The Improvement of the Closed Bounded Volume (CBV) Evaluation Methods to Compute a Feasible Rough Machining Area Based on Faceted Models

Himawan Hadi Sutrisno¹, Gandjar Kiswanto ², Jos Istiyanto ²
¹ Mechanical Engineering Dept., Universitas Negeri Jakarta, Indonesia
² Dept. of Mechanical Engineering, Universitas Indonesia, Depok, Indonesia

corresponding author: gandjar_kiswanto@eng.ui.ac.id

Abstract. The rough machining is aimed at shaping a workpiece towards its final form. This process takes up a big proportion of the machining time due to the removal of the bulk material which may affect the total machining time. In certain models, the rough machining has limitations especially on certain surfaces such as turbine blade and impeller. CBV evaluation is one of the concepts which is used to detect of areas admissible in the process of machining. While in the previous research, CBV area detection used a pair of normal vectors, in this research, the writer simplified the process to detect CBV area with a slicing line for each point cloud formed. The simulation resulted in three steps used for this method and they are: 1. Triangulation from CAD design models, 2. Development of CC point from the point cloud, 3. The slicing line method which is used to evaluate each point cloud position (under CBV and outer CBV). The result of this evaluation method can be used as a tool for orientation set-up on each CC point position of feasible areas in rough machining.

1. Introduction
Along with the development in the manufacturing industry, milling machine models have gone through a transformation with the purpose of fulfilling the consumers’ needs. Although this transformation has significantly improved the complex models, the 5-axis milling still has its limitation. Also, there hasn’t been any improvement of the machining method especially for rough machining. Until now, the rough machining commonly adopts 3-axis machining methods. Certain models which contain the CBV need specific handling. This type of machining depends on machine operator’s experience so it is likely for error to occur in the product.

At the same time, the developers are still trying to improve the model to solve other difficulties in the manufacturing. To improve productivity, a new development method is needed for the machine to reduce the total machining time. This paper introduced a more advanced method called 5-axis milling for improving the rough machining efficiency.

2. Related work
In order to reduce errors during handling made by the operators, a few researchers have attempted to optimize the rough machining methods[1-5], the tool orientation methods[6-8], the prevention of collision [9, 10], integrated CAM system[11], and the improvement of tool sequence selection[12]
There have been a few studies conducted discussing the CBV evaluations. Woo[13] introduced the concept of visibility cone which can be mapped onto a unit sphere. This visibility method provides an alternative way for detection if there is an obstacle on the workpiece surface during the machining process. Suthunyatanakit, et al. [14] evaluated CBV by putting an obstacle on a light ray on the surface machining. If the light ray is blocked by the obstacle, then the area will be focused to calculate inaccessibility indication where it is not visible by the simple method.

Balasubramaniam et. al. [10] proposed that graphic engine was used for visibility computation for generating collision free in 5-axis machining process especially for finishing. This concept uses a graphic hardware to show the machine limitation, the tool tilt and the accessibility tool for surface machining. Li, et al. [15] used the concept of visibility to describe the reachability of a light ray to the object surface.

In previous research, development CBV grouping method for complex faceted models has been presented by Kiswanto and Panuju[16] by pairing two triangles of normal vectors in one particular field and the two triangles were grouped in a bucket. The two triangles of normal vectors were grouped according to the machining characteristic surface which are CBV area and OBV area. The OBV area is usually called free surface without any obstacles for other surfaces. It needs a normal vector visualization which occupies the workpiece surfaces. In one workpiece model, the many triangles can be used to simplify the calculation process and for this purpose, bucketing method is required.

3. Improvement of CBV evaluation methods to compute feasible rough machining

In this study, the database for the calculation is the faceted models (triangulation), or also known the triangular mesh, polyhedral models, as well as tessellated models. This model has been used by many researchers due to its advantages in data processing and time computing. Since the CAD design is represented by a triangle, the shape of the triangulation result is an approximation. The accuracy of triangulation depends on the number of triangle resolution. Higher resolution results in a better shape visualization.

To get visualization needed, the computation uses Matlab software with STL file as the first data. The STL file contains information related with the workpiece model, the features and the dimension of the workpiece which should be computed. The Figure 1 below is the CAD models while figure 2 illustrates STL file formed. It can be seen in figure 2, the information needed in STL file is the triangle positions in an indexed list as calculation objects.

![Figure 1. CAD models](image-url)
Figure 2. STL file form

Referring to the explanations above, the principles of this method are:

3.1. **STL reader and positioning to coordinate system**

The STL files (ASCII format) contains the following data:

```plaintext
solid
  Facet normal---- ---- ----
  Outer loop---- ----- ----- 
  Vertex ---- ---- ----
  Vertex ---- ---- ----
  Vertex ---- ---- ----
  Ensloop
  Endfacet
  -------
  ----- 
  End
```

The content of STL file above are elaborated below:

1. **Solid, End Solid**, marks faceted models representation
2. **Facet normal**, provides information on the position of normal vector on a triangle
3. **Outer loop, end loop** are the looping of the coordinates of the vertices
4. **Vertex** is the point position of the triangle on the coordinate system.

The following is the data processing model used to manage the STL file which contains the data facet and vertices that form the triangle. The function of STL reader can be seen in psedocode below:
folder = 'folder name';
filename = 'file name';

stlpath = strcat(folder, '/', filename, '.txt');
triangles_csv = strcat(folder, '/', filename, '_t.csv');
vertices_csv = strcat(folder, '/', filename, '_v.csv');

if exist(triangles_csv) && exist(vertices_csv)
    disp(['Read from file...',triangles_csv,' and ', vertices_csv, '.']);
    T = csvread(triangles_csv);
    V = csvread(vertices_csv);
else
    [T, V] = stlreader(stlpath);
    csvwrite(triangles_csv, T);
    csvwrite(vertices_csv, V);
end

The Matlab software is used to compute the above function to get information and position of all triangles at coordinated positions. In addition to the STL read import function, here are two STL reads with m files available on the Matlab file exchange.

3.2 Generating of point clouds
By triangulating the workpiece, the length, width and height of the workpiece are calculated from triangle position against a three-dimensional plane. Location of the furthest triangle on X axis identifies the maximum value of x and the same thing shall apply to Y axis and Z axis. The workpiece dimension will be counted from X max to X min, for Y axis, from Y max to Y min, for Z axis, Z max to Z min.

In the point cloud creation process, which was explained in figure 3, the distances between point cloud towards the flat plane as well as vertical plane are determined by the density value. Initial formation of point cloud is carried out along a flat plane for example XY plane from the minimum value to the maximum value. The point cloud here is a virtual point construction to simplify calculation and it is used as initial reference at the work piece surface. As explained before, every point cloud on XY place at elevation z = 0 is also made on each altitude level (Z level) with the value between levels of density equal to density of point cloud. The making of point cloud on every Z level is a layering step on the feeding process, considering the depth of cut on the machining process.

![Figure 3. generated point cloud](image)

3.3 Slicing line method
By using ray triangle intersection method, by Moller & Trumbore[17], this process produces less memories in computation. This step is needed to compute the intersection of ray with a triangle and then to compute the barycentric coordinates of that intersection. Slicing method been applied in research[18]. Look at figure 4 below:

![Figure 4. slicing line with ray triangle intersection method[17]](image)

A ray with origin \( o \) and direction \( d \) intersects a triangle defined by its vertices, \( v_0, v_1, \) and \( v_2 \) and the ray intersects with a point at triangle areas by the equation of plane:

\[
d = ax + by + cz
\]

When \( a, b, c \) are coefficient forms, a vector of the plane is \( n = [a \ b \ c]^T \) so the equations are

\[
n \cdot x = d
\]

And

\[
x = [x \ y \ z]^T
\]

To calculate the intersection of the ray and the triangle, the equation above becomes:

\[
f(t) = o + t \cdot d
\]

and

\[
n \cdot f(t) = d
\]

\[
n \cdot [o + t \cdot d] = d
\]

\[
t = \frac{d - n \cdot o}{n \cdot d}
\]

The function and simulation result (figure 5) are shown below:

```matlab
function [flag, u, v, t] = rayTriangleIntersection (o, d, p0, p1, p2)
    Input:
    o : origin.
    d : direction.
    p0, p1, p2: vertices of the triangle.
    Output:
    flag: (0) Reject, (1) Intersect.
    u,v: barycentric coordinates.
    t: distance from the ray origin.
    epsilon = 0.00001;
    e1 = p1-p0;
    e2 = p2-p0;
    q  = cross(d,e2);
    a  = dot(e1,q);  % determinant of the matrix M
    if (a<-epsilon || a>epsilon)
        [flag, u, v, t] = deal(0,0,0,0);
        return;
    end;
    ```
3.4 Slicing line that cutting thru a triangle
This subchapter will explain how to get the intersection of the slicing line with a model (triangles). As described in the previous subchapter, the virtual point cloud was computed by utilizing of a ray triangle intersection method. The distance of point cloud was formed by the density value. This means the position of the point cloud as the object computation has been determined as the origin of the ray. The ray direction was determined by the direction of Z axis as well as by adding the value of \( k \) in the \((i, j, k)\) vectors. Pointed on the equation above, each slicing line will calculate of \((t)\) value where the value of \((t)\) indicates intersection with triangle. Next, each slicing line will look like a triangle intersection by iterations and will be stored in the data index. The steps to get the triangle to intersect with the slicing line as following:

```matlab
%% cutting_sl: find all SL that cutting thru a triangle.
%% This returns sub elements of points cloud 'sub_points'.
function sub_points = cutting_sl(tri_vertices, points_cloud)
    mins = min(tri_vertices);
    maxs = max(tri_vertices);
    density = points_cloud(1,2,1) - points_cloud(1,1,1);
    min_x = points_cloud(1,1,1);
    from_x = mins(1);
    sl_x_from = ceil( (from_x + density - min_x) / density );
    to_x = maxs(1);
    sl_x_to = floor( (to_x + density - min_x) / density );
    min_y = points_cloud(1,1,2);
    from_y = mins(2);
    to_y = maxs(2);
    sl_y_from = ceil( (from_y + density - min_y) / density );
    sl_y_to = floor( (to_y + density - min_y) / density );
    sub_points = points_cloud( sl_y_from:sl_y_to, sl_x_from:sl_x_to, : );
end
```

3.5 CBV determination
According with the previous description, for each triangle intersected by the slicing line will be stored at the database and will be evaluated for CBV area grouping. The grouping method is illustrated in figure 6 below. The basic concept is to classify the location of each point cloud based on position at the models. The point cloud which is on CBV area is located between two solid models of a slicing line whereas the point cloud as OBV area is located outside of CBV area and beyond solid models. The point cloud at CBV area can be seen in figure 7 below.
4. Result And Discussions

In this study, the CC points under the CBV area were determined by using virtual points in the form of point clouds. The distances among the point clouds were used as the base value, or what is known as the density value. Each point cloud became the reference to determine the grouping: CBV or OBV area. Meanwhile, in the previous study, the CBV and OBV areas were determined before the point clouds were made and therefore, there were steps previously done to determine the machining area which was simplified in this study.

**Table 1.** the difference between the previous and the new study.

| Steps for determination | Previous method | New method |
|-------------------------|----------------|------------|
| 1  STL reader           | yes            | yes        |
| 2  Plot normal vector   | yes            | no         |
| 3  Calculate of average point | yes        | no         |
| 4  Bucketing method     | yes            | no         |
5. Conclusion

Compared to the previous concept, this method is simpler and more suitable for rough machining because the points used as computation objects are virtual points with certain distances so that the result data is still rough. The previous method is more suitable for finishing process because main point of object computation uses triangles at surface curves.

References

[1] B. L. G. K. J. P. Kruth, "Development of a Five axis Milling Tool Path Generation Algorithm based on faceted model," 2003.
[2] S. C. Park and Y. C. Chung, "Tool-path generation from measured data," Computer-Aided Design, vol. 35, p. 9, 2003.
[3] H.-T. Young, L.-C. Chuang, K. Gersehwiller, and S. Kamps, "A five axis Rough Machining Approach for a Centrifugal Impeller," The International Journal of Advanced Manufacturing Technology, p. 16, 2003.
[4] H. T. Young, L. C. Chuang, K. Gerschwiler, and S. Kamps, "A five-axis rough machining approach for a centrifugal impeller," The International Journal of Advanced Manufacturing Technology, vol. 23, pp. 233-239, 2004.
[5] M. Balasubramaniam, P. Laxmiprasad, S. Sarma, and Z. Shaikh, "Generating 5-axis NC roughing path directly from a tesselated representation," Computer-Aided Design, vol. 32, p. 17, 2000.
[6] Y.-S. Lee, "Admissible tool orientation control of gouging avoidance for 5 axis complex surface machining," computer aided design, vol. 29, 1997.
[7] K. C. Cha-Soo Jun, Yuan-Shin Lee, "Optimizing tool orientation for 5-axis machining by configuration space search method," computer aided design, vol. 35, p. 18, 2003.
[8] G. Kiswanto, H. H. Sutrisno, and J. Istiyananto, "Development of Initial Tool Orientation Method At Close Bounded Area for 5-Axis Roughing Based On Faceted Models," ICMM, vol. 3, 2016.
[9] G. Kiswanto, B. Lauwers, and J.-P. Kruth, "Gouging elimination through tool lifting in tool path generation for five-axis milling based on faceted models," Int. J Adv Manuf Technol, vol. 32, p. 21, 2007.
[10] M. Balasubramaniam, S. E. Sarma, and K. Marciniak, "Collision-free finishing toolpath from visibility data," Computer-Aided Design, vol. 35, p. 16, 2003.
[11] J. P. K. B. Lauwers, P. Dejonghe, R. Vreys, "Efficient NC programming of Multi axis Milling Machine Through the intregation of tool path Generation and NC simulation," computer aided design, 2000.
[12] A. Krimpenis and G.-C. Vosniakos, "Optimisation of Multiple tool CNC Rough Machining of a Hemisphere as a Genetic Algorithm Paradigm Application," Int. J Adv Manuf Technol, vol. 20, p. 8, 2002.
[13] W. TC, "Visibility maps and spherical algorithms," Computer Aided Design, vol. 26, p. 10, 1994.
[14] K. Suthunyatanakit, E. L. J. Bohez, and K. Annanon, "A new global accessibility algorithm for a polyhedral model with convex polygonal facets," Computer-Aided Design, vol. 41, p. 14, 2009.

[15] Y. Li and M. C. Frank, "Computing non-visibility of convex polygonal facets on the surface of a polyhedral CAD model," Computer-Aided Design, vol. 39, pp. 732-744, 2007.

[16] G. Kiswanto. and A. Y. T. Panuju, "Development of closed bounded volume(CBV) grouping method of complek faceted model through CBV Boundaries identification.," IEEE, vol. 3, p. 5, september 2010.

[17] T. Moller and B. Trumbore, "Fast MinimumStorage RayTriangle Intersection," 1997.

[18] S. C. Park, "Sculptured surface machining using triangular mesh slicing," Computer-Aided Design, vol. 36, 2004.