한국기계가공학회지, 제16권, 제3호, pp.1～8(2017.06) ISSN 1598-6721(Print)
Journal of the Korean Society of Manufacturing Process Engineers, Vol.16, No.3, pp.1~8(2017.06) ISSN 2288-0771(Online)

https://doi.org/10.14775/ksmpe.2017.16.3.001

An Algorithm for the Removing of Offset Loop Twists during the Tool Path Generation of FDM 3D Printer

Islam Md. Olioul[*] and Ho-Chan Kim[*,#]
[*]Department of Mechanical and Automotive Engineering, Andong National UNIV.

FDM 3D 프린팅의 경로생성을 위한 옵셋루프의 꼬임제거 알고리즘

올리올 이슬람[*], 김호찬[*,#]
[*]안동대학교 기계자동차공학과

(Received 15 March 2017; received in revised form 20 March 2017; accepted 5 April 2017)

ABSTRACT

Tool path generation is a part of process planning in 3D printing. This is done before actual printing by a computer rather than an AM machine. The mesh geometry of the 3D model is sliced layer-by-layer along the Z-axis and tool paths are generated from the sliced layers. Each 2-dimensional layer can have two types of printing paths: (i) shell and (ii) infill. Shell paths are made of offset loops. During shell generation, twists can be produced in offset loops which will cause twisted tool paths. As a twisted tool path cannot be printed, it is necessary to remove these twists during process planning. In this research, An algorithm is presented to remove twists from the offset loops. To do so the path segments are traversed to identify twisted points. Outer offset loops are represented in the counter-clockwise segment order and clockwise rotation for the inner offset loop to decide which twisted loop should be removed. After testing practical 3D models, the proposed algorithm is verified to use in tool path generation for 3D printing.

Keywords : Offset Loop Twist(옵셀루프꼬임), STL Slicing(STL 슬라이싱), Tool Path Generation(경로생성), 3D Printing(3D 프린팅)

1. Introduction

3D printing technology is making progress day by day to be used industrially[1]. It is a promising technique for fabricating complex 3D architectures based on the CAD/CAM system, and it has been extensively investigated to manufacture structures in the fields of mechanical engineering, space technology, automobiles, and biomedical and electrical applications[2-4]. 3D printing or Additive Manufacturing (AM) is a process where a 3D CAD model is sliced into several 2D cross-sections and produced physically layer-by-layer using material deposition[5]. Instead of conventional material removal methods, the layer-by-layer material deposition manufacturing approach has gained worldwide popularity over the past thirty years[6,7]. Therefore, the manufacturing by

# Corresponding Author : hckim@andong.ac.kr
Tel: +82-54-810-5269, Fax: +82-54-810-5044

3D printer is divided into two main strategies; (a) virtual strategy or process planning, (b) physical strategy or actual printing and postprocess[8]. To describe the 3D CAD model, STL files are the most commonly used data format in this technology[9]. STL files contains triangular tessellation to define the surface of the model. It is a geometrical representation of 3D models.[9] In process planning, STL file is sliced in different heights and tool path is generated for all the sliced layers. One of the steps of STL slicing, mentioned in the next section, offset loops are generated in order to generate shell tool path. In this step, twists can be produced in offset loops. Fig. 1 shows an example of the twists in offset loops.

In this paper, an algorithm was developed to remove twists in offset loops. The generated tool paths for different samples were tested using FDM 3D system. Resin was used to print the sample models. The result shows that this algorithm was sufficiently feasible to use in tool path generation for 3D printing.

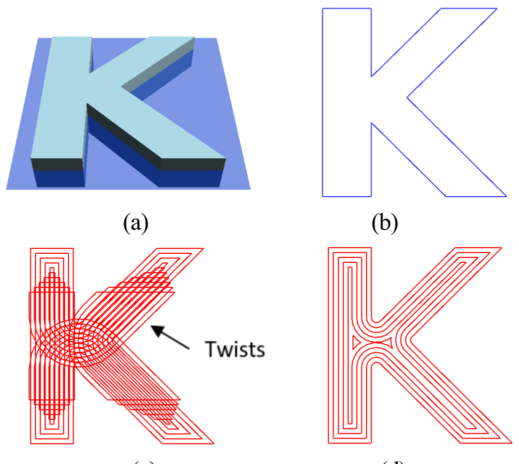

Fig. 1 (a) Slicing STL model in a height; (b) Sliced cross-sectional; (c) Twists in offset loops; (d) Tool path after removing twist.

2. Slicing STL Model

2.1 Introduction

Slicing STL model is one of the main section in process planning. As the slicing, itself is a complex procedure and includes large data handling, a slicing software runs into computer instead of AM machine and it slices STL models layer-by-layer along the Z-axis to generate tool path. Later the software saves the tool path into a file as G-code. The following are the main steps of slicing process:

#Step 1. Slicing STL Model:

Step 1.1. Reading and handling large STL file
Step 1.2. Finding and fixing STL errors
Step 1.3. Slicing the model in different heights
Step 1.4. Extracting cross-sections from the sliced layers
Step 1.5. Determining inner and outer loops
Step 1.6. Extracting loop inclusion tree
Step 1.7. Generating offset loops
Step 1.8. Removing offset loop twists

#Step 2. Generating Tool Path:

Step 2.1. Generating cutter location
Step 2.2. Generating position code
Step 2.3. Generating tool path
Step 2.4. Formatting tool path into G-code

2.2 Slicing STL

To read the STL file, first it needs to determine whether the file is ASCII formatted or binary formatted. After determining the file format, separate algorithms are used to read the file because they have different syntax formats. In 3D mesh model, one vertex is shared by more than one triangles. Whatever, in STL file, each vertex data stored redundantly among all the facets the vertex is associated with. Kim Seon Yeop described that if the redundant vertices can be identified, the STL data size can be reduced in a notable amount.[10] This identification of redundant vertices helped to extract the topological information of the input mesh object.

Final Equations for $P_h < x_h, y_h, z_h >$:

$$x_h = \left(\frac{z_h - z_1}{z_2 - z_1}\right)(x_2 - x_1) + x_1$$

$$y_h = \left(\frac{z_h - z_1}{z_2 - z_1}\right)(y_2 - y_1) + y_1$$

$$z_h = \textit{Slice Height}$$

Fig. 2 3D line equations for STL slicing

Moreover, the computer processing time reduced while working with identified small sized data. By counting the faces that each half edges have, the STL hole errors and invalid edges errors can be found easily. After finding and fixing the STL errors, the triangles of the mesh are sliced in different heights. In 3-dimensional surface, the slicing was done using 3D line equations which was shown in Fig. 2.

2.3 Extracting loop information

After slicing the mesh triangles in different heights, the points were being connected to create cross-sectional loops. Then the loop information is extracted from the cross-sectional loops. Loop information includes the following three types of information:

a. Loop Inclusion Tree, b. Open Loop (Error)

c. Closed Loop

i. Inner Loop, ii. Outer Loop

Islam M. Olioul described the algorithms to extract loop information in his work titled "Classification of Loop Inclusion for Slicing STL Model".[11] The basic idea was, if a loop has the same start and end points, it is classified as a closed loop and if not then it is an open loop. One of the basic property of outer loop is the faces of the triangles are always towards outer direction and the faces of the triangles of the inner loop is always towards inner direction .[11] The algorithm can be summarized into the following steps:

Step 1. Finding the bottom-leftmost point

Step 2. Extracting two loop segments on that point

Step 3. Considering the segment with grater angle from X-Axis as left segment

Step 4. Making decision based on the normal vector (face) of left segment

Loop inclusion tree explains the arrangement of loops in a cross-section using parent-child tree data structure. If a straight line is drawn from any point of one loop, the line will intersect with all its parent loops in that cross-section. This strategy was used to find the loop inclusion tree.

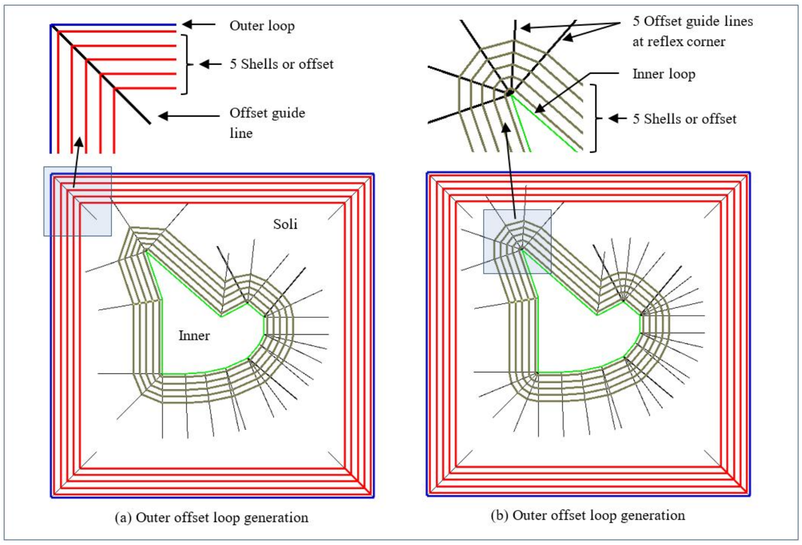
Outer loop
5 Shells or offset
Offset guide line
Soli
Inner
(a) Outer offset loop generation
5 Offset guide lines at reflex corner
Inner loop
5 Shells or offset
(b) Outer offset loop generation

Fig. 3 Generating offset loops using offset guide lines

2.4 Generating tool path

In tool path generation for FDM 3D printer, the process planning includes two types of path generation: (i) shell generation and (ii) infill generation. Both for shell and infill generation, those need to generate the offset loops first. This process includes removing the twists in offset loops.

2.4.1 Generating offset loop

The main idea of the algorithm which was used in offset loop generation is, taking some guide lines at each of the turns in a cross-sectional loop. The number of guide lines in each turn depends on the angle between the adjacent segments from which the turn or corner was created. After taking the guide lines, some offset points are calculated on the guide lines. The number of the offset points are equal to the number of requested shell. Adding those points among the guide lines creates offset loops.

When the path of an offset loop intersects itself, it creates a twist. In a result, several twisted loops can be created from one offset loop. But not all those twisted loops are to be printed. Fig. 1(c) shows an example model with twisted offset loops. Fig 3 shows the process of generating offset loops. In the following sections, we will be describing the algorithm to remove twists in offset loop. Fig. 4 shows a sliced layer where the tool path contains both shell and infill.

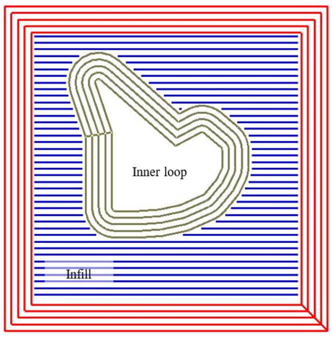

Fig. 4 Tool path for a sliced layer with shell and infill

2.4.2 Removing twists in offset loop

A recursive method was used to remove the twist from offset loops. The method traverses through the offset loop segments and search for the intersections with the previously traversed segments of the same loop. That is why the newly created twisted loops demand to be removed. Whenever the algorithm finds any intersection with the previous segments, it creates a new loop where the start and end of the loop is that very intersecting point. In the following Fig. 5, Step 1(a) is showing a sliced layer of the STL model. The STL model is presented in Fig. 6(a). Step 1(b) is showing the cross sectional layer with a single offset loop. Step 2 is showing the traversing through the loop segments and the detection of the first intersection between segment 10 and segment 4. At this step, at that intersecting point, segment 4 and 10 were divided. The newly created closed loop from the second part of segment 4 to the first part of segment 10 was then listed as a new sub-offset loop in Step 3(b). Step 2 was called again as a recursive method with the remaining segments in Step 3(a).

This recursion was continued while the traversing reached to the end, which again intersects the first segment and creates the last listed loop. After separating and listing all the sub-offset loops, it shows that there is no twist left in the offset loop.

From the characteristics of STL file format, it is known that the triangles rotate counter clock wise in outer loop and clock wise in inner loop. So, this same property was implemented to the loop segments when the cross-sectional loops were classified. The segments of the outer loops were stored counter clock wise and the segments of the inner loops were stored clock wise. At this point, the rotation direction is used to identify invalid sub-offset loops which are created by the twist. To do that, the bottom-leftmost point was selected for each listed sub-offset loops and the left segments from that points are checked that either it goes towards up direction or down direction. If the sub-offset loop was coming from an outer loop of sliced cross-section, the left segment should go towards down direction. Otherwise the loop is listed as an invalid and is removed from the list of sub-offset loops. This property is reverse for the inner offset loops. In the above example, the loop in the Step 5(b) was removed as a twist. Fig. 5 shows different steps of removing twists in offset loop.

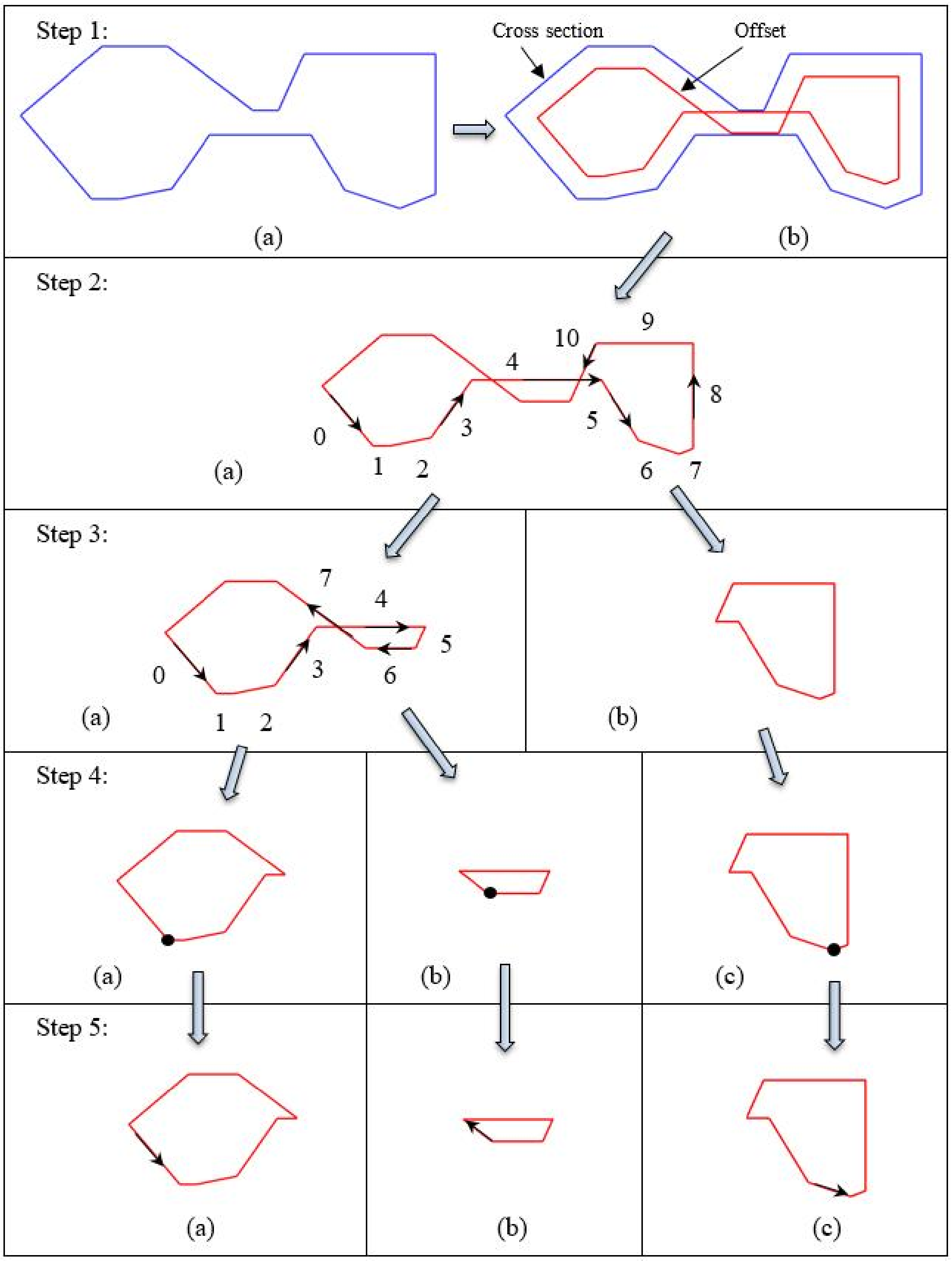

Fig. 5 Different steps of removing twists in offset loop

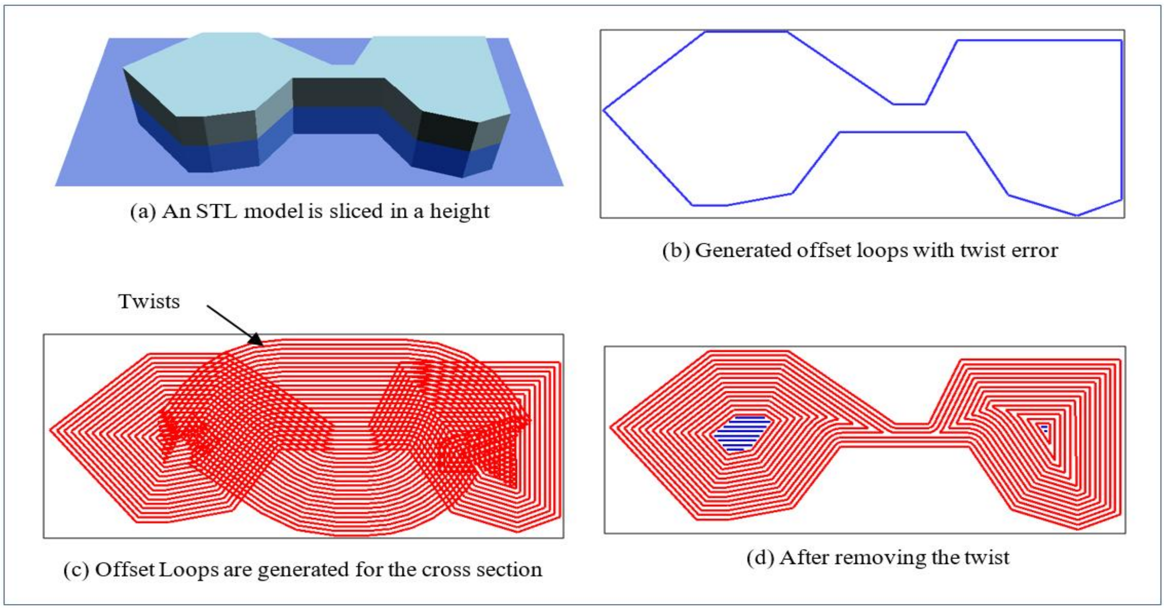
(a) An STL model is sliced in a height
(b) Generated offset loops with twist error
Twists
(c) Offset Loops are generated for the cross section
(d) After removing the twist

Fig. 6 Slicing an STL model in a height and generation tool path after removing twists in offset loop

The following Fig. 7 and Fig. 8 show the flowchart of the algorithm to remove twist from offset loop. In details, the pseudocode for the algorithm is as following:

```
#Start
Initiate _segments with all segments of offset loop
For each nth segment from the segments
  For each xth segment, where 0<x<n
    If _segments[n] intersects with _segments[x]
      Store intersecting point into _iPoint
      Devide _segments[n] and _segments[x] at _iPoint
      Create _newLoop from _iPoint of _segments[x] -
      - to _iPoint of _segments[n]
      Add _newLoop to _newLoops
      Remove _segments[x] to _segments[n] from point _iPoint
    End If
    Increase n value by one to proceed to next segment
  End For
End For
For each _newLoop
  Store bottom leftmost point into _blPoint
  Select left segment from _blPoint
  If left segment is going towards down direction
    If the _newLoop created from outer loop
      Keep the loop as valid
    Else
      Remove the loop as invalid
    End If
  If left segment is going towards up direction
    If the _newLoop created from inner loop
      Keep the loop as valid
    Else
      Remove the loop as invalid
    End If
  End If
End For
#End
```

This algorithm was applied to several complex models and found working correctly. Fig. 6 shows one simulated result of removing twist for a test sample. In the following section, we will present the result of actual printing experiments using out generated tool path.

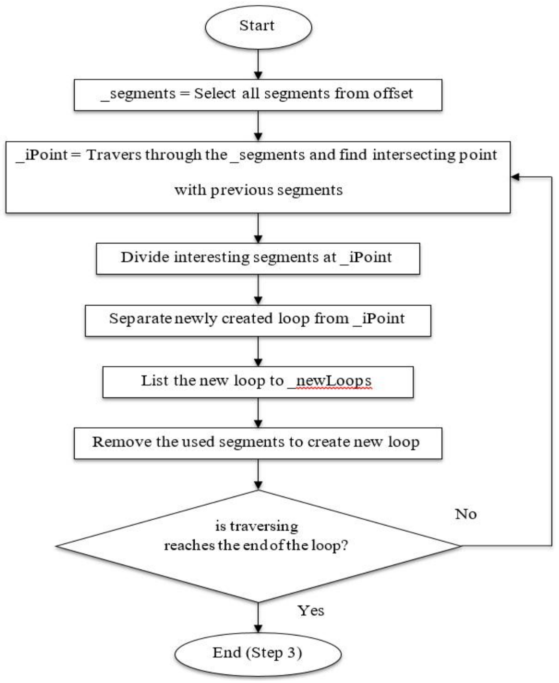
Start
_segments = Select all segments from offset
_iPoint = Travers through the _segments and find intersecting point with previous segments
Divide interesting segments at _iPoint
Separate newly created loop from _iPoint
List the new loop to _newLoops
Remove the used segments to create new loop
is traversing reaches the end of the loop?
No
Yes
End (Step 3)

Fig. 7 (Step 1 to Step 3) Removing twist from offset loop)

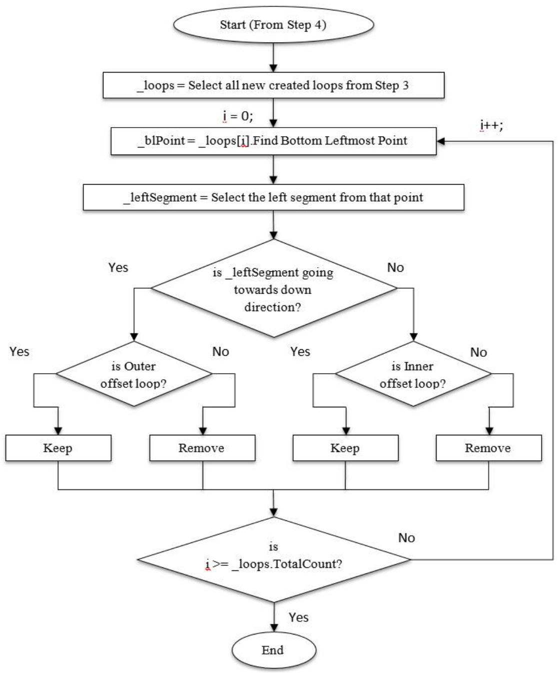
Start (From Step 4)
_loops = Select all new created loops from Step 3
i = 0;
i++;
_blPoint = _loops[i].Find Bottom Leftmost Point
_leftSegment = Select the left segment from that point
is _leftSegment going towards down direction?
Yes
No
is Outer offset loop?
is Inner offset loop?
Keep
Remove
Keep
Remove
is i >= _loops.TotalCount?
No
Yes
End

Fig. 8 (Step 4 to Step 5) Removing twist from offset loop)

3. Experiment

3.1 Experiment setup and conditions

The FDM 3D printing system developed by Yun Hae-yong,[12] which uses resin as printing material, was used in this experiment. The experimental parameters were listed in Table 1.

4. Results and Discussion

4.1 The characteristics of printed products

Fig. 10 represents the actual printing experiment using our developed tool path generator. The printing parameters and machine setup were used as described in Table 1. It clarifies that there is no twists exist in the offset loops.

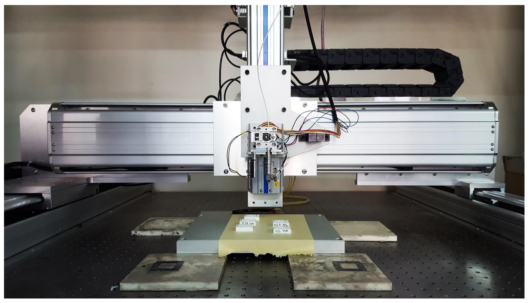

Fig. 9 Experimental set-up – FDM 3D system

Table 1 Experimental parameters

Parameter	Value
Material	PLA
Nozzle diameter	0.4 mm
Layer height	0.2 mm
Platform temperature	60 ℃
Material temperature	230 ℃

5. Conclusion

In this study, the area of the inner most offset loop was not considered. It may cause some material overlap in an inner most offset loop with very small area. Regardless of this error, the simulation and

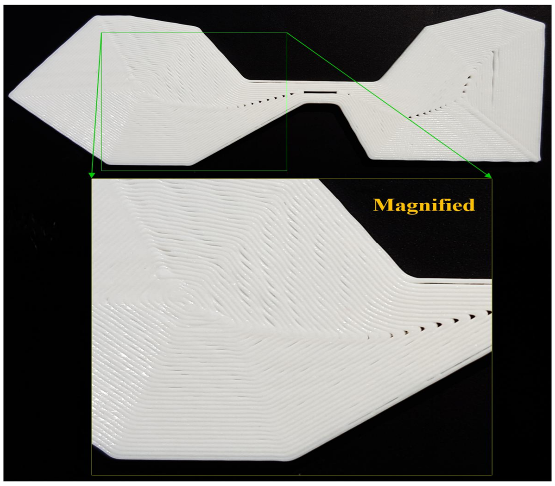

Fig. 10 Actual printed product using FDM 3D printing system and resin material

printing results show that this algorithm is feasible to use in tool path generation for 3D printing.

Acknowledgement

We thank Hae-yong Yun, PhD and Sung Chur Park, BSc for assistance with actual printing experiment that greatly improved the manuscript.

REFERENCES

1. Kim, T. H., Kim, J. S., Yoon, J. M., Kim, J. H., Cho, K. W. and Kim, S. H., “Popular 3D Printer,” Proceedings of the KSMPE Conference, pp. 171-171, 2015.
2. Yun, H. Y., Kim, H. C. and Lee, I. H., "Fabrication of 3D-Printed Circuit Device using Direct-Write Technology," Journal of the Korean Society of Manufacturing Process Engineers, Vol. 15, No.2, pp. 1-8, 2016.
3. Zhang, Y., and Bernard, A., "AM Feature and Knowledge Based Process Planning for Additive Manufacturing in Multiple Parts Production Context," Proceedings of 25th Annual International Solid Freeform Fabrication Symposium, pp. 1259-1276, 2014.
4. Lee, I. H., Shin, J. M. and Cho, H. Y., "Design and Operation of 3D Printing Education Curriculum in Mechanical Engineering," Journal of the Korean Society of Manufacturing Process Engineers, Vol. 14, No. 3, pp. 21-26, 2015.
5. Jang, J. N. and Cho, D. W., "A Review of The Fabrication of Soft Structures with Three-dimensional Printing Technology," Journal of the Korean Society of Manufacturing Process Engineers, Vol. 14, No. 6, pp. 142-148, 2015.
6. Levy, G. N., Schindel, R. and Kruth, J. P., "Rapid Manufacturing and Rapid Tooling with Layer Manufacturing (LM) Technologies, State of The Art and Future Perspectives," CIRP Annals-Manufacturing Technology, Vol. 52, No. 2, pp. 589-609, 2003.
7. Shim, J. H., Yun, W. S. and Ko, T. J., "Successful Examples of 3D Printing Technology-based Start-up Enterprises," Journal of the Korean Society of Manufacturing Process Engineers, Vol. 15, No. 2, pp. 104-110, 2016.
8. Ahsan, A. N., Habib, M. A. and Khoda, B., "Resource Based Process Planning for Additive Manufacturing," Computer-Aided Design, Vol. 69, pp. 112-125, 2015.
9. Szilvśi-Nagy, M. and Mάtyάsi, G., "Analysis of STL Files," Mathematical and Computer Modelling, Vol. 38, No. 7, pp. 945-960, 2003.
10. Kim, S. Y., Kim, H. C., and Islam, M. O., "Extracting the Topology Information from STLModel in Order to Generate Cross-sectional Loops Efficiently," Proceedings of the KSPE Conference Spring, pp. 12, 2016.
11. Islam, M. O. and Kim, H. C., "Classification of Loop Inclusion for Slicing STL Model," Proceedings of KSMPE Conference Spring, pp. 42, 2016.
12. Yun, H. Y., Kim, H. C. and Lee, I. H., "Research for Improved Flexible Tactile Sensor Sensitivity," Journal of Mechanical Science and Technology, Vol. 29, Issue 12, pp. 5133-5138, 2015.