Design of 3D Printer-Like Data Interface for a Robotic Removable Machining

Fusaomi Nagata[1(✉)], Shingo Yoshimoto[1], Kazuo Kiguchi[2], Keigo Watanabe[3], and Maki K. Habib[4]

[1] Tokyo University of Science, Yamaguchi, 1-1-1 Daigaku-Dori, Sanyo-Onoda 756-0884, Japan
nagata@rs.tusy.ac.jp
[2] Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan
[3] Okayama University, 3-1-1 Tsushima-naka, Kita-ku, Okayama 700-8530, Japan
[4] American University in Cairo, AUC Avenue, P.O. Box 74, New Cairo 11835, Egypt

Abstract. In this paper, a 3D printer-like data interface is proposed for the machining robot. The 3D data interface enables to control the machining robot directly using STL data without conducting any CAM process. This is done by developing a robotic preprocessor that helps to remove the need for the conventional CAM process by converting directly the STL data into CL data. The STL originally means Stereolithography which is a file format proposed by 3D Systems, and recently is supported by many CAD/CAM softwares. The STL is widely used for rapid prototyping with a 3D printer which is a typical additive manufacturing system. The STL deals with a triangular representation for a curved surface geometry. The developed interface allows to control the machining robot through a zigzag path generated according to the information included in STL data. The effectiveness and potential of the developed approach are demonstrated through actual experimental machining results.

Keywords: Machining robot · CAD/CAM · STL data · 3D printer-like data interface · CL data · Robotic preprocessor

1 Introduction

In manufacturing industries, there exist two representative systems for prototyping. One is the conventional removal manufacturing systems, such as NC milling or NC lathe machines which can precisely perform metalworking. Figure 1(a) shows a typical conventional machining process. The second is the additive manufacturing systems, such as optical shaping apparatus or 3D printer which enables to quickly transform a design concept into a real model. As for removal machining, Lee introduced a machining automation using an industrial robot [1]. The robot had double parallel mechanism and consequently performed a large work space as well as a high stiffness to reduce deformation and vibration. Schreck et al. launched Hard Material Small-Batch Industrial Machining Robot (HEPHESTOS) project, in which the objective was focused on developing

N. Kubota et al. (Eds.): ICIRA 2016, Part I, LNAI 9834, pp. 40–50, 2016.
DOI: 10.1007/978-3-319-43506-0_4

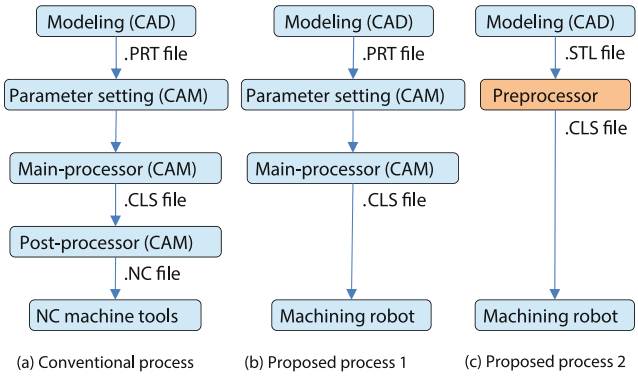

Fig. 1. Comparing the proposed two processes with the conventional process.

robotic manufacturing methods in order to give rise to a cost-efficient solution in hard materials machining [2].

As one of examples of post-processor in CAM system, CL data written in ISO format produced by main-processor in CAM system could be transformed into G-codes files (Numerical Control files) and an industrial 5-axis machine with a nutating table could be actually controlled using the NC files [3]. However, it is not easy but complicated for the CAM system to generate each robot language according to different industrial robot makers. The authors have developed an industrial machining robotic system for foamed polystyrene materials [4]. In the machining robot, the developed robotic CAM system called the direct servo system provided a simple interface for NC data and CL data, without using any robot languages between operators and the machining robot [5]. The direct servo system simplified the machining process without a post-processor as shown in Fig. 1(b). However, a CAM process to generate CL data after the design process has to be further passed through in order to machine the designed model using the robot. Although the authors searched related papers, e.g., [6,7] to make the process easier, any suitable system was not seen.

In this paper, a robotic preprocessor is proposed for the machining robot to directly convert STL data into CL data as the proposed process 2 shown in Fig. 1(c), and this helps to remove the need for having the conventional CAM process. The STL originally means Stereolithography which is a file format proposed by 3D Systems and recently is supported by many CAD and CAD/CAM softwares. It is also known as Standard Triangulated Language in Japan. The STL is widely used for rapid prototyping with a 3D printer which is a typical additive manufacturing system [8]. The STL deals with a triangular representation of a 3D surface geometry [9,10]. The robotic preprocessor allows the machining robot to be controlled along continuous triangular polygon mesh included in STL data or along a zigzag path generated by analyzing triangle patches in the data. The effectiveness and potential of this unique machining system are demonstrated through actual machining experiments.

2 Machining Robot Incorporated with Robotic CAM

A robotic CAM was proposed without using any robot languages to enhance the affinity between an industrial robot RV1A and a CAD/CAM Creo [5]. Figure 2 shows the developed industrial machining robot incorporated with the robotic CAM. The tip of a ball endmill can be controlled as to follow position and orientation components in cutter location within CL data. The CL data were generated by using CAD/CAM Creo provided by PTC Inc. Our current interest is to enable the industrial machining robot to run through STL data that consists of unstructured triangulated patches as shown in Fig. 3. In this paper, a preprocessor is proposed to convert STL data into CL data and integrated with the developed industrial machining robotic system to execute an assigned machining job using CL data. Accordingly, the system can implement its task and control its sequence of machining actions based on STL data.

Fig. 2. Machining robot RV1A for foamed polystyrene materials.

3 Preprocessor Based on Triangle Patches in STL Data

The proposed preprocessor for the industrial machining robotic system consists of three processes, (a) essential process, (b) smart process, and (c) advance process. As for (c) advance process, the details are described in the next section.

3.1 Essential Process

The STL is a file format proposed by 3D Systems and recently is supported by many CAD and CAD/CAM softwares. STL is known as Standard Tessellation Language using triangular patches $\boldsymbol{P}_i = [\boldsymbol{n}_i^T \ \boldsymbol{v}_{1i}^T \ \boldsymbol{v}_{2i}^T \ \boldsymbol{v}_{3i}^T]^T$ as shown in Fig. 3, in which $\boldsymbol{n}_i = [n_{xi} \ n_{yi} \ n_{zi}]^T$ is normal vector, $\boldsymbol{v}_{1i} = [x_{1i} \ y_{1i} \ z_{1i}]^T$,

$\boldsymbol{v}_{2i} = [x_{2i}\ y_{2i}\ z_{2i}]^T$ and $\boldsymbol{v}_{3i} = [x_{3i}\ y_{3i}\ z_{3i}]^T$ are position vectors of vertexes forming a triangular patch, respectively. i denotes the number of the triangular patch in STL file. The content of a triangulated patch in a binary file format is composed of 80 characters header, number of triangulated patches, normal vector to triangle face, position vectors of three vertexes describing each triangulated patch, and attribute byte count, which is written as

```
char[80]    //Header
uint32      //Number of triangles i in a STL file
float[3]    //Normal vector n(1)
float[3]    //Vertex vector v1(1)
float[3]    //Vertex vector v2(1)
float[3]    //Vertex vector v3(1)
uint16      //Attribute byte count
  ⋮                 ⋮
float[3]    //Normal vector n(i)
float[3]    //Vertex vector v1(i)
float[3]    //Vertex vector v2(i)
float[3]    //Vertex vector v3(i)
uint16      //Attribute byte count
  ⋮                 ⋮
```

As can be seen from Fig. 3, three "GOTO" statements with the same normal vector can be generated as CL data from one STL triangulated patch as below,

GOTO/Vertex vector $\boldsymbol{v}_1^T(i)$, Normal vector $\boldsymbol{n}^T(i)$
GOTO/Vertex vector $\boldsymbol{v}_2^T(i)$, Normal vector $\boldsymbol{n}^T(i)$
GOTO/Vertex vector $\boldsymbol{v}_3^T(i)$, Normal vector $\boldsymbol{n}^T(i)$

Figure 4 shows an example of plotted output of all converted data (STL data → CL data → NC data), in which the original STL data was generated by CAD/CAM Creo. The NC data is used as unique desired trajectory based on continuous triangular polygon mesh. As can be seen, if the conversion is

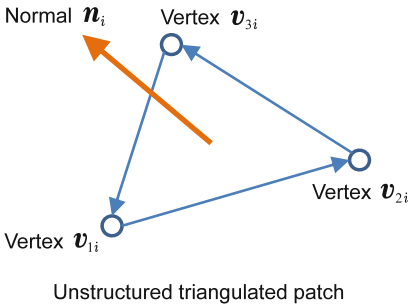

Fig. 3. STL file format proposed by 3D systems.

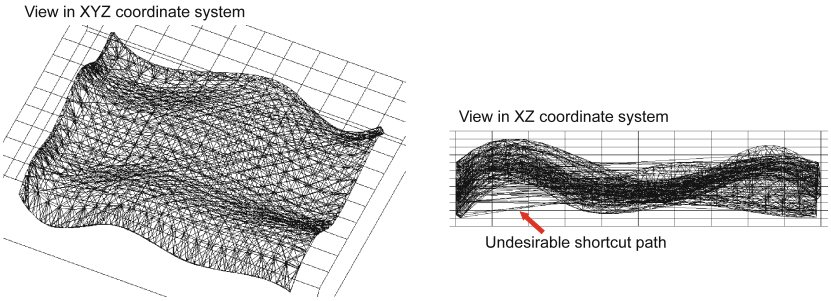

Fig. 4. Output of NC data directly converted from STL data.

conducted without any step and consideration, undesirable shortcut paths frequently appear. However, such problems will be avoided when the STL data are used for additive layered machining with a 3D printer. But, when the NC data directly obtained from the STL data is applied to the removal machining with a ball endmill, serious mis-removal cutting tends to occur according to the undesirable shortcut paths.

3.2 Smart Process

To overcome the problem of mis-removal cutting, a smart process is developed within the robotic machining preprocessor. Figure 5 shows the Window dialogue designed for the robotic machining preprocessor and post-processor. There are two parameters for the smart process in the robotic machining preprocessor that should be defined as shown in Fig. 6. One is the height h [mm] of tool escape in z-direction. The other is the distance $d_i = \|\boldsymbol{v}_{1i} - \boldsymbol{v}_{3(i-1)}\|$ [mm] from a vertex $\boldsymbol{v}_{3(i-1)}$ to the one $\boldsymbol{v}_{1i}$ in the next triangle patch. If d_i is longer than

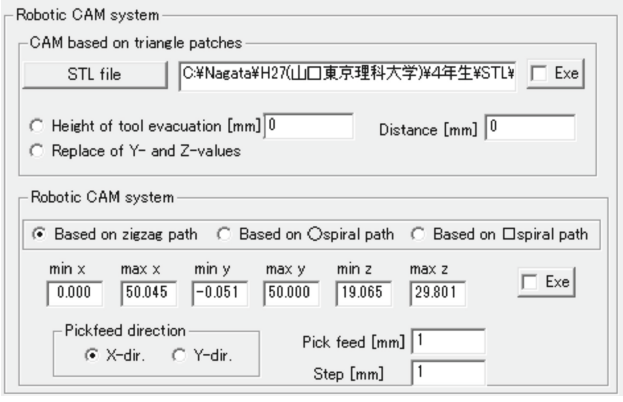

Fig. 5. Dialogue designed for robotic machining preprocessor to process STL data.

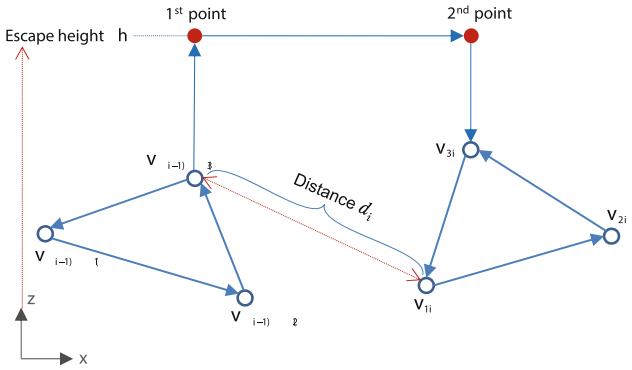

Fig. 6. Two points with the height h are added to avoid mis-removal cuttings.

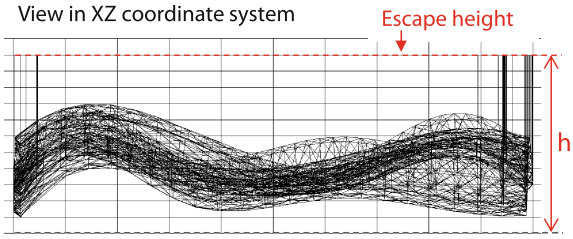

Fig. 7. An example of paths corrected by the smart process.

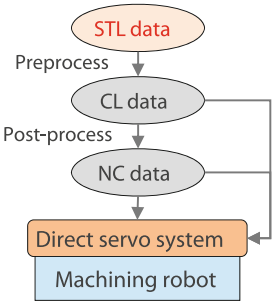

Fig. 8. Robotic CAM system available for NC data, CL data and STL data.

the reference value d_r of distance set in the dialogue, then the preprocessor inserts two positions for escaping. The first position is for the escape just upper direction. Also, the second position is for the movement from the current patch to the next one. Figure 7 shows an example of paths improved by the smart process, in which h and d_r were empirically set to 20 mm and 10 mm, respectively, by checking the sizes of the model and the triangulated patches in the STL data.

With this process, the proposed machining robot can run based on not only NC data and CL data but also STL data as shown in Fig. 8.

4 Preprocessor by Intelligently Analyzing Triangle Patches in STL Data

4.1 Automatic Dimension Extraction of STL Data

In the two processes explained in the previous section, CL data are basically generated along original STL data consisting of many triangle patches. In this subsection, (c) advance process is described, in which the CL data for robot control are generated along a zigzag path by intelligently analyzing STL data. The significant advantage of this approach is to eliminate the need to use any commercial CAM, and accordingly, 3D printer-like data interface can be smartly realized.

First of all, dimensions of the STL data are extracted by retrieving all patches in a STL file and they are set to two constants $\boldsymbol{v}_{\min} = [x_{\min}\ y_{\min}\ z_{\min}]^T$ and $\boldsymbol{v}_{\max} = [x_{\max}\ y_{\max}\ z_{\max}]^T$. From the next subsection, a base zigzag path viewed in xy-plane is designed considering the extracted dimensions.

4.2 Design of Base Zigzag Path

This subsection explains how the base zigzag path is designed. Two effective machining parameters, i.e., pick feed and step, are respectively set to p_f and s_p by referring the dimensions. s_p is a constant pitch viewed in xy-plane between two adjacent points $\boldsymbol{c}_j = [c_{xj}\ c_{yj}\ 0]^T$ and $\boldsymbol{c}_{j+1} = [c_{x(j+1)}\ c_{y(j+1)}\ 0]^T$ on a zigzag path. The scallop height is caused by the machining with a ball endmill. Although the height depends on the pick feed and the ball radius of an endmill, they are experimentally determined while avoiding the interference between the endmill and the designed STL model.

One pass point $\boldsymbol{c}(j)$ is appended into a CL file as a "GOTO" statement. $j(1 \leq j \leq m)$ is the number of the pass point generated from STL data. m is the total number of triangle patches in a STL file. Figure 9 illustrates a base zigzag path drawn within STL data consisting of multiple triangulated patches. As can be seen, c_{xj} and c_{yj} are located just on the zigzag path, so that remained height c_{zj} has only to be determined by analyzing triangle patches in STL data.

4.3 Generation of CL Data Along Base Zigzag Path

Figure 10 shows an example of generation of pass points $\boldsymbol{c}_j = [c_{xj}\ c_{yj}\ c_{zj}]^T$ in a patch (2). The number within the patch depends on the length of the step in Fig. 9. The dashed line in the upper figure shows one section of the base zigzag paths $\boldsymbol{c}_j = [c_{xj}\ c_{yj}\ 0]^T$ viewed in xy-plane and the chained line in the lower figure draws the generated path $\boldsymbol{c}_j = [c_{xj}\ c_{yj}\ c_{zj}]^T$ along the triangulated patch viewed in yz-plane. The pass points for constructing CL data are generated along

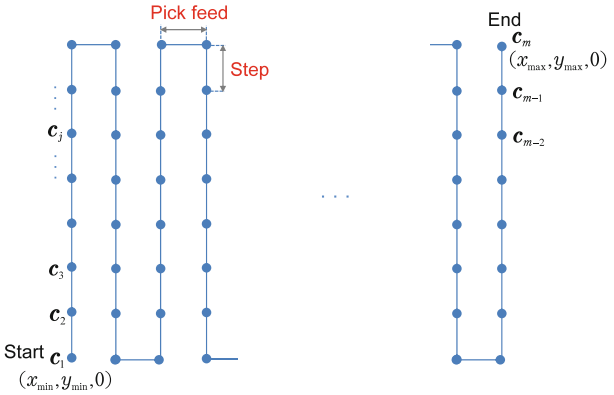

Fig. 9. Base zigzag path $\boldsymbol{c}_j = [c_{xj}\ c_{yj}\ 0]^T$ along STL data consisting of multiple triangulated patches.

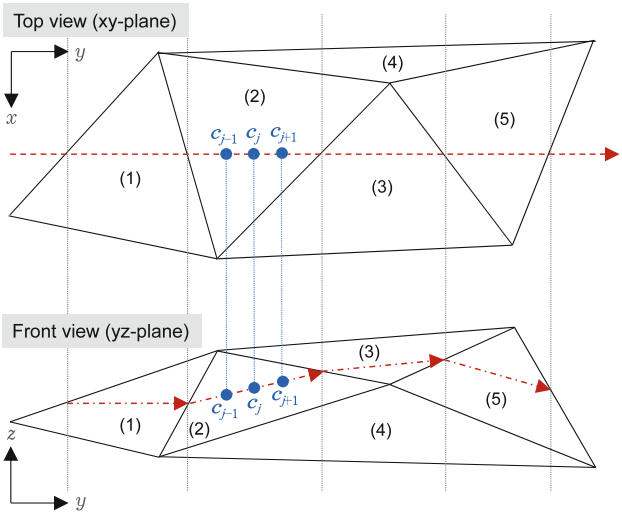

Fig. 10. Generation of pass points. Upper and lower figures show the top and front views of five adjacent triangulated patches, respectively.

the chained line. The pass points in other patches such as (1), (3), (4), (5) can be similarly obtained.

To realize the preprocessor based on STL data without any CAM process, the z-component c_{zj} must be calculated just on a triangulated patch. In order to calculate c_{zj}, first of all, a triangulated patch, in which the point $\boldsymbol{c}_j$ viewed in xy-plane is included, is searched in the target STL data. Figure 11 shows the scene where the pass point $\boldsymbol{c}_j$ is located within a patch. Note that $\boldsymbol{c}_{j-1}$ and $\boldsymbol{c}_{j+1}$ may be also within the triangle patch as shown in Fig. 10. Whether $\boldsymbol{c}_j$ is located

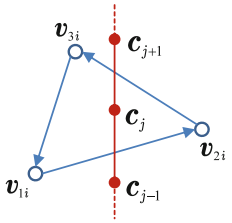

Fig. 11. Pass point $\boldsymbol{c}_j$ is located within a triangle patch viewed in xy-plane.

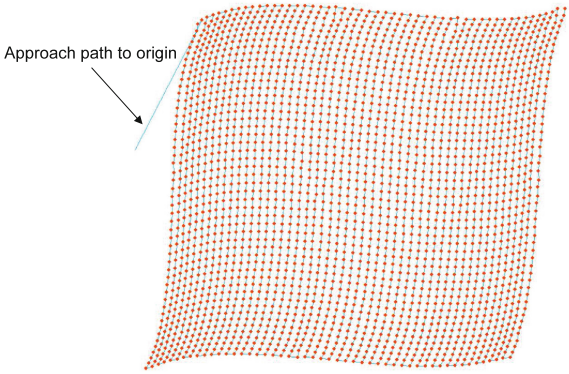

Fig. 12. Regular and precise zigzag path $\boldsymbol{c}_j = [c_{xj}\ c_{yj}\ c_{zj}]^T$ generated by the pre-processor.

in the patch or not can be known by checking the following outer products.

$$(\boldsymbol{v}_{2i} - \boldsymbol{v}_{1i}) \times (\boldsymbol{c}_j - \boldsymbol{v}_{1i}) \tag{1}$$
$$(\boldsymbol{v}_{3i} - \boldsymbol{v}_{2i}) \times (\boldsymbol{c}_j - \boldsymbol{v}_{2i}) \tag{2}$$
$$(\boldsymbol{v}_{1i} - \boldsymbol{v}_{3i}) \times (\boldsymbol{c}_j - \boldsymbol{v}_{3i}) \tag{3}$$

If $\boldsymbol{c}_j$ is located within the patch, then the above three equations have the same sign. After finding the first triangle satisfying this condition, the equation of the plane including the triangle is determined by the perpendicular condition to the normal vector $\boldsymbol{n}_i$, which leads to

$$n_{xi}(x - x_{1i}) + n_{yi}(y - y_{1i}) + n_{zi}(z - z_{1i}) = 0 \tag{4}$$

By respectively substituting c_{xj} and c_{yj} into x and y in Eq. (4), if $n_{zi} \neq 0$ then z-directional component c_{zj} can be calculated by

$$c_{zj} = z_{1i} - \frac{1}{n_{zi}}(n_{xi}(c_{xj} - x_{1i}) + n_{yi}(c_{yj} - y_{1i})) \tag{5}$$

where c_{xj} and c_{yj} are extracted from the base path shown in Fig. 9.

Consequently, by repeating the above calculations, all pass points $\boldsymbol{c}_j = [c_{xj}\ c_{yj}\ c_{zj}]^T$ $(1 \leq j \leq m)$ can be obtained. Figure 12 shows the regular and precise zigzag path (CL data) generated from a STL file by the preprocessor.

4.4 Removal Machining Experiment

In the earlier subsections, the preprocessor that can generate a regular and precise zigzag tool path without conducting any CAM process has been proposed. In this subsection, a machining experiment is conducted using the tool path generated by the preprocessor. Figure 13 shows the successful machining scene using the CL data made along a zigzag path (left side) and the resultant surface (right side). The feasibility and effectiveness were confirmed from the actual experiment using the machining robot.

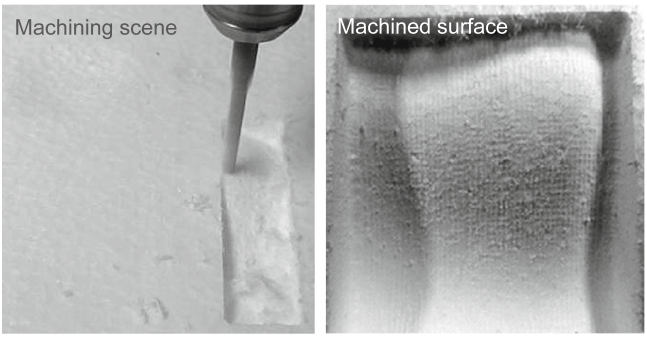

Fig. 13. Machining scene using CL data generated by preprocessor and its resultant surface.

5 Conclusions

The STL file format was designed for fabbers in 1989. Fabbers means specialists who can perform 3D rapid prototyping from digital data, e.g., using a 3D printer. The 3D printer is recognized as a typical additive manufacturing system. In this paper, a robotic preprocessor has been proposed for the machining robot to convert STL data into CL data forming a zigzag path. The STL means Stereolithography which is a file format proposed by 3D Systems and recently is supported by many CAD/CAM softwares. The robotic preprocessor has allowed the machining robot to be controlled along continuous triangular polygon mesh included in STL data or along a zigzag path obtained by analyzing triangle patches in the data. The effectiveness and promise of this unique machining system are demonstrated through actual machining experiments. The noteworthy point is that the machining robot has a promising data interface without CAM process like a 3D printer.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Number 25420232.

References

1. Lee, M.K.: Design of a high stiffness machining robot arm using double parallel mechanisms. In: Proceedings of 1995 IEEE International Conference on Robotics and Automation, vol. 1, pp. 234–240 (1995)
2. Schreck, G., Surdilovic, D., Krueger, J.: HEPHESTOS: hard material small-batch industrial machining robot. In: Proceedings of 41st International Symposium on Robotics (ISR/Robotik 2014), pp. 1–6 (2014)
3. My, C.A.: Integration of CAM systems into multi-axes computerized numerical control machines. In: Proceedings of 2010 Second International Conference on Knowledge and Systems Engineering (KSE), pp. 119–124 (2010)
4. Nagata, F., Otsuka, A., Watanabe, K., Habib, M.K.: Fuzzy feed rate controller for a machining robot. In: Proceedings of the 2014 IEEE International Conference on Mechatronics and Automation (IEEE ICMA 2014), pp. 198–203 (2014)
5. Nagata, F., Yoshitake, S., Otsuka, A., Watanabe, K., Habib, M.K.: Development of CAM system based on industrial robotic servo controller without using robot language. Rob. Comput.-Integr. Manuf. **29**(2), 454–462 (2013)
6. Al-Ahmari, A., Moiduddin, K.: CAD issues in additive manufacturing. In: Comprehensive Materials Processing. Advances in Additive Manufacturing and Tooling, vol. 10, pp. 375–399 (2014)
7. Matta, A.K., Ranga Raju, D., Suman, K.N.S.: The integration of CAD/CAM and rapid prototyping in product development: a review. Mater. Today Proc. **2**(4/5), 3438–3445 (2015)
8. Brown, A.C., Beer, D.D.: Development of a stereolithography (STL) slicing and G-code generation algorithm for an entry level 3-D printer. In: Proceedings of IEEE African Conference 2013, pp. 1–5 (2013)
9. Szilvasi-Nagy, M., Matyasi, G.: Analysis of STL files. Math. Comput. Model. **38**(7/9), 945–960 (2003)
10. Iancu, C., Iancu, D., Stancioiu, A.: From CAD model to 3D print via STL file format. Fiability Durability **1**(5), 73–81 (2010)