Automatic detection of manufacturing issues in CAD parts for DFM analysis

Sergey Slyadnev[1], Andrey Voevodin[1]

[1] *Quaoar Studio LLC, Nizhny Novgorod, Russia*

Abstract

This paper discusses some computational principles behind automatic manufacturability tests for CAD models of machined parts. The presented approach is a combination of automatic feature recognition supplemented with ray casting for accessibility, thickness and clearance checks. Most of the discussed approaches are applicable for different production methods, such as milling and turning (lathing) as they are purely geometric in nature. We distinguish between meshless and mesh-based methods for the analysis of manufacturing issues in a CAD part. For the mesh-based methods, two types of surface triangulation are employed: a sparse non-uniform triangulation having a relatively small number of triangles representing the geometric boundary of a part, and a dense grid covering the input part with uniform triangulation. The latter kind of mesh is constructed using quadtree-based subdivision of the UV spaces of the corresponding curvilinear trimmed surfaces.

Keywords: Computer-aided design, design for manufacturing (DFM), feature recognition

1. Introduction

Manufacturing effort is the measure of the time, care, and challenge needed to manufacture a design. Such effort is one of the key metrics in the *design for manufacturability* (DFM) discipline that is well introduced through the context of DFX ("Design for X") in the seminal paper by AT&T authors in [1]. As mentioned by Gupta S.K. [2], DFM simultaneously considers design goals and manufacturing constraints in order to identify and alleviate manufacturing problems while the product is being designed; thereby reducing the lead time for product development and improving product quality.

In this paper, we present a set of manufacturability analysis methods that can be used during the design stage or at a post-processing stage conducted by manufacturers when they receive a part for fabrication. As a result of DFM diagnostics, the input part is either verified or, if not, the identified local design issues are reported, both analytically and graphically. It is assumed that a problematic design has to be changed to eliminate all the detected issues, or, at least, the manufacturer can draw the designer's attention to the problematic zones and foresee deviations of the fabricated part from its as-designed shape.

Another problem tightly related to DFM is assessing the manufacturing costs for a part. The presence of particularly difficult features, such as variable-radius fillets or non-prismatic pockets, does not make a part infeasible but requires more care and machining time and is consequently more expensive. It is therefore beneficial to include DFM and price estimations into the design process, so that a designer can always keep track of manufacturability aspects and reduce, if not eliminate, *create-test-redesign* loops when a part is handed to production.

Speaking the DFM language requires thinking at a higher level of abstraction. The initial CAD design is reviewed from the manufacturing standpoint in the DFM context. The following list enumerates some typical questions often addressed to a DFM analysis system:

1. Can this sharp corner be fabricated?
2. Is a specific feature accessible by a tool (mill or drill) from a specific direction?
3. Is a specific feature thick enough?
4. Are there any slots too deep or too narrow in the part?

Email addresses: `sergey@quaoar.pro` (Sergey Slyadnev), `andrey@quaoar.pro` (Andrey Voevodin)
URL: `http://analysissitus.org` (Sergey Slyadnev)

5. Is several-pass 3D milling required to finish some part features?
6. How to assess the machining time and number of setups required to fabricate a part?

The number of checks and their complexity differ depending on the production method. An additional complication is due to the fact that there is usually more than one way of manufacturing the same part. In their recent technical note, A. McKay and A. Pennington [3] give an example of "coating" for a simple shape that illustrates how a single design can be comprehended in different ways, depending on the context. Still, many checks have much in common, and therefore can be based on the same geometric core tailored to subtractive machining processes. We do not address the emerging additive manufacturing technologies that would require specific DFM rules and different input data (e.g., STL files instead of curvilinear boundary representation). Moreover, the following discussion is limited to single parts, so we exclude assemblies and their relevant DFM checks from consideration in this paper.

The paper is organized as follows. Section 2 overviews the previous work in the DFM field, embracing both feature recognition methods and manufacturability tests per se. Section 3 is our contribution to the DFM field. We present a set of checks that can be implemented as individual functions in the corresponding software package. Our focus is on manufacturability tests, so we deliberately exclude feature recognition methods from consideration. It should be noted, though, that feature recognition should be considered as an integrated part of our approach. We do not cover the recognition algorithms for drilled holes, fillets, chamfers, engravings, threads, etc. because such discussion would shift our focus from DFM to generic feature identification. Section 4 describes our approach to building a dense surface triangulation over curvilinear trimmed CAD faces. A dense surface mesh is required for conducting accessibility tests based on ray casting. The same approach is adopted for clearance and thickness analysis. Section 5 concludes the paper.

2. Previous work

Validating CAD parts from a manufacturing point of view has been an actively researched problem for decades. Most of the studies on DFM methods imply feature decomposition of the part CAD model, then associating manufacturability evaluation with each feature.

W. Regli [4] extracted alternative feature-based interpretations for a CAD part using Material Removal Shape Element Volumes (MRSEVs). The author treats features as parameterized solid objects reconstructed from the initial part and stock using a set of functions of a chosen modeling system (commercial ACIS in their case). Although W. Regli emphasizes the importance of feature recognition for manufacturability analysis, his work does not go further with the identification of realistic design issues that are of high interest in machine shops.

O. Kerbrat et al [5] propose an octree decomposition method for automatic assessment of manufacturing difficulties. The output of their method is a scalar map representing potentially problematic zones over the CAD surfaces. No feature recognition is employed, so this method does not output any analytical properties of the machined features. Still, we think that the method by O. Kerbrat et al. can be used as a part of a more sophisticated DFM analysis pipeline. For example, the same octree decomposition, if done adaptively, can potentially help in estimating rough milling time for a part.

More references to the previous works in the DFM field are given below in the corresponding sections.

3. Manufacturability checks

In what follows, we employ the attributed adjacency graph G [8] to reason about topological relationships between faces and basic geometric properties over their edges, such as the convexity of dihedral angles.

3.1. Check sharp corners

Checking for sharp corners is arguably the simplest manufacturability test one can conduct. The algorithm 1 outlines the procedure. The idea is to find all vertices that have at least three concave incident edges with sharp dihedral angles.

Such vertices are commonly considered inaccessible for a mill because of the inherent tool radius, leaving undercuts near concave sharp corners. It should be noted, though, that false positives for sharp corners are often identified for features that are not supposed to be milled. Therefore, to make this DFM check more reliable, it should exclude all vertices lying on the features that are not milled (engravings, modeled threads, etc.). To cope with such redundant features, the algorithm 1 accepts U as a universum for all vertices under consideration.

Algorithm 1 Check sharp corners

```
1:  procedure CheckCorners($S$, $U$)          ▷ $S$ is a CAD model, $U$ is a universum
2:      $R \leftarrow \emptyset$                              ▷ Result set
3:      $G \leftarrow$ initialize AAG from $S$
4:      for each vertex $v \in U$ do          ▷ Iterate over all vertices in the universum
5:          $E \leftarrow$ incident sharp edges of $v$
6:          $n \leftarrow 0$                          ▷ Number of sharp concave edges
7:          for each edge $e \in E$ do             ▷ Iterate over all incident edges
8:              $\alpha \leftarrow$ convexity of $e$
9:              if $\alpha$ is concave then
10:                 $n \leftarrow n + 1$
11:             end if
12:         end for
13:         if $n \geq 3$ then
14:             $R \leftarrow R \cup v$                    ▷ Add to the result set
15:         end if
16:     end for
17:     return $R$
18: end procedure
```

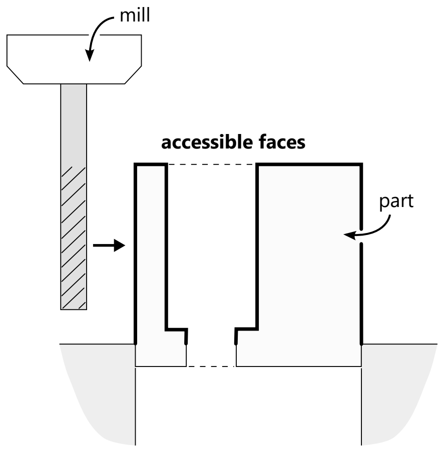

Figure 1. Accessible zones (bold lines) over CAD faces for the given tool direction.

3.2. Check face accessibility

To machine a surface geometry, it must be reachable by the tool cutting surface. This means that there is an orientation (setup) of the part and the tool where the normal vector of the cutting surface is parallel to the normal vector of the part surface. Fig. 1 illustrates the reachability of faces with respect to a square end mill (shown in 2D for simplicity).

Accessibility analysis has found its way to commercial CAD packages. As of 2022, such a function is available in Autodesk Fusion360, where it works in real-time mode. Because of the inherent simplicity of accessibility analysis and its similarity to ray tracing, GPU-based implementations look reasonable. R. Khardekar et al. [6] present an approach for checking demoldability of casted solid parts using GPU. Their approach, just like ours and the approach of Fusion360, is essentially a visibility test. Thus, determining whether a part is castable in a given direction reduces to checking for invisible facets when the object is looked at from the mold removal direction. The same problem is solved by A. Surti and N. Reddy [7] using silhouette curves constructed for precise B-rep models.

Our algorithm for accessibility checks is outlined below as a sequence of steps. The algorithm is performed for all the input directions that are supposed to be the principal machining directions. If a single face f is detected as an accessible face in at least one direction, it is deduced to be accessible in general, so

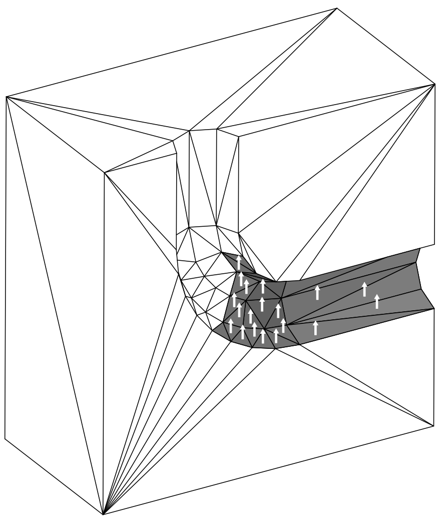

Figure 2. Ray casting for face accessibility test.

it goes to the result. The choice of the input directions determines the type of the machine, e.g., 3-axis or 5-axis. It should be noted that for the 5-axis machines, additional axes extracted by feature recognition are to be used (axes of holes, normal directions for pocket bottom faces, etc.).

Algorithm 2 Check face accessibility

```
1: A part is tessellated with a dedicated surface meshing algorithm (see Section 4) to obtain a piecewise
   approximation of the initial curved geometry. The values of the linear and angular deflections required
   by the tessellation algorithm are selected automatically. Each triangle of the resulting mesh stores the
   corresponding CAD face ID as a back reference.
2: The BVH structure is constructed for fast ray-triangle intersections.
3: The algorithm accepts a tool direction (axis) and associates with each triangle the reversed direction as
   a ray to emit. To cope with possible mesh inaccuracies and round-off errors, the ray source is slightly
   shifted in the direction of the local normal vector at the corresponding facet.
4: For each ray, the intersection test is conducted (Fig. 2). If a facet happens to be occluded, it is recursively
   subdivided with the limited subdivision depth. The recursive subdivision technique allows us to slightly
   refine the large facets contributing to the occluded regions without modification of the initial mesh. The
   efficient ray-box intersection is done accordingly to the paper by A. Williams et al. [9]
5: For all faces, the total "score" of tests, hits, and void intersections is counted. If the number of
   intersections over the total number of tests is high enough (e.g., 95%), the face is considered non-
   accessible.
6: All occluded facets with their possible subdivisions are returned as a result.
```

Let M be the surface triangulation covering the model boundary ∂S. Fig. 3 illustrates a realistic model having three features partially occluded. The model is tested from six directions: $D = \{\pm X, \pm Y, \pm Z\}$. The occluded sets $B \subset M$ and $C \subset M$ are the drilled holes that are not aligned with the principal directions D. The feature region $A \subset M$ is a pocket feature that is partially occluded by the opposite wall. While the regions B and C might require extra setups (thus increasing total machining time) or a 5-axis machine, the region A is likely to be redesigned.

3.3. Check edge accessibility

The edge accessibility check is done for the sharp concave edges exclusively. The concavity is taken from the AAG attributes without recomputing the dihedral angles. Every edge is turned into a polyline by uniform discretization (Section 4). For each point, the algorithm computes local "border trihedron" axes [20].

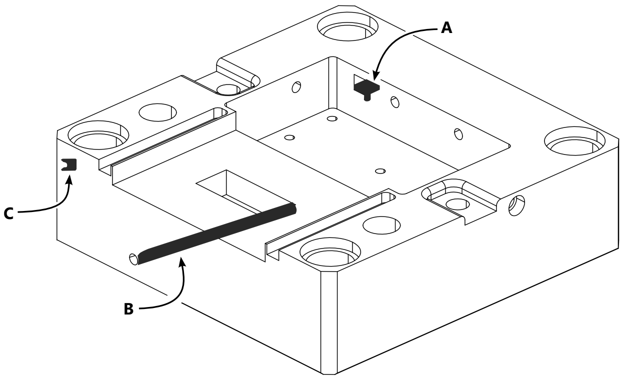

Figure 3. Inaccessible facets (regions A, B, C in shaded dark grey) for six principal directions on a test part.

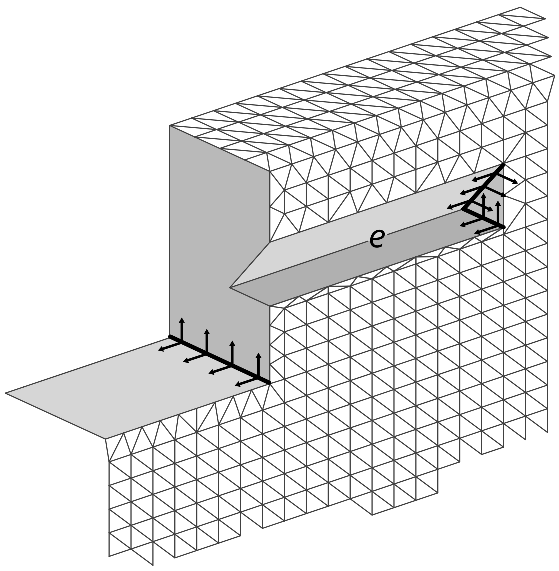

Figure 4. Edge accessibility check by ray casting.

If a concave sharp edge e has a dihedral angle less than 90 degrees, it is labeled "inaccessible" without a consequent accessibility test (e.g. edge e in Fig. 4). If not, a bunch of rays is emitted from the edge along its neighboring faces f_1, f_2. If both rays emitted for f_1 and f_2 hit an obstacle, the corresponding segment of an edge is marked "inaccessible" (Algorithm 3).

3.4. Check thickness

The thickness of individual walls and ribs is important for calculating the allowable stresses and strains of a machined part. Two methods for thickness calculation are widely adopted in CAD/CAM/CAE applications: *ray-based* method and *sphere-based* method. The ray-based method measures the thickness value at each point by casting a ray orthogonally to that point until the intersection with the opposite surface element. The sphere-based method attempts to inscribe spheres into the solid body instead.

While the ray-based method is easier to conduct, the thickness distribution obtained with that method is ambiguous. The point **P** in the Fig. 5 can potentially get two different thickness values: d_1 or d_2, depending on how the measurement is done. For the sake of simplicity, we use a convention that the thickness value at any point **P** is measured by shooting a ray from this point (d_2), although one could argue that the better value is d_1 from the incoming ray.

In contrast, the sphere-based method does not suffer from such an ambiguity. It yields results that are consistent with the mechanical drawing definition of thickness. In the sphere-based method, the thickness value at a point **P** is defined as the diameter of the maximum inscribed sphere (MIS) contacting the surface at this point. The locus of the center points of the inscribed spheres constitutes the medial surface (medial axis in 2D) of a CAD model. M. Inui et al. [10] uses distance fields hosted at voxelization to calculate

Algorithm 3 Check edge accessibility

```
procedure CheckEdgeAccessibility($S, U$)                    ▷ $S$ is a CAD model, $U$ is a universum
    $R \leftarrow \emptyset$                                               ▷ Result set
    $G \leftarrow$ initialize AAG from $S$
    for each concave sharp edge $e \in U$ do                   ▷ Iterate over all candidate edges
        if $\alpha(e) < 90^\circ$ then                                 ▷ $\alpha$ is dihedral angle
            $R \leftarrow$ all segments of $e$
        else
            $F = \{f_1, f_2\} \leftarrow$ incident faces of $e$
            for each segment $s \in e$ do                      ▷ Iterate over all polygon links of $e$
                $h_1 \leftarrow$ raycast $f_1$ over $s$
                $h_2 \leftarrow$ raycast $f_2$ over $s$                 ▷ $h_i$ stores the number of hits over $f_i$
                if $h_1 > 0$ and $h_2 > 0$ then               ▷ Segment $s$ is invisible in both directions
                    $R \leftarrow s$
                end if
            end for
        end if
    end for
    return $R$
end procedure
```

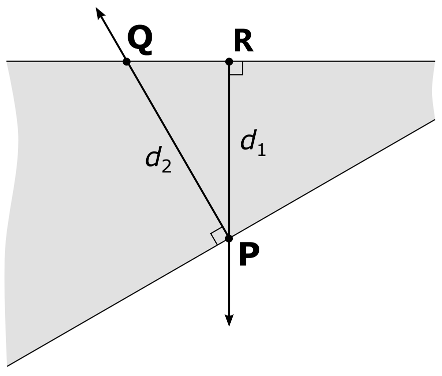

Figure 5. To the ray-based thickness definition at point **P**.

the diameters of the inscribed spheres. The authors propose a method of shrinking spheres, which can be implemented using the same algorithmic core as in the ray-based method. Both approaches start from a triangulation covering the boundary of a CAD model. To obtain accurate results, the initial mesh should be fine enough. The shrinking spheres method attempts to construct a series of MIS starting from the mean point of each triangle. From each mean point, the algorithm casts a ray in the direction opposite to the surface normal. The midpoint between the ray source and its closest intersection is taken as the initial center of a sphere. All triangles intersected by this sphere are collected, and the closest one is chosen to shrink the initial guess. The shrinking procedure stops when the radius of the newly generated sphere is stabilized at some value, which is taken as the result.

For the ray-triangle intersection test, we use the algorithms published in the monograph by C. Ericson [11]. In the broad-phase of the intersection, these algorithms use the bounding volume hierarchy structures (BVH) constructed on the axis-aligned bounding boxes (AABB) to which the initial CAD model is decomposed after the surface meshing.

The output of the thickness analysis algorithm is a scalar map of thickness values distributed over the surface triangulation M.

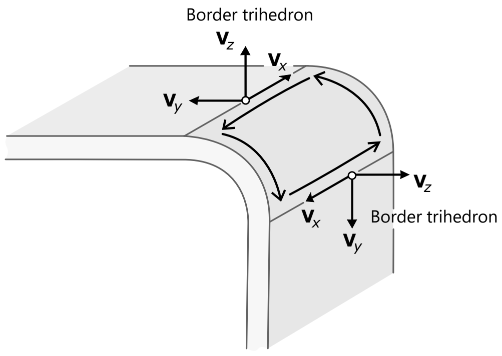

Figure 6. Example of border trihedrons computed on a sheet metal part.

3.5. Check clearance

We define clearance as the minimal distance from a single face f or a feature F to its orthogonally opposite face. The algorithm of clearance measurement is identical to the thickness check (Section 3.4) but conducted for the inverse normal field over the surface mesh M.

3.6. Check milling axes

A set of milling axes A_f associated with the face f defines all milling directions the corresponding face can be approached from. The usage of these axes is two-fold. First, knowing the axes, it is possible to group faces into meaningful features, indicating that the corresponding group can be manufactured as a whole (e.g., slots, pockets). Second, the milling axes can be used to compute local depth d_f from the corresponding face to the stock boundaries. The latter estimate approximates the tool length required to finish the face f. The milling axis candidates for a face f are extracted using the following assumptions:

1. The face f is of an analytic type, i.e., the corresponding surface $\mathbf{s}(u, v)$ is a plane, cylinder, cone or torus. If not, canonical conversion should first be applied [21].
2. Planar faces can have 5 axes: one normal direction (end-milling) and 4 in-plane vectors. In the simplest implementation, the in-plane vectors can be chosen aligned with the U and V parametric directions of $\mathbf{s}(u, v)$, although it is recommended to reorient those axes to follow along the straight edges of a face.
3. If a cylindrical face is not a blend feature, it can have only a pair of axes aligned with the axis of the cylinder (except for the cases of ball-ended mills). If a cylindrical face is a blend feature, a couple of end-milling axes are added.
4. Only one local system of axes is associated with a single face, i.e., we are not checking face accessibility on its whole span here, leaving such a test to a dedicated algorithm (Subsection 3.2). The origin point for the axes is chosen within the face's interior at a random point.

A candidate axis $a \in A_f$ undergoes the following heuristic checks $P_1, P_2, ..., P_5$ for planar faces:

P_1: `CheckAllConcaveParallel`
Side milling is only possible when all concave edges are straight and parallel to each other.

P_2: `CheckTowardConcave`
A side milling axis directed toward a concave edge should be collinear with the local V_y axis at this edge (V_x, V_y, V_z axes are defined according to the border trihedron, see Fig. 6).

P_3: `CheckAlongSharpConcave`
A side milling axis cannot be aligned with any sharp concave edge.

P_4: `CheckAccessible`
All axes should be accessible. This fact is checked by global ray casting.

P_5: `CheckSharpConcaveNeighbors`

An end-milling axis can potentially be cancelled by the presence of a concave corner (concave vertex) in the candidate plane. However, if such a cancellation would lead to no axes available (i.e., all side-milling axes were canceled beforehand), such an axis is still allowed (otherwise, such a face would have to be marked as non-machinable, although this latter fact can be checked separately, see Subsection 3.2).

For the cylindrical and conical surfaces, only `CheckTowardConcave` and `CheckAccessible` tests are performed.

4. Uniform triangulation

For accessibility and thickness/clearance analysis, we seek a simple yet fast mesh generation technique that would allow for uniform coverage of the model boundary ∂S. The simplicity of the algorithm is mainly possible due to the relaxed quality criteria imposed on the generated triangles. The algorithm is not required to control aspect ratio and other conditions typically posed on mesh elements used in FEA-based simulation. Another advantage of a simple meshing technique is its better potential to overcome grid generation problems, such as those mentioned in [17].

In this section, we describe a "hybrid" approach following the classification from [12]. The mesh is constructed in the 2D parametric spaces of curvilinear faces and then mapped to the 3D modeling space. The construction of a 2D mesh is based on the so-called Adapted Mesh Template, according to the classification from [13]. In our case, quadtree is used to overlay the UV domains of the curvilinear faces, as, for example, in [14, 15, 16] and others.

The main parameters for constructing a mesh are listed below (see also Fig. 7):

s_{min}	The minimum size of the mesh element.
s_{max}	The maximum size of the mesh element.
q_{max}	The maximum depth of a quadtree.
d_h	The linear deflection.
d_α	The angular deflection.

Algorithm 4 Uniform triangulation workflow

```
1: Discretize edges.
2: Overlay a quadtree-based grid over the $UV$ space of each face.
3: Classify each quad node as IN-ON-OUT w.r.t. the discretized face boundaries in the $UV$ space.
4: Generate polygons by quad splitting for the inside quads and tessellating the boundary-crossing quads.
```

In what follows, each step of the Algorithm 4 is discussed in more detail.

4.1. Discretize edges

Curve discretization is a typical first step in mesh generation algorithms [15], [18]. To construct a polygon for a curved edge, the following principal checks are used (see Fig. 7):

- Angular deviation check in 3D.
- Linear deviation check in 3D.
- Segment length check in 3D.

The discretization process is done according to the following steps:

1. Split curve by half division.
2. Check for deviations and segment size.
3. The first two steps are repeated until each generated segment satisfies the deviation and size criteria listed above.

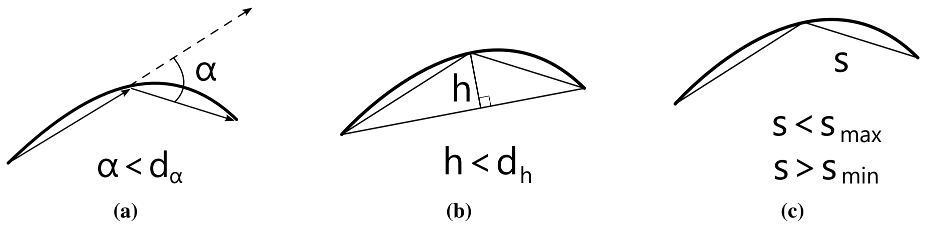

Figure 7. Simple tests for discretization: (a) angular deviation check, (b) linear deviation check, and (c) segment length check

4.2. Construction of a quadtree

The axis-aligned bounding box (AABB) of a face f in its UV space is used as the root of the quadtree. The main parameters used to subdivide a quad q are as follows:

- Maximum tree depth q_{max}.
- Element size: $s^2_{max} > area(q) > s^2_{min}$.

Each quad is subdivided no deeper than q_{max} times provided that the element size keeps exceeding the value of s_{min}. The distance between quad nodes is controlled in 3D space (Fig. 8).

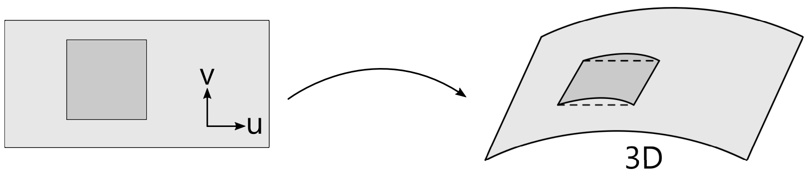

Figure 8. All characteristics for quad are calculated in 3D.

If subdivision is ambiguous in the case of nearly equal side lengths, then the largest deviation angle between the surface and quad normal vectors is taken into account (Fig. 9).

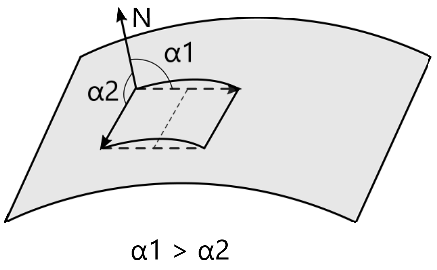

Figure 9. An example of dividing a quad with respect to the angular deviation parameter.

4.3. Classify quads

The corner points of each quad are classified with respect to the discretized face contour in the UV space. We use a ray casting technique capable of returning the ON status for the classified point (unlike some publicly available fast techniques [19]). As a result of classification, the membership status of a quad as a whole is determined:

- If all nodes are IN, then the quad is also IN.
- If all nodes are OUT, then the quad is also OUT.
- In other cases, a quad has the ON status.

4.4. Generate polygons

The segments of the discretized contours are used as the sides of the outcome facets, so that the constructed triangulation remains constrained by the initial CAD boundaries. The triangles are constructed by tessellating the classified quads using the following rules:

- Two connected IN nodes produce a facet link directly.
- For the ON nodes, the nearest point on the contour is searched for. The IN node and the found point are connected by a link.
- For the OUT nodes, we search for the nearest point on the intersected contour segment. A link is built from the adjacent nodes to the found point.

All polygons having more than three nodes are divided using the following rules:

- The division is made according to the larger angle.
- Small angles after division are not allowed.
- The length of the dividing element/edge is greater than s^2_{min}.
- Small cycles (polygons) are removed together with nodes and adjacent edges.

The general scheme of the mesher is shown in Fig. 10.

5. Conclusions and future work

The DFM checks presented above are implemented in the C++ language on top of our Analysis Situs CAD platform [22]. The input data for the DFM analysis system is a file in STEP format (ISO 10303). The output is two-fold. First, a JSON file containing the descriptions of all recognized features and model properties is composed. Second, the visual DFM feedback containing problematic facets, links, and points is dumped in glTF format. The latter facilitates usage of the developed system in contemporary web-based MaaS (Manufacturing as a Service) platforms.

Our system was tested on a broad range of realistic CAD models. In the Fig. 11, two phases of our CAD processing workflow are depicted. The preparation phase (Fig. 11a) includes meshing the input geometry and computing its basic properties, such as mass-inertia characteristics. This stage also performs feature recognition for holes, blends, modeled threads, prismatic features, etc. The second phase (Fig. 11b) corresponds to the elapsed time for the DFM checks alone. As one can see, the preprocessing time generally exceeds the time spent on DFM tests, while both phases are comparable in their contribution to the overall performance. Our observation is that both phases are efficient enough to provide almost instant manufacturability feedback.

Close collaboration with industrial practitioners makes us believe that the DFM toolbox is never complete. We anticipate adding more methods in the future to better handle nested prismatic features and make the recognition of milling axes more accurate. It should be noted that all the methods listed above do not answer completely the main DFM question: if a part is manufacturable with a reasonable amount of effort in a specific machine shop. To make estimates more precise, it would be necessary to accommodate geometric information for tools and probably add some elements of material removal simulation.

References

[1] *Gatenby D.A. and Foo G.* Design for X (DFX): Key to competitive, profitable products, in AT&T Technical Journal. 1990. V. 69. P. 2–13.

[2] *Gupta S.K., Das D., Regli W.C., et al.*, Automated manufacturability analysis: a survey, in Research in Engineering Design. 1997. V. 9. P 168–190.

[3] *McKay A. and Pennington A.* Shape Embedding : A Means of Superimposing Alternative Design Descriptions on Shape Models, in Computer-Aided Design. 2022. V. 152.

[4] *Regli W.C., Gupta S.K. and Nau D.S.* Feature recognition for manufacturability analysis. 1994. Institute for Systems Research Technical Reports.

[5] *Kerbrat O., Mognol P. and Hascoe, J.* A new DFM approach to combine machining and additive manufacturing, in Computers in Industry. 2011. V. 62, P. 684–692.

[6] *Khardekar R., Burton G. and McMains S.* Finding feasible mold parting directions using graphics hardware, in CAD Computer Aided Design. 2006. V. 38(4), P. 327–341.

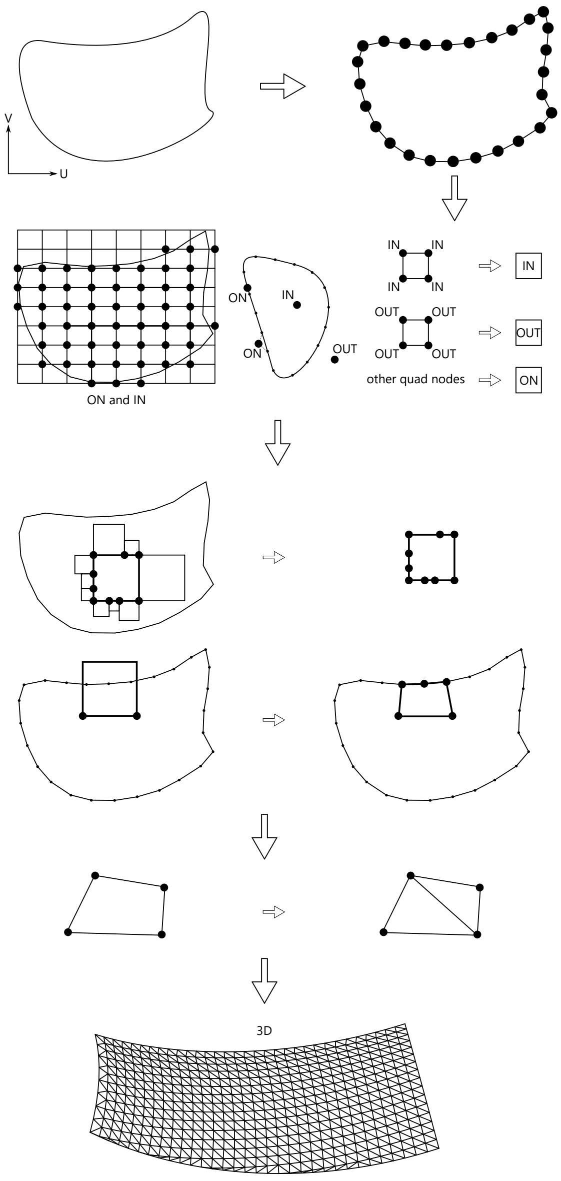

Figure 10. The general scheme of the uniform mesher.

[7] *Surti A. and Reddy N.V.* Non-discretized approach to visibility analysis for automatic mold feature recognition using step part model, in Journal of Advanced Manufacturing Systems. 2012. V. 11(1), P. 1–16.

[8] *Slyadnev S., Malyshev A. and Turlapov V.* On the Role of Graph Theory Apparatus in a CAD Modeling Kernel, in Proceedings of GraphiCon 2020.

[9] *Williams A., Barrus S., Keith R. and Shirley M.P.* An efficient and robust ray-box intersection algorithm, in Journal of Graphics Tools. 2003.

[10] *Inui M., Umezu N., Wakasaki K. and Sato S.* Thickness and clearance visualization based on distance field of 3D objects, in Journal of Computational Design and Engineering. 2015. P. 183–194.

[11] *Ericson C.* Real-time Collision Detection, Morgan Kaufman, 2005.

[12] *Guo J., Ding F., Jia X. and Yan D.-M.* Automatic and High-quality surface mesh generation for CAD Models. Computer-Aided Design, vol. 109, P. 49–59, 2019.

[13] *Ho-Le K.* Finite element mesh generation methods: a review and classification. Computer-Aided Design, vol. 20, P. 27–38, 1988.

[14] *Yerry M. and Shephard M.* A modified quadtree approach to finite element mesh generation. IEEE Comput. Graph. & Applic. vol. 3 (1983) P. 39–46.

[15] *Frey P.J. and Marechal J.* Fast adaptive quadtree mesh generation. In 7th International Meshing

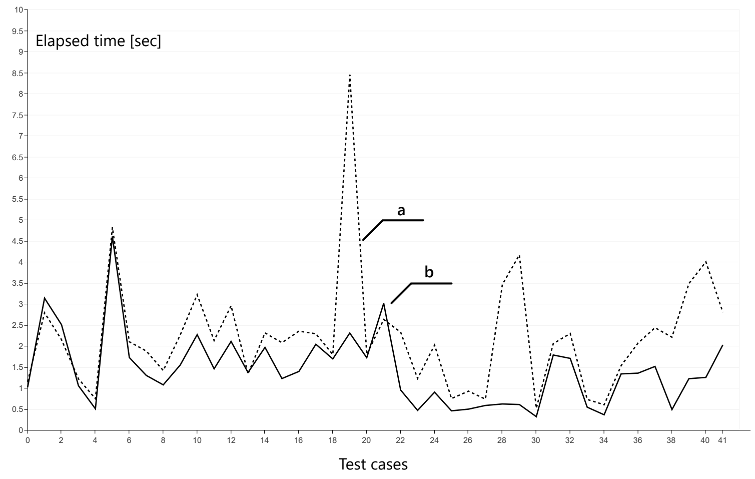

Figure 11. Two phases of the DFM analysis system: (a) preparation and feature recognition, (b) DFM checks.

Roundtable, P. 211–224. Citeseer, 1998.

[16] *Pochet A., Celes W., Lopes H. and Gattass M.* A new quadtree-based approach for automatic quadrilateral mesh generation. Eng. Comput. Ger. 2017,33, P. 275–292.

[17] *Gammon M., Bucklow H. and Fairey R.* A review of common geometry issues affecting mesh generation. AIAA SciTech 2018-1402, Kissimmee FL, Jan. 2018.

[18] *Cuillière J.C.* A direct method for the automatic discretization of 3D parametric curves. Computer-Aided Design, vol. 29, pp. P. 639–647, 1997.

[19] *Haines E.* Point in Polygon Strategies. Graphics Gems IV, ed. Paul Heckbert, Academic Press, P. 24–46, 1994.

[20] *Manifold Geometry.* On sheet metal unfolding (Part 4). URL: http://quaoar.su/blog/page/on-sheet-metal-unfolding-part-4-1 (accessed: 30.07.2022)

[21] *Thepade S.D., Sudhakar D.S.S., Jadhav S.N.* Geometric Forms Recognition: Approaches to Find Shapes from NURBS. 2012.

[22] *Slyadnev S., Malyshev A. and Turlapov V.* CAD model inspection utility and prototyping framework based on OpenCascade, in Proceedings of GraphiCon 2017. P. 323–327. Perm, Russia.