International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:10, No:12, 2016

Retraction Free Motion Approach and Its Application in Automated Robotic Edge Finishing and Inspection Processes

M. Nemer, E. I. Konukseven

***Abstract*—**In this paper, a motion generation algorithm for a six Degrees of Freedom (DoF) robotic hand in a static environment is presented. The purpose of developing this method is to be used in the path generation of the end-effector for edge finishing and inspection processes by utilizing the CAD model of the considered workpiece. Nonetheless, the proposed algorithm may be extended to be applicable for other similar manufacturing processes. A software package programmed in the application programming interface (API) of SolidWorks generates tool path data for the robot. The proposed method significantly simplifies the given problem, resulting in a reduction in the CPU time needed to generate the path, and offers an efficient overall solution. The ABB IRB2000 robot is chosen for executing the generated tool path.

***Keywords*—**Offline programming, CAD-based tools, edge deburring, edge scanning, path generation.

I. Introduction

DEBURRING processes have been identified as the bottleneck in many machine industries. The burr removal methods can induce dimensioning errors to the workpiece if improperly executed. Burrs are caused by many machining processes including milling, drilling, turning, and broaching. An amazing transformation of edge finishing has occurred over the past 50 years trying to mimic the adaptive nature of human intelligence in order to replace manual deburring. Anthropomorphous robots are the best state-of-the-art compromise between performance and flexibility for automated deburring tasks [2]. They provide larger work volumes, safety and efficiency at a lower cost than CNC machines.

Several methodologies for robotic deburring have been actually been proposed in the literature. Some are based on servo control and compliance control to stabilize the deburring condition [3]–[6]. These methods focused not on the path generation itself but on the improvement of the tool path. As for the off-line path generation, while most of the robots are controlled under a teaching-playback mode, some research has been done on generating a tool path based on the contour of a workpiece, such as teaching method [7], computer vision [8], CAD/CAM approach [2], [9], [6]. However, these researches either focus on planar workpieces or include some user input and feedback in order to achieve an overall collision-free path. The main difficulty in this subject is the generation of collision-free motions that are able to move the end-effector from one edge to another on the same workpiece.

There are many different known algorithms in the literature [10] to yield an efficient collision-free path. One popular approach for the case of robotic manipulators in static environments is the probabilistic roadmaps planner [11], which is a probabilistically complete algorithm. Khatib [12] offered a very interesting algorithm known as an artificial potential field approach; however, the main drawback of this method is the possibility of sticking in local minima; that is, the algorithm is not necessarily complete. Another algorithm inspired by the Khatib is the artificial force field approach. Many papers discuss slightly different models of such an algorithm, and [13] is one example; nonetheless, the main idea is that obstacles apply forces on the robot links in such a way that they repel the robot dynamically. Unfortunately, this method has the possibility of sticking in local minima, similar to the artificial potential field method. On the other hand, due to the nature of the problem in hand, a novel but simple approach is going to be discussed in this study.

In this paper, there are two main objectives. The first is the introduction of a simple yet effective path generation algorithm that tries to overcome the difficulty and complexity of finding a collision-free path between two edges on the same workpiece. The second is a full implementation of the algorithm in order to perform an off-line path planning by utilizing the CAD model of the considered workpiece; where such a plan will generate a nominal collision-free path for the end-effector to go over the desired edges to be deburred.

The paper is organized as follows. First is Section II in which a Retraction-Free-Approach (RFA) motion algorithm is introduced. In Section III, a fully connected path for performing edge deburring and/or inspection is generated by the use of RFA motion. In Section IV, the experimental setup and experimental results are presented, while the conclusions are drawn in Section V.

E. I. Konukseven is with Mechanical Engineering Department, Middle East Technical University, 06800, Ankara, Turkey (corresponding author to provide phone: +90-533-2331806; fax: +90-533-2102536; e-mail: konuk@metu.edu.tr).

M. Nemer was with Mechanical Engineering Department, Middle East Technical University, 06800, Ankara, Turkey (e-mail: mahmoudhnimer@gmail.com).

II. RFA Motion

This approach is going to be used to generate non-processing motions. Non-Processing motion is a motion of which is needed to change the end-effector position after processing an edge to go to another one. Such motion does not include any actual processing (deburring or scanning), and therefore, it is not bounded to a specific trajectory, as far as the trajectory is collision-free.

A. Calculating RFA Motions

One may take an advantage from the nature of the working space in hand. Notice that if the manipulator assumes an upper arm configuration throughout its motion, there will not be any collision between the part and the robotic hand as long as the end-effector is above the bounding box of the workpiece. There are eight vertices of such a box, the ones adjacent to the top plane share an elevation of $z = z_0$. Adding half the width of the tool holder, it can be guaranteed that the manipulator will not collide with the part as long as the end-effector's elevation (z-axis component of the location of the end-effector) is above $z = z_0 + 0.5w_6$, where w_6 is the maximum width of the tool holder. Therefore, a collision-free motion can be generated by finding a retraction motion that takes the end-effector from the starting position to free space and an approach motion to place the end-effector from free space to the goal position.

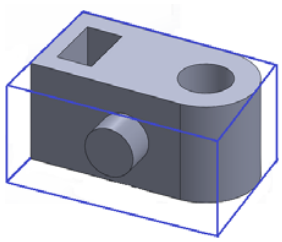

Fig. 1 The bounding box of an arbitrary part

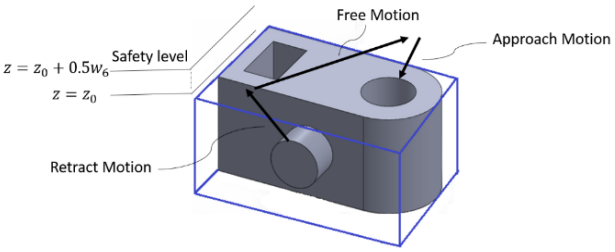

Fig. 2 Outline representation of the non-processing motion generation algorithm

Notice that the starting and ending positions of the non-processing motions are actually also the starting positions of the processing motions. From the reachability test it is guaranteed that there are not any collisions at these positions.

There are two scenarios when the end-effector's location is under the safety level. First, only the last link is below $z = z_0 + 0.5w_6$. The second scenario is to have the last link to be fully immersed and a portion of the fifth link be below $z = z_0 + 0.5w_6$.

For the first case, since the manipulator has six DoF, it is possible to retract the end-effector with a constant orientation along $\vec{u}_3^{(6)}$ from the start position to the safety level with a guarantee that this motion is collision-free. Mathematically, the orientation of the end-effector remains constant through the motion, $C_1 = C_2 = C^{(0,6)}$. While the location of the end position is $\bar{r}_2^{(0)} = \bar{r}_1^{(0)} - sC^{(0,6)}\bar{u}_3$. s is found using simple trigonometry as follows:

$$s = \frac{z_0 + 0.5w_6 - r_{1,3}}{\cos\beta} \tag{1}$$

where

$$\cos\beta = \frac{r_{1,3} - w_{1,3}}{d_6} \tag{2}$$

$r_{1,3}$ is the third component of $\bar{r}_1$ and $w_{1,3}$ is the third component of $\bar{w}_1$. holder.

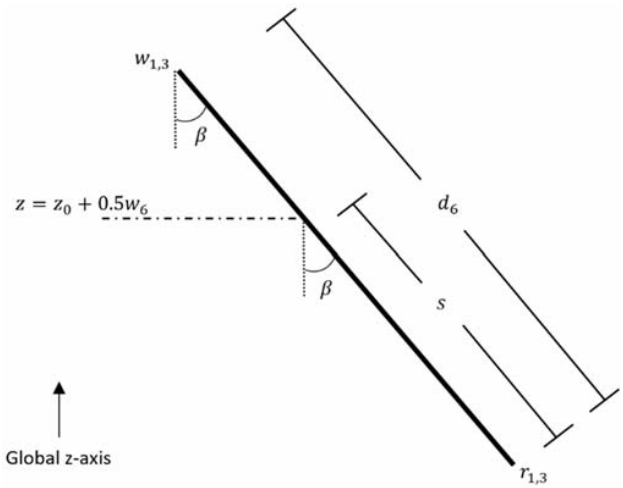

Fig. 3 Trigonometry used to find the value of s

For the second case, this motion can be divided into two segments. First one moves the end-effector from the start position to the wrist point with an orientation similar to the initial orientation of the fifth link. A collision test is to be performed at this case to make sure it is a safe motion. As for the second segment, the end-effector moves with a constant orientation along the new $\vec{u}_3^{(6)}$ until it reaches safety level.

Mathematically, the end position of the first segment motion can be defined by the initial orientation of the fifth link, $C_2 = C^{(0,4)}$. While the location is equal to the initial wrist point, $\bar{r}_2^{(0)} = \bar{r}_1^{(0)} - d_6C^{(0,6)}\bar{u}_3$. Consequently, for the second segment, the start position is basically the end position of the first segment. The end position, on the other hand, consists of the same orientation as with the start position, $C_3 = C_2$, and the location point $\bar{r}_3^{(0)} = \bar{r}_2^{(0)} - sC^{(0,6)}\bar{u}_3$. s is found from (1) and (2).

It can be realized that the approach motion for a given

position can be generated by simply reversing its retraction motion.

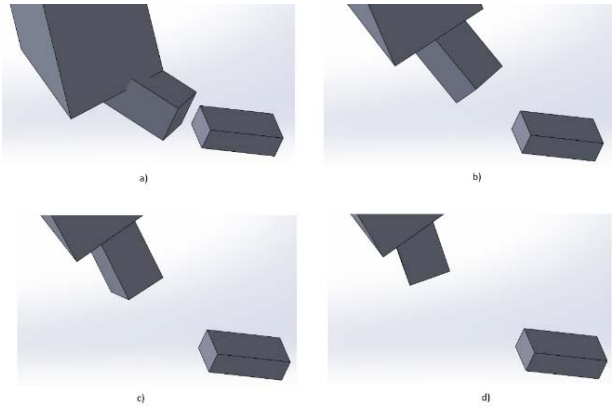

Fig. 4 Different snap shots of the first segment of the retraction motion from a critical position where both last two links are under the safety level

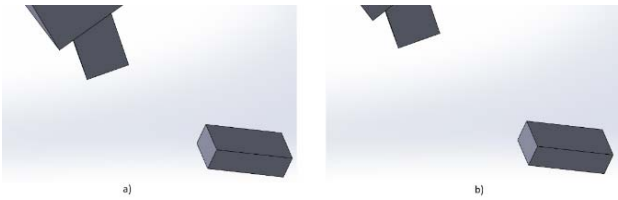

Fig. 5 Outline representation of the non-processing motion generation algorithm

Therefore, given a starting and an ending positions to generate a non-processing motion between them, all is needed is to first generate the retraction and approach motion for the corresponding positions. Then, connect these two motions with a straight line motion in free-space. Such algorithm will be referred to as the RFA Motion. This approach significantly reduces the complexity that most collision-free path generation methods suffer from. The one disadvantage is that this approach might overlook some edges, if found unreachable by the robot manipulator.

The RFA motion may not be the shortest possible collision-free path between two given positions. Nonetheless, it offers a very efficient solution regarding CPU time and path length.

B. Collision Detection Test

The manipulator's links are modeled as cuboids, as shown in Fig. 6. Such a model is conservative, which can accommodate for some deviations. Also, it is faster to maneuver cuboids in SolidWorks which will require fewer memory and time compared to using an actual geometric model.

The non-processing motions generated between each two positions need to be collision-free. Only the retraction and approach parts of the RFA motion need to be checked. Since the portion of the RFA motion made in free space is by definition free of collisions.

It is critical to realize that both the approach and retract motions for a given critical position is in fact the same motion but reversed in the order of execution. Therefore, the collision detection test boils down to checking the retraction motion of each critical position separately.

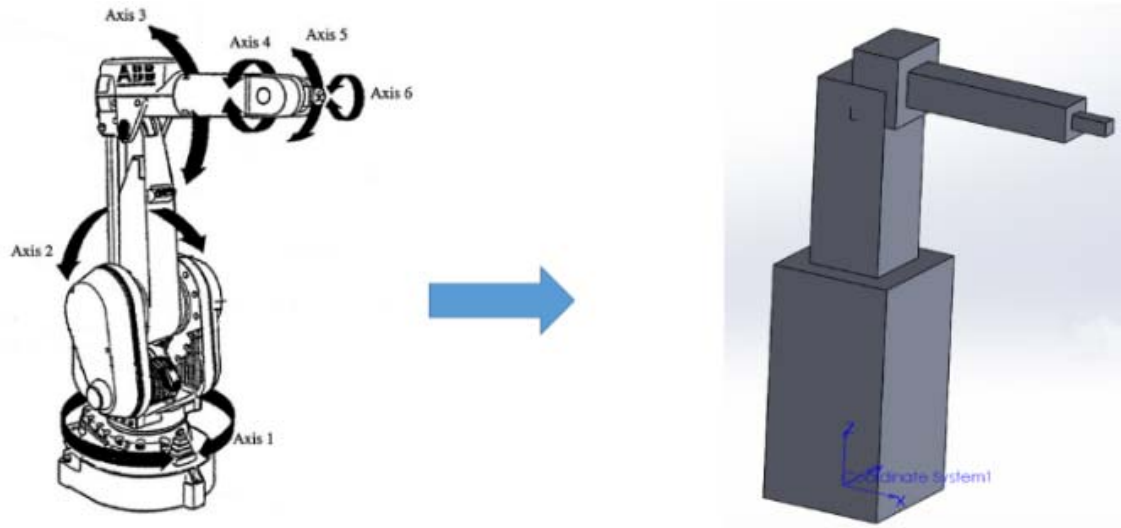

Fig. 6 Geometric model of the robotic arm used for performing the collision detection test

The test nature is similar with the one performed on processing motions. That is, the motion is discretized into a number of poses that in turn covers the whole range of motion. Figs. 4 and 5 show an example of such discretization of a retraction motion.

While checking for collisions, the program will also be monitoring the joints' angles. If an angle exceeds its physical limitation, then such motion cannot be performed by the robot.

III. RFA Motion's Application in Edge Finishing and Inspection

The RFA motion algorithm is going to be used to generate

the non-processing motions type in the overall nominal tool path of an edge finishing process.

The overall path of the end-effector is going to be generated in order to be later executed by the robotic manipulator. Starting with the preparation stage, the edges needed to be scanned or deburred are selected from the CAD model. Next, for each selected edge the corresponding needed motion of the end-effector in order to process that specific edge is generated, such motions will be referred to as processing motions. Then, all possible combination of motions that connect these separated processing motions together are generated using RFA algorithm, such motions will be referred to as non-processing motions. After that, a search algorithm is applied on the constructed graph to select a subset of the non-processing motions.

A. Processing Motion Generation

Each process motion corresponds to one edge on the part; therefore, these motions are fixed. Two types of edges are considered in this study, namely, Straight edges and Circular ones.

In case of a straight line, these motions can be found by taking the desired offset value and direction (approach angle) from both ending vertices of that edge. The offset value is the relative distance between the edge and end-effector during the processing motion, such parameter is numerically entered by the user. While the offset direction can be chosen as either the average of the adjacent faces' normals, check Fig. 7 (a), or normal to the direction of the formed burr, Fig. 7 (b). Generally, the burr forms in the direction normal to both the normal vector of the later machined face between the two adjacent faces and the direction of the edge itself [10].

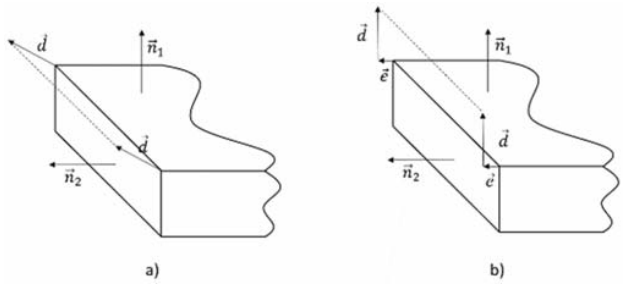

Fig. 7 (a) Illustration of an offset direction, $\vec{d}$, that is equal to the average of the two normals of the adjacent faces of the edge, $\vec{n}_1$ and $\vec{n}_2$ (b) Illustration of an offset direction, $\vec{d}$, in the normal direction to the formed burr

As for the tip point orientation, it is found as follows, the opposite direction of the offset vector, $\vec{d}$, is taken to be the z-axis of the end-effector. The y-axis is oriented such that it is parallel to the edge, the sign of the direction is decided such that the x-axis will be pointing downwards with respect to the global coordinates, using the right-hand rule. Fig. 8 gives a good example of such calculation.

In case of an arc edge, the corresponding processing motion can be sufficiently defined by the start and end positions of the end-effector along with a middle position. A similar approach with the straight line edge can also be applied to calculate the processing motion for an arc edge.

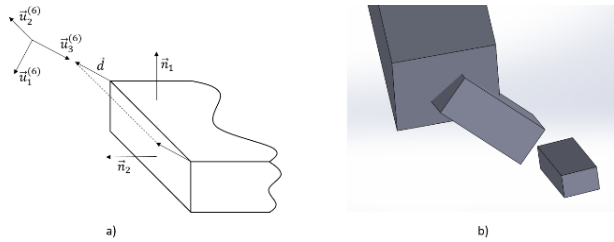

Fig. 8 (a) Calculating the orientation of the end-effector of the robotic arm (b) Simulation representation

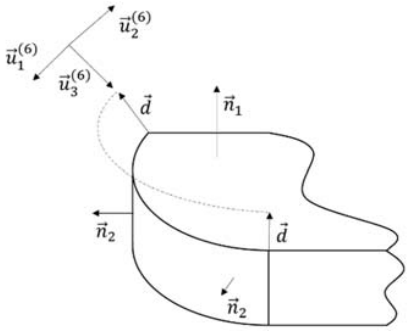

Fig. 9 Calculating the orientation of the end-effector of the robotic arm in the case of an arc edge

Collision detection test is performed on each of these calculated motions. The test nature is similar with the one performed on RFA motions. Note that such motions are necessary to perform the task, either scanning or deburring. Hence, if any of these motions turns to have a collision it means that the corresponding edge cannot be processed as far as the part in hand is positioned in the given orientation in space.

B. Non-Processing Motion Generation

The non-processing motion between two positions is generated by the RFA algorithm. However, two special cases are treated differently for better results.

The first case is when both start and end positions share the same vertex, check Fig. 10. The motion is a direct one (straight line in the configuration space). Such motion is collision-free since the shared vertex is reachable.

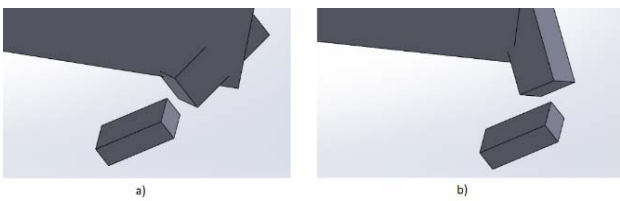

Fig. 10 Illustration of the non-processing motion when both start and end positions point at the same vertex

The second case is when both positions share a reachable

edge. A direct motion can be made by utilizing the previously generated processing motion of the shared edge, see Fig. 11. This motion is also collision-free, since it is constructed of smaller collision-free motions calculation.

For generating the non-processing motion between two given positions, the algorithm will first check if these two positions share a vertex or an edge, if so the method explained above is used to generate the path of the end-effector. Otherwise, an RFA motion is generated.

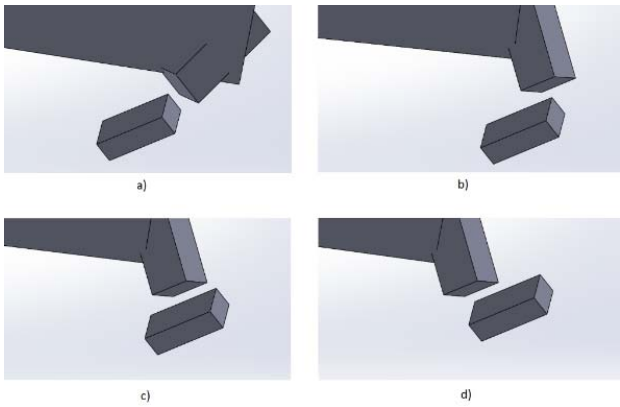

Fig. 11 Illustration of the non-processing motion when both the start and end positions share a reachable edge

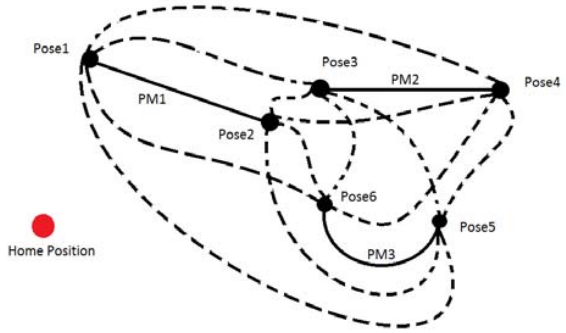

Fig. 12 An example of three processing-motions (PM) and the representing graph for finding route solution of the modified TSP "The non-processing motions between the home position and other poses are not shown for simplicity"

This approach reduces the complexity of finding all possible non-processing motions from $O(n^2)$ to $O(n)$. Instead of generating a motion between each pair of critical positions, the problem is reduced to calculate one retraction motion for each critical position space.

C. Overall Path Planning

After generating all possible non-processing motions, the next step is to select a subset of them that creates a connected path together with the processing motions. In fact, such problem is a TSP (Traveler Salesman Problem).

To further understand this representation, refer to Fig. 12. Notice that each node in the graph represents a starting or an ending position of the end-effector, while the connected edges represent the processing motions and finally the dashed ones is for the non-processing motions. Unfortunately, general TSP is known to be an NP-complete problem. There is not a systematic approach to reach the optimum solution in a polynomial time algorithm.

In this study, the nearest neighbor algorithm is adopted and implemented in the software. The solution of the TSP will correspond to a collision-free motion trajectory which represents the overall path to be executed by the robotic manipulator.

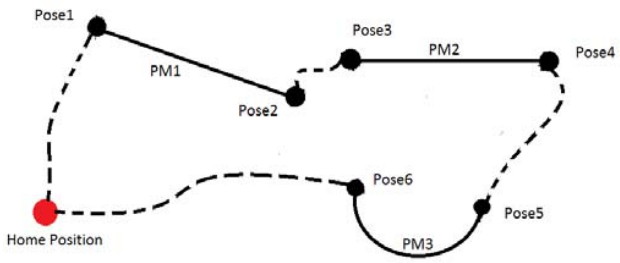

Fig. 13 The solution of the TSP graph given in Fig. 12 using nearest neighbor algorithm

IV. Experimental Results

In this section two sample workpieces are presented to test the practical functionality of the study. The test setup consists of the ABB IRB2000, working table and the workpiece to be processed. In addition, for the two last parts, a spindle holder and a flex attachment spindle (Dremel 225) are used, see Fig. 14. As for the computer connected to ABB IRB200 it has an Intel® Core™ i7-4700HQ CPU processor working at 2.4 GHz.

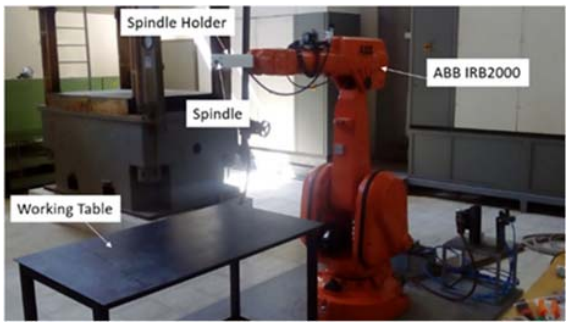

Fig. 14 Test Setup Illustration

A. Sample Part 1

This workpiece is an arbitrary part that was manufactured for the sole purpose of testing and confirming the overall algorithm of the study, see Fig. 15. The bounding box of the part is 125mmx95mmx70mm.

To further understand this representation, refer to Fig. 15. Notice that each node in the graph represents a starting or an ending position of the end-effector, while the connected edges represent the processing motions, and finally the dashed ones is for the non-processing motions. Unfortunately, general TSP is known to be an NP-complete problem. There is not a systematic approach to reach the optimum solution in a polynomial time

algorithm.

The objective is to generate a path for the end-effector in order to scan the edges of the given part. The part is not directly mounted on the working table. Instead, a smaller piece, acts like a base, is mounted on the working table and then the part is placed on top of the piece. This is done just to give the robot extra freedom to reach more edges.

The part has a total of 21 edges, all convex. Excluding the six lower edges, the software tries to generate a path to scan the remaining 15 edges. However, two of the edges turn out to be unreachable in the given orientation, shown in red in Fig. 16. Therefore, the overall generated path goes over 13 edges of the part. The order at which these edges are processed is given in Fig. 16. The software took 19 seconds to generate the output motion.

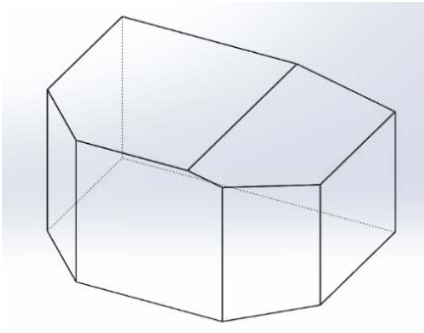

Fig. 15 Trimetric view of part 1

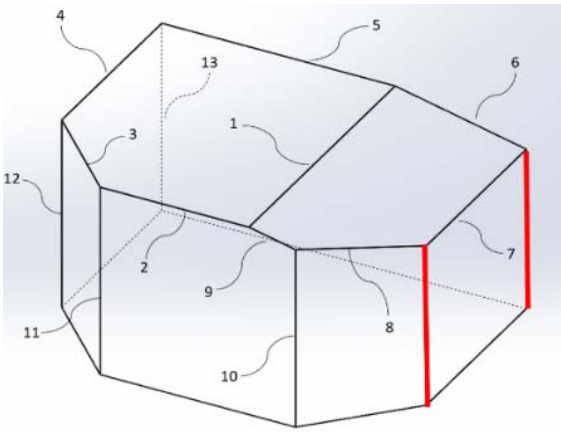

Fig. 16 The order of which the edges of part one are scanned

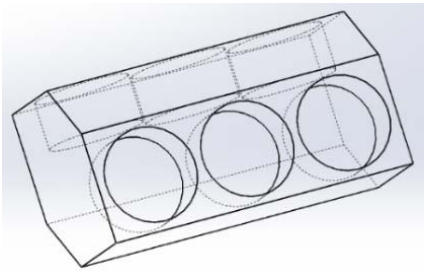

Fig. 17 Trimetric view of part 3

B. Sample Part 2

This part represents the general characteristics of a V6 engine block, see Fig. 17. The bounding box of the part is 200 mm x 150 mm x 100 mm motion.

The objective is to generate a path for the end-effector in order to deburr the convex edges of the given part. Since no actual deburring is taking place, the part is directly mounted in the working table without fixing it to a fixture.

The part has a total of 24 convex edges. Excluding the four lower edges, the software tries to generate a path to process the remaining 20 edges. Four of these edges turn to be unreachable in the given orientation by the end-effector, shown in red in Fig. 18. Therefore, the overall generated path goes over 16 edges. The order at which these edges are processed is given in Fig. 18. Where the time needed by the software to complete the path generation was 26 seconds.

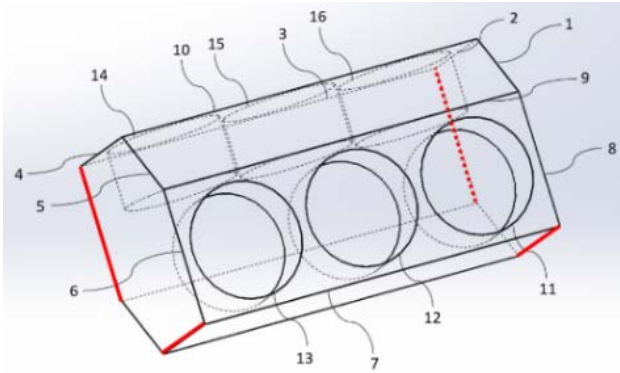

Fig. 18 The order of which the edges of part two are deburred

V. Conclusion

In this paper a new method for generating non-processing motions of a 6-DoF was introduced and applied in automated edge finishing and scanning processes. This objective was achieved by developing software in SolidWorks API using Visual Basic programming language. The ABB IRB2000 model was considered as the robotic manipulator throughout the study. The performance of the software and the proposed method were verified by running tests on two workpieces. The RFA motion approach reduced the complexity of both calculating a motion between two positions and the number of calculations needed for constructing all possible non-processing motions. In both runs the algorithm was able to generate a path in less than a minute for each workpiece, making it a fast off-line path generator algorithm for edge finishing and scanning. Compared with the classical point-to-point teaching method, which takes several hours to a day to perform, depending on the skill of the worker and the complexity of the piece. While other off-line path generation approaches using the CAD model of the considered workpiece, such as [1], need a processing time around three hours on average.

ACKNOWLEDGMENT

This study is supported by Scientific and Technological Research Council of Turkey (TÜBİTAK), 114E274.

REFERENCES

[1] F. Leali, M. Pellicciari and F. Pini, "An Offline Programming Method for the Robotic Deburring of Aerospace Components," *Robotics in Smart Manufacturing, Communications in Computer and Information Science*, vol. 371, pp. 1-13, 2013.

[2] O. Valente, "A New Approach for Tool Path Control in Robotic Deburring Operations," in *17th international Congress of Mechanical Engineering*, Sao Paulo, 2003.

[3] X. L. Liao, "Modeling and control of automated polishing/deburring process using a dual-purpose compliant tool head," *International Journal of Machine Tools & Manufacture*, pp. 1454-1464, 2008.

[4] H. Zhang, H. Chen, N. Xi, G. Zhang and J. He, "On-Line Path Generation for Robotic Deburring of Cast Aluminum Wheels," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, 2006.

[5] H. Song, K. Byeong-Sang and S. Jae-Bok, "Tool Path Generation based on Matching between Teaching Points and CAD Model for Robotic Deburring," in *The 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Kaohsiung, 2012.

[6] S. Lee, C. Li, D. Kim, J. Kyung and H. ChangSoo, "The Direct Teaching and Playback Method for Robotic Deburring System Using the Adaptiveforce-control," in *2009 IEEE International Symposium on Assembly and Manufacturing*, Suwon, 2009.

[7] L. Princely and S. T, "Vision Assisted Robotic Deburring of Edge Burrs in Cast Parts," *Procedia Engineering*, vol. 97, pp. 1906-1914, 2014.

[8] N. Asakawa, K. Toda and Y. Takeuchi, "Automation of chamfering by an industrial robot; for the case of hole on free-curved surface," *Robotics and Computer Integrated Manufacturing*, vol. 18, p. 379–385, 2002.

[9] S. Lavalle, *Planning Algorithms*, Cambridge: Cambridge University Press, 2006.

[10] L. E. Kavraki, P. Svestka, J.-C. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, p. 566–580, 1996.

[11] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, p. 90–98, 1986.

[12] P. Chotiprayanakul, D. K. Liu, D. Wang and D. G., "A 3-Dimensional Force Field Method for Robot Collision Avoidance in Complex Environments," in *24th International Symposium on Automation & Robotics in Construction*, Madras, 2007.

[13] H. B. J. Kazerooni and B. Kramer, "An Approach to Automated Deburring by Robot Manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 108, pp. 353-359, 1986.