Computing Cutter Engagement Values in Milling Tessellated Free-Form Surfaces

Zhiyang Yao¹, Ajay Joneja²
¹(corresponding author) zyyaomae@gmail.com, Hong Kong
²joneja@ust.hk, Dept of IELM, HKUST, Hong Kong

Abstract

High speed milling (HSM) has great potential use in die/mold cutting, but traditional machining plans do exploit HSM capabilities effectively. An important consideration for in HSM is to limit cutting force variations, and one way to do so is to reduce Cutter-Workpiece engagement (CWE) variations. CWE is measured as the area of the tool instantaneously engaged with the part. Estimating CWE as a function of the tool path requires repeated, expensive computations. This paper develops algorithms for a discretized computational model to make CWE computations for arbitrary shaped parts.

1. Introduction and Background

HSM employs machine tools operating at high spindle speeds and feed rates to remove material faster while maintaining the quality of part finishing with high precision, improved surface quality and high efficiency [1]. It has great potential use in fabrication of a wide range of automotive, electrical, and plastic molding parts [2]. To achieve high speed during milling operations, the cutter should have a constant chip load with a small step-over and very high feed rates, otherwise the likelihood of unexpected and undesirable tool breakage increases [3]. Cutter engagement is a measure that describes what portion of the cutter is actually involved in machining at a given instant of time [4], and the CWE value is critical in designing cutter paths and controlling machining operations. For complex shaped parts, CWE values change along the cutter path – but for HSM it is preferable that CWE must stay below a safe upper bound throughout the cutting.

The importance of CWE has been recognized in the past [3-7]. A good computational model for CWE can be used in two possible ways. Firstly, it can be used to predict potential regions of large changes of cutting forces [4, 6-7, 8]. This in turn can be used for either controlling the machining parameters, or adjusting the cutter path. Secondly, it can be used as a diagnostic tool – estimating whether tool failure at some point in the machining was caused by tool wear, or machining forces related issues; in such cases, it can be used to locally modify the cutting plan for future runs of the same part. Computationally, a voxel-based approach is used by Fusell et al [8]. In a more recent work, a feature-based approach is adopted by Yip-Hoi et al [7,9]. The work considers 2.5D milling operations using flat end milling cutters. As cutting progresses, the shape of the workpiece changes due to the elimination of 2.5D swept volumes. A CAD system along with Boolean operations are used to estimate different geometric cutter engagement features (ceF’s), namely different types of tool-part engagement geometry, such as uniform depth of cut, stepped depth of cut etc. These are used to estimate cutting forces. For simpler tool geometry, Yao et al [6] and later, Gupta et al [4] developed some geometric methods for CWE estimations; they only consider 2.5D milling operations and flat-ended milling tools. Similar efforts in other processes, e.g. boring using a single point tool have been reported [10].

For 2D milling, there are numerical and analytical methods for finding CWE values [4, 5]. For 3D operations, it is possible to use CAD-style Boolean operations [11-14, 15], however they require extensive computational work load. Thus, approximation methods like the z-map method are still widely-used [16]. Another approach is to use approximation models of the part as well as the tool. In this paper, we show that for several commonly used tool shapes, efficient algorithms exist to allow the use of a hybrid approach: namely, using exact model of the tool to intersect with a tessellated model of the part.
2. Definitions and Problem Formulation

We will assume CWE to be defined as the area on the envelop surface of a cutting tool that is in contact with the workpiece at a given instance during cutting. The point set representing the tool envelop is denoted by \( T \), and the workpiece as \( W \). The instantaneous machined volume is defined by \( M_v = T \cap W \), and cutter engagement = area of \( b(M_v) \cap b(T) \), where \( b \) denotes the boundary.

The intermediate part geometry (i.e. the current state of the workpiece) is given in the form of a triangulated model. For computing Booleans, we construct a half wing-edged graph [17] data structure from it. The list of vertices is stored in a data structure appropriate for faster range-searching, e.g. a 3D binary-tree. The main idea of the cutting simulation algorithm is simple:

Given the intermediate part geometry, the initial position of the tool such that no part of the workpiece lies in its interior, and the final position (location, orientation) of the tool, e.g. as it moves a small distance along a tool path, compute the CWE area and the updated part geometry = initial part geometry - tool in final position.

The method is discrete: we only compute the CWE at the end position of the tool, and not over its entire swept path at each step. Suppose \( T_i \) and \( T_f \) are the point sets of the tool in its initial, and final positions respectively; \( p(t) \) describes the motion of the tool during this step; the swept volume of the tool along the path is \( S_v \). We define a facet \( f \) to be ‘cut-through’ if \( f \cap T_i = \emptyset \); \( f \cap T_f = \emptyset \); and \( f \cap S_v = f \).

We will assume that the CWE is computed at fairly small step-sizes, and therefore we ignore possible ‘cut-through’ triangles. To successfully generate this output, we need two robust computations: (i) computing the CWE area for different types of tools, and (ii) updating the part geometry after each step. Our main contribution in this work is the development of analytical techniques to compute the CWE, which is discussed in sections 3.1-2. In section 3.3, we briefly discuss how the part geometry is updated – omitting the details since the Boolean operation required for this function is known [15,10].

3. Methodology

As the tool engages some part of the workpiece, some of the facets that are initially outside the tool envelop will now move to its interior. We consider the following tools: flat-, ball-end or tapered mills (numerical techniques must be employed for bull-nosed cutters, and therefore our approach does not extend to them). The inclusion/intersection tests are based on standard geometry, as summarized below:

3.1. Computing the intersection of the part geometry with the tool.

The algebra of surface-triangle intersection is well known. Below we summarize the computations to show that closed form solutions exist for intersecting triangular facets with several common tool shapes – flat-ended, conical, or ball-ended mills.

![Figure 1. plane: z=0; cylinder: \( x^2+y^2 = R^2 \); sphere: \( x^2 + y^2 + z^2 = R^2 \); cone: axis along Z.](image-url)
(a) Computing intersection point(s) of triangle edge with tool surface

Denoting the vertices of the triangle edge by position vectors \( a = [x_a, y_a, z_a] \), \( b = [x_b, y_b, z_b] \), the edge lies on the line \( t a + (1-t)b \).

Again, assuming the shape element (and triangle edge) to be transformed to the standard position, the intersection point(s) are computed as:

**Line-plane:** \( tz_a + (1-t)z_b = 0 \), giving the parameter at intersection point \( t^* = z_b / (z_b - z_a) \)

**Line-cylinder:** \((tx_a + (1-t)x_b)^2 + (ty_a + (1-t)y_b)^2 = R^2\), which gives the following quadratic equation for \( t \):

\[
t^2(x_a^2 + y_a^2 - x_b^2 - y_b^2) + 2t(x_a x_b + y_a y_b) + (x_b^2 + y_b^2 - R^2) = 0.
\]

**Line-sphere:** \((tx_a + (1-t)x_b)^2 + (ty_a + (1-t)y_b)^2 + (tz_a + (1-t)z_b)^2 = R^2\), which gives the following quadratic equation for \( t \):

\[
t^2(x_a^2 + y_a^2 + z_a^2 - x_b^2 - y_b^2 - z_b^2 - 2x_a x_b - 2y_a y_b - 2z_a z_b) + 2t(x_a x_b + y_a y_b + z_a z_b) + (x_b^2 + y_b^2 + z_b^2 - R^2) = 0.
\]

**Line-cone:** Again, we represent the line by \( a + tb \), and the cone in parametric form as:

\[
x = u \tan \theta \cos \phi \\
y = u \tan \theta \sin \phi \\
z = u
\]

Substituting the coordinates of a point at parameter \( t \) into the above, we get a quadratic in \( t \):

\[(x_a + t x_b)^2 + (y_a + t y_b)^2 = u^2 \tan^2 \theta = z^2 \tan^2 \theta = (x_0 + t z_b)^2 \tan^2 \theta.
\]

(b) Intersection of (triangular portion of) a plane with the tool surface

To compute the CWE area, an also to update the part geometry after the cutter moves a small distance, we also need to compute the curves of intersection between the part and tool. The base components for this are plane-cylinder, plane-sphere, and plane-torus intersections, which are computed as follows. Again, we transform the elements into standard position as shown in the Figure 1.

**Plane-Cylinder:**

Let the surface of the cylinder be \((r \cos \phi, r \sin \phi, v)\), and the plane in implicit form be \(z = Ax + By + D\); substituting for \( z \), we get the intersection curve in parametric form as \((r \cos \phi, r \sin \phi, A r \cos \phi + B r \sin \phi + D)\).

**Plane-Sphere:**

The intersection is a circle that is easily derived by transforming the plane (together with the sphere) such that it is parallel to the XY-plane, followed by a translation that takes the center of the sphere to the origin. This yields the equation of the intersection as \((a \cos \phi, a \sin \phi, b)\), where \( b \) is the shortest distance of the plane from the center of the sphere, and \( a^2 = (r^2 - b^2) \).

**Plane-Cone:**

The intersection of the plane of the triangle with the cone face of a tapered mill yields a conic section, familiar from basic geometry.

3.2. Computing the cutter workpiece engagement

3.2.1. CWE area of planar region:

Each CWE loop is a region bounded by \( n \) sides, each of which is either a straight line or a circular arc. If there are no circular arcs, then its area can be computed in \( O(n) \) time using any standard computational geometry algorithm [18]. Below we discuss the case when there are one or more circular edges.
Figure 2 shows a possible planar CWE region (blue edges), where two edges are circular arcs. Consider arc $a-b$ in the Figure. We extend the radial line $Oa$ to a vertex $A$, and $Ob$ to vertex $B$, such that a straight line $AB$ does not intersect arc $a-b$. We replace arc $a-b$ by the edges $aA$, $AB$, and $Bb$ in the $n$-gon. Repeating this process for each arc in the CWE yields a polygon with at most $O(n)$ edges, and whose area can now be computed in $O(n)$ time. It is thus easy to compute the actual CWE area, since we can easily compute the areas of triangle $OAB$ and sector $Oab$. This process can be carried out for each arc that subtends less than $180^\circ$ from the center. Larger arcs can be sub-divided into at most two components, and replaced by a sequence of four edges to apply the same method.

3.2.2. CWE area of cylindrical region:
Each edge of any CWE loop on a cylindrical region is formed by a planar section, and is therefore either an elliptical or circular arc whose length is easily computed. By splitting open and unfolding the cylinder, we obtain a planar polygon. The coordinates of each vertex are also readily obtained by first writing the coordinates of each vertex on the cylinder in a cylindrical coordinate frame with $z$-axis lying along the cylinder axis, and then transforming these coordinates into an equivalent Cartesian frame. All these conversions take $O(n)$ time. Subsequently, computing the area of the planar polygon takes $O(n)$ time. Thus the CWE area on a cylinder is obtained in $O(n)$ time.

3.2.3. CWE area on spherical region:
The ball-ended milling cutter has a hemi-spherical bottom. The intersection of the part with this hemisphere is one or more simple spherical polygons. Each such polygon is a closed loop made up of circular edges, which may be great-arcs or small-arcs. Here we discuss how to compute the area of such spherical polygons.

(i) 1-sided polygon
This polygon is formed when the sphere is cut by a single plane, and the area is given by $2\pi r^2 (\cos (\sin^{-1}(d/r)))$, where $d$ is the distance of the plane from the center of the sphere, and $r$ is the radius.

(ii) 2-sided polygon
If both edges are great-arcs, then the region is a lune of area $= 2R^2a$, where $a$ is the lunar angle.
Figure 3. (a) Lune with lunar angle = a and surface area = 2R²a  (b) Semi-lune

If the two edges are small arcs, then we generate a great arc passing through the two vertices, and yielding two semi-lunes. The area of a semi-lune is computed as follows; consider semi-lune between a great arc BCₙ (dashed line) and a small arc BCₙ (solid line) in Figure 3. Let P be the pole of the small circle through BC. Draw the great arcs PB, PC. Let the co-latitude of B and C = a, and the interior angle BPC = b; then the area of the small triangle PBCₙ is given by b(1-cos(a))R². Further, we know from Girard’s theorem that the area of the great triangle PBCₙ = (Σθₙ - π)R², where θₙ are the interior angles of the great arc triangle PBCₙ. Thus the area of the semi-lune = (Σθₙ - π - b(1-cos(a)))R². Note that this area may be negative or positive, depending on whether the semi-lune falls outside or inside the great triangle PBC.

(iii) Polygon with 3 or more sides.

Such a polygon will have sides that are either great arcs or small arcs. If the polygon is ‘well-behaved’, like the red-colored one in Figure 4, then the area is computed in O(n) time, as indicated in the figure. However, it is possible that one or more semi-lunes formed during such a computation may overlap (partially) with the polygon. In the general case, such self-intersections can be accounted for using some extra book-keeping. The different cases are easier explained as follows.

Consider a Riemann map of the hemisphere. This creates a planar map of the polygon, called a pocket, which (a) is a simple n-sided loop, (b) maps all great arcs to lines, and (c) maps small arcs to circles. We shall convert this pocket into a polygon, by replacing each circular arc, in sequence, by a line segment joining its end points (shown as dashed lines in Figure 4). Such a line segment may or may not intersect with an edge of the pocket. Consider the no-intersections cases first. If the line lies entirely inside the pocket, then we add the area of the corresponding semi-lune, and the map has one arc replaced by a line. Required area = area of simplified pocket + area of clipped semi-lune. If the line lies entirely outside the pocket (Figure 5(b), (c)), again the arc is replaced by a line in the map. Required area = area of resulting simplified pocket - area of clipped semi-lune. In the case shown in Figure 5(c), we just take the absolute value of the area computed. There are two types of cases to account if the simplifying line segment intersects with the pocket. In the example shown in Figure 5(d), the dashed line for clipping arc 2 intersects with an edge (arc 1) of the pocket. However, if we first eliminate arc 1, followed by arc 2, then our previous book-keeping method correctly compute the area. A simple iterative loop can identify the sequence in which we can eliminate all such arcs from our pocket. Finally, we handle the arcs that still remain. In such cases (see Figure 5(e)), the simplifying line will result in a non-simple n-gon. In this case, we compute the intersection point, and compute the area of the resulting simple polygons separately. Note that the simplifying edge is broken into two (or more) parts, one for each simple polygon. We compute the area of each simple polygon individually. Those polygons where the simplifying edge lies inside the original pocket, the area has the same ‘sense’ (i.e. ‘+’) as the area of the clipped semi-lune, and the remaining ones have the opposite sense (-ve). Once all arcs are eliminated, we are left with (a set of) polygons on the Riemann map, which are mapped back to the sphere and their areas computed using the formula in Figure 4.
Polygon area = \( \left( \sum \theta_i - (n-2) \pi \right) R^2 - \sum E_L(i) \),

Figure 4. Computing area of a simple polygon on a sphere.

Figure 5. Different cases of computing polygon area on a sphere.

3.2.4. CWE area on cone:
For tapered-side milling cutters, the intersection of each intersecting triangle is a conic section, and therefore each loop of the CWE area is a conical polygon bounded by a sequence of edges that are segments of a line, circle, ellipse, parabola, or hyperbola. The area of any such closed n-gon can be computed exactly. To do so, we first project the n-gon to a plane coincident on the base of the cone; we call this projection as the axial projection below.
Observation 1. Axial projection of a conic section (circle, ellipse, hyperbola, parabola, line) is also a conic section of the same type (or, in the degenerate case, a straight line).

Observation 2. Suppose a sequence of (conic-section) edges form a simple loop on the surface of a right cone (i.e. no self-intersections inside, or between any pair of edges). Then the axial projection of the curve is also a simple loop. This is readily proved: if the projection contains an intersection point, project this point upwards in the axial direction to where it meets the cone surface. It follows that that the two intersecting edges must also meet at this point, which contradicts the assumption that the original loop is simple.

Observation 3. Consider a right cone with cone angle $\theta$ and radius $r$. Suppose a region on the surface of the cone has area $A$; then the axial projection of the area on the base will have an area $= A \sin \theta$.

The CWE region (a set of simple n-gons bounded by conic sections on the cone surface) is first generated by plane-cone intersections followed by linking of the intersection curves. For each loop of the CWE region, each edge is projected onto the base along the cone axis. The area of the resulting planar shape is computed by following the same approach as for the spherical case detailed in section 3.2.3 with one difference: for each edge, the segment area, namely the area between the curve and the line joining the vertices, is computed by integration of the corresponding curve. Since each edge can be handled by explicit integration, therefore a constant time effort is required for each edge: the total run time for computing the area is $O(n)$ in the number of edges of the CWE n-gon. Thus the algorithm run time depends mainly on the efficiency of the clipping and range search on the triangles selected for intersection with the cone. Figure 6 shows a schematic.

![Figure 6. Computing CWE on tapered tools](image)

3.3. Updating the workpiece geometry

After the cutter moves, the workpiece geometry needs to be updated for the next iteration. This update can be computed in several ways. We can generate the envelop surface of the tool over its motion, then perform a Boolean subtract of this shape from the workpiece shape; the resulting geometry must then be tessellated. This approach is accurate but expensive. We propose two simplifications: firstly, since the CWE must be computed over small motions (or steps), we only use the final position of the tool for the Boolean subtraction. Secondly, we replace the “subtract $\rightarrow$ tessellate” step by a “tessellate $\rightarrow$ subtract” operation. This allows for a much faster geometry update operation. We can initially impose a tessellation on the tool according to a specified tolerance; for all subsequent Boolean operations, we only need to transform this shape to the required position. Secondly, only a subset of the triangles in the Boolean intersection must be considered for updating the part geometry.
The approach is discussed below, but only briefly, since such Boolean operations have already been fairly well studied in the past, see for example [17,15].

Based on the level of resolution desired, a triangulation mesh is superposed on the tool surface. After the intersection loops are discovered as per section 3.2 above, those workpiece triangles which are completely inside the tool are deleted. The ones that intersect the tool are updated as follows: (1) Any tool-triangle that is completely inside the CWE region will be added to the list of triangles in the workpiece; (2) All triangles that contribute to the CWE edge are modified, joining the remaining workpiece triangles to the newly added triangles contributed by the tool surface. Details of the process are omitted for brevity.

4. Example

Figure 7 shows the geometry of a mold cavity that will be used for production of the top body of a toy car. It must be machined from a solid rectangular block, and since the part has a deep cavity, the machining is planned in several layers; the first such layer is shown in Figure 8.

For illustration, we focus on the CWE values for a small but interesting region of the cutting, as shown in the Figure 9. Assume that the top layer to be machined has been rough-milled using a flat end milling cutter, leaving approximately a 1mm offset zone from the check-surface boundary to be removed by finishing operations. As the cutter moves along this path with constant feed rate, the CWE is computed at steps of 1mm each. Figure 9 shows some intermediate positions, and Figure 10 shows a plot of the variation in the CWE values as the tool moves along this path.
5. Summary and Future work

This paper presents a framework for a system to compute changes of CWE along planned cutter paths in milling. We show that for several common tool shapes, an efficient solution is to use a hybrid approach, namely by using the exact tool envelop geometry and tessellated part models. There are several potential improvements. Firstly, we estimate the CWE to be proportional the tool-workpiece contact area; in reality, the force variations are likely related also to the local feed direction. The simplest model to account for this is to only consider the forward facing half of the milling tool (with respect to the local feed direction). Another aspect that we have not directly considered in this paper is robustness – repeated intersection operations can eventually cause the part model to develop topological errors due to stacked numerical inaccuracies. One reason for this is the increasing number of small triangles generated as the cutter is swept along tiny motions steps along the path. An alternate approach is to compute the actual swept volume of the tool along a sequence of a few motion commands, and subtract this volume from the part using an exact model. Once the CWE computation is completed for these motion statements, the (unstable) intermediate part model can be replaced by the corresponding exact shape with a new tessellation.

6. Acknowledgements

Part of this research was supported by the CERG Grants HKUST/GRANT 614205 and CUHK/GRANT417906 for which the authors are grateful to RGC, Hong Kong.
References