero and the value of the zone was set to 10 mm. In Fig. 6, we can see the cross section of the turbine by the plane at the shortest trajectory. This plane contains 4 transition points. The length of this trajectory is 166.517 mm.

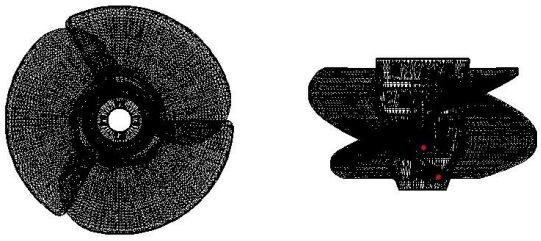

Figure 5. The avoided turbine - STL format

The coordinates of the transition targets from the shortest trajectory are in Tab. 1.

Targets	Coordinates		
	X	Y	Z
Initial	-80	0	30
Transition 1	-11.6337	10.2679	43.4541
Transition 2	-1.75313	11.1293	44.6955
Transition 3	-0.658056	11.2962	44.9138
Transition 4	0.507731	11.2326	44.8738
End	80	5	40

Table 1. The coordinates of the targets on the shortest path

Fig. 6 shows a cross-sectional plane with the shortest trajectory. In Fig. 7 is a cross-sectional plane with

the longest trajectory. The length of the longest trajectory is 179.118 mm and it contains 19 transition points.

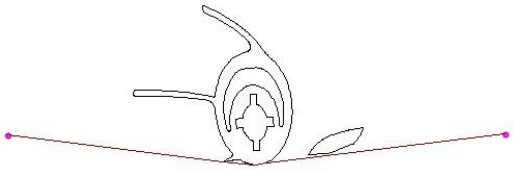

Figure 6. The view of the shortest trajectory in the cross-sectional plane (*XY' plane, α =165°*)

The calculated coordinates of the targets in Table 1. and in the Fig. 6. and Fig. 7. are based on the origin of the coordinate system, which can also be referred to as a global coordinate system (in this case it is under the turbine on its axis of rotation). The *XY'* plane denotes a cross-sectional plane rotated by an angle *α* from the *XY* plane.

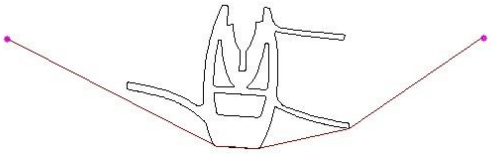

Figure 7. The view of the longest trajectory in the cross-sectional plane (*XY'* plane, *α =45°*)

The resulting coordinates of the transition points can be written into the language understood by the industrial robot control unit according to the certain syntax - Fig. 8. This generated code can be directly used in a real application, allowing the visual verification of this solution.

```
MODULE Module1
    CONST robtarget Target_10:=[[-80,0,30],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[-11.6337,10.2679,43.4541],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_30:=[[-1.75313,11.1293,44.6955],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40:=[[-0.658056,11.2962,44.9138],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_50:=[[0.507731,11.2326,44.8738],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60:=[[80,5,40],[0,0,1,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    PROC main()
        MoveL Target_10,v1000,z10,tool0;
        MoveL Target_20,v1000,z10,tool0;
        MoveL Target_30,v1000,z10,tool0;
        MoveL Target_40,v1000,z10,tool0;
        MoveL Target_50,v1000,z10,tool0;
        MoveL Target_60,v1000,z10,tool0;
    ENDPROC
ENDMODULE
```

Figure 8. Program code generated for the ABB industrial robot

Finding a solution is not time-consuming and is cost-effective. The generation of a trajectory around an object is possible in a few seconds. The algorithms are not demanding for computational memory and the system is not complicated graphically. All results are in the form of images, as well as text files, which can be used in software applications for off-line programming of industrial robots. This software solution can be visualized as shown in Fig. 9.

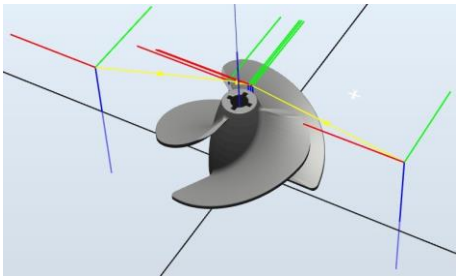

Figure 9. Trajectory visualization in software RobotStudio from company ABB

6 CONCLUSIONS

The article provides an overview of how to generate trajectories. It presents the algorithms to generate a robot effector trajectory to avoid obstacles. The solution described is to accelerate the movement along the trajectory by setting zones around the targets. The aim was to design a collision-free trajectory based on the cross-sectional plane of the object. Specific algorithms enable solutions to be found with minimal computational and time demands. In practice, the algorithms mainly work with vector calculations, equations of straight lines and planes and their intersections. Another subject for research may be to take into account the real dimensions of the effector and its orientation in space.

ACKNOWLEDGMENTS

This study was supported by the Cultural and Educational Agency of the Ministry of Education of the Slovak Republic under the contract KEGA 035STU-4/2017 The introduction of progressive educational methods for manufacturing systems to car production.

REFERENCES

Paper in a journal:

[Ata 2007] Ata, A.A. Optimal trajectory planning of manipulators: A review. *Journal of Engineering Science and Technology Vol. 2, No. 1*. 2007 pp 32-54.

[Carbone 2015] Carbone, G. and Gomez-Bravo, F. Motion and Operation Planning of Robotic Systems. Background and Practical Approaches. January 2015. ISBN 978-3-319-14705-5

[D.Luca 1991] D.Luca, A. et al. A Sensitivity Approach to Optimal Spline Robot Trajectories. Automatica, Vol. 27, No. 3. 1991 pp 535-539.

[Fang 2012] Fang, H.C. et al. Interactive robot trajectory planning and simulation using Augmented Reality. Robotics and Computer-Integrated Manufacturing 28. 2012. pp 227 – 237.

[Henrich 1998] Henrich, D., et al. Multi-directional search with goal switching for robot path planning. Computer Science Department, University of Karfsruhe

[Chettibi 2004] Chettibi, T. et al. Minimum cost trajectory planning for industrial robots. European Journal of Mechanics A/Solids 23. 2004 pp 703–715.

[Koleda 2012] Koleda, P. and Nascak, L. Generate trajectory using a neural network (in Slovak: Generovanie trajektórie pomocou neuronovej siete). ACTA FACULTATIS TECHNICAE, XVII. 2012 pp 43–49.

[Leng 1997] Leng, D.Y. and Chen, M. Robot Trajectory Planning using Simulation. Robotics & Computer-Integrated Manufacturing, Vol. 13, No. 2, 1997. pp 121-129.

[Martin 1997] Martin, B.J. and Bobrow, J.E. Minimum effort motions for open chain manipulators with task-dependent end-effector constraints. In: Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on , vol.3, no. 20-25 Apr 1997, pp.2044-2049.

[Saha 2006] Saha, M., et al. Planning Tours of Robotic Arms Among Partitioned Goals.

[Saramago 1998] Saramago, S.F.P. and Steffen Jr., V. Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system. Mech. Mach. Theory Vol. 33, No. 7. 1998 pp 883 – 894.

[Saramago 1999] Saramago, S.F.P. and Steffen Jr., V.: Dynamic Optimization for the Trajectory Planning of Robot Manipulators in the Presence of Obstacles. Journal of the Brazilian Society of Mechanical Sciences. 1999

[Saramago 2000] Saramago, S.F.P. and Steffen Junior, V. Optimal trajectory planning of robot manipulators in the presence of moving obstacles. Mechanism and Machine Theory 35. 2000 pp 1079 – 1094.

[Tian 2004] Tian, L. and Collins, C. An effective robot trajectory planning method using a genetic algorithm. Mechatronics 14. 2004. pp 455–470.

[Toyoda 2004] Toyoda, Y. and Yano, F. Optimalizing Movement of A multi-Joint Robot Arm with Existence of Obstacles Using Multi-Purpose Genetic Algorithm. EMS Vol.3. No.1. 2004. pp 78-84.

[Valero 2006] Valero, F. et al. Trajectory planning in workspaces with obstacles taking into account the dynamic robot behaviour. Mechanism and Machine Theory 41. 2006 pp 525–536.

[Vaz 2004] Vaz, A.I.F. et al. Robot trajectory planning with semi-infinite programming. European Journal of Operational Research *153. 2004* pp 607–617.

[Vitralab 2011] Automation and robotics. Guide. (in Slovak: Automatizacna a roboticka technika. Prirucka.) VITRALAB. Kosice: 2011

[Voronoiov 2015] Voronoiov diagram. http://mathworld.wolfram.com/VoronoiDiagram.html [2.11.2015]

[Wang 2012] Wang, X. et al. Energy Optimal Point-to-Point Motion Using Model Predictive Control. Annual Dynamic Systems and Control Conference. Florida: 2012

[Zha 2002] Zha, X.F. Optimal pose trajectory planning for robot manipulators.

WWW page:

[Klingensmith 2013] Klingensmith, M. Overview of Motion Planning. 07.05.2017 http://www.gamasutra.com/

[Wikipedia 2017] Stereolithography. https://en.wikipedia.org/wiki/Stereolithography

CONTACTS:

Ing. Martin Komak, PhD.
doc. Ing. Marian Kralik, CSc
doc. Ing. Vladimir Jerz, CSc.
Slovak University of Technology in Bratislava,
Faculty of Mechanical Engineering,
Institute of manufacturing systems, environmental technology and quality management,
Namestie slobody 17, 812 31 Bratislava 1, Slovak Republic,
martin.komak@stuba.sk; Tel.: +421 908 099 406
marian.kralik@stuba.sk; Tel.: +421 (2) 57 296 579
vladimir.jerz@stuba.sk; Tel.: +421 (2) 57 296 554