Collision-free tool orientation optimization in five-axis machining of bladed disk

Li Chen, Ke Xu, Kai Tang*  
Department of Mechanical and Aerospace Engineering, Hong Kong University of Science and Technology, Hong Kong  
Received 11 April 2015; received in revised form 30 May 2015; accepted 2 June 2015  
Available online 16 June 2015  

Abstract  
Bladed disk (BLISK) is a vital part in jet engines with a complicated shape which is exclusively machined on a five-axis machine and requires high accuracy of machining. Poor quality of tool orientation (e.g., false tool positioning and unsmooth tool orientation transition) during the five-axis machining may cause collision and machine vibration, which will debase the machining quality and in the worst case sabotage the BLISK. This paper presents a reference plane based algorithm to generate a set of smoothly aligned tool orientations along a tool path. The proposed method guarantees that no collision would occur anywhere along the tool path, and the overall smoothness is globally optimized. A preliminary simulation verification of the proposed algorithm is conducted on a BLISK model and the tool orientation generated is found to be stable, smooth, and well-formed.

© 2015 Society of CAD/CAM Engineers. Production and hosting by Elsevier. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Keywords: Five-axis machining; Tool orientation; Optimization; Bladed disk

1. Introduction

In five-axis machining, tool orientation formed by the two rotary axes plays a crucial role in machining complex parts like aeronautic BLISKs. Poorly defined tool orientations in such applications can cause fatal collisions and damage the in-processing part, while unsmooth patterns of tool orientation may lead to unwanted vibrations due to sudden fluctuations in tool motions, and thus inevitably lowering the machining accuracy.

Much effort has been dedicated to the five-axis tool orientation determination and optimization. Currently, there are three main considerations, namely:

- interference (local gouging and global collisions) – free between the tool and the machine and the workpiece itself [1–5];  
- machine kinematics (feedrate, acceleration and jerk for each axis) [6–9];  
- finished surface quality [10–12].

Most of the current studies are mainly carried out considering only one or two of these aspects. However, to optimize one aspect may compromise the others. For instance, some kinematic-based optimization methods strive to smooth the tool orientation under the constraints of maximum allowable angular acceleration and jerk, while at the same time the hard requirement of collision-free is ignored. On the other hand, due to the fact that the input part geometry pattern can be arbitrary, most collision-free tool orientation determination methods in commercial software are based on trial-and-error and require large amount of human involvement. As the modification of tool orientation is inherently a local operation, unsmooth geometrical patterns of tool orientation tend to appear, thus undermining the kinematics performance of the tool path as well as adversely affecting surface finish quality.

Still, a great deal of effort has been contributed towards better planning and determination of tool orientation in five-axis machining, by settling several aspects as boundary conditions and others as optimization goals. Choi [13] proposed a configuration space method to map the obstacles and machine's limits to a 2-D configuration space to give a feasible tool orientation range. Balasubramaniam [2] proposed
a visibility based method which captures tool accessible ranges in 3-D Cartesian space. Castagnetti et al. proposed a DAO (Domain of Admissible Orientation) concept to optimize a tool path [14]. The general idea of these methods is to geometrically limit the tool orientation selection to a certain area, namely an accessible area for the machine tool. In such an area, a number of aspects are considered. For instance, collisions and local gouging can be strictly forbidden by constraining the tool accessible range. The drawbacks of this kind of schemes are obvious. Firstly, they usually demand a huge amount of computational resource to compute the tool accessible range for every CC point along the tool path. Then, the theoretical tool accessible range often has complex and irregular boundaries, under which it is often very difficult to solve the optimization problem.

In this paper, a plane based accessible region calculation algorithm is proposed. This algorithm simplifies the accessible region by expressing it on a reference 2D plane. For each CC point along the tool path, a reference plane is assigned, on which the tool accessible range will be calculated. And finally, tool orientations are selected and optimized in those ranges. One advantage of our plane based algorithm is that the smoothness of the tool orientation can be pre-determined to some extend in the earlier plane assignment phase. The task of tool orientation smoothing is thus divided into two stages, i.e., the reference plane assignment stage and the tool orientation selection stage. Both stages contribute to the final overall tool orientation patterns. Dividing the task into two phases makes the optimization task more manageable.

The rest of this paper will provide a detailed description of the proposed algorithm. Chapter 2 introduces the algorithm for calculating a planar accessible region. Chapter 3 develops the optimization scheme for the tool orientation selection and smoothing. Chapter 4 presents the simulation results, together with a conclusion.

2. Identification of plane based accessible region

A particular tool orientation (usually represented by a 3D vector in workpiece coordinate system) should be assigned to every cutter location (CL) point after the CL curve has been generated. Due to the complexity of the blade geometry, the tool orientation should be confined inside a certain region to avoid potential collision and local gouging, which is also known as the accessible region of the tool orientation.

Normally, the real accessible region is a closed region on the unit Gaussian sphere centered at the specific CL point. However, to calculate the exact accessible region for every CL point is still a heavy workload for today’s PCs. Besides, as we will pick up a group of smoothed tool orientations inside each corresponding region, the regions are desired to be simplified as regular geometric bodies such as cones on the Gaussian sphere, as illustrated in Fig. 1. As we require that the tool orientation for any CL point be restricted in a plane, the corresponding set of tool orientations for selection now degenerates to a geodesic arc on the Gaussian sphere.

2.1. Selection of the reference plane

The CL curve for machining a blade surface on a blisk, regardless of the root and hub surface, is usually in a spiral pattern, starting from the top and down to the bottom (see Fig. 2). To guarantee that the tool orientation along the CL curve changes smoothly, a set of continuously changing reference planes is the foremost necessity. Moreover, each plane should be uniquely determined for each CL point, thus to certify a fixed accessible region.

To fulfill these requirements, we determine the eligible reference plane \( \mathbf{RP}_i \) : \( \{ \mathbf{p} \in \mathbf{RP}_i | (\mathbf{p} - \mathbf{CLp}_i) \cdot \mathbf{n}_i = 0 \} \), per CL point \( \mathbf{CLp}_i \), through the following two steps:

**Step 1:** calculate the geometric center \( \mathbf{C} \) of the blade to be machined, by connecting the center point \( \mathbf{C}_0 \) of the rotor, a virtual axis of the blade can be obtained as \( \mathbf{C}_0 \mathbf{C} \).

**Step 2:** find out the blade surface normal \( \mathbf{s}_i \) at the CC point corresponding to the specified CL point; the normal vector \( \mathbf{n}_i \) of the reference plane \( \mathbf{RP}_i \) is calculated as the cross product of \( \mathbf{C}_0 \mathbf{C} \) and \( \mathbf{s}_i \):

\[
\mathbf{n}_i = \mathbf{C}_0 \mathbf{C} \times \mathbf{s}_i \quad (2.1)
\]

Therefore, the reference plane is now fully determined by a point \( \mathbf{CLp}_i \) and a normal vector \( \mathbf{n}_i \), as shown in Fig. 2. All the possible tool orientations \( \mathbf{T} \) for this particular CL point are restricted to lie in this plane as spanned by the two vectors...
To find a smoother tool orientation transition, the two coefficients $a_1$ and $a_2$ should be kept as constants or within certain variation, the only thing left that affects the continuity of tool orientation is $dsn_i/ds$. As long as the surface normal along the CC curve changes smoothly, the tool orientation is guaranteed to be smooth.

### 2.2. Calculating the accessible region for a ball-end mill

Once the reference plane has been settled, the tool axis is essentially locked inside this 2D region. Unlike the visible region of a ray emitting from a CL point, the accessible region of the tool is a strictly proper subset of the former, since the ball-end tool has a specified shape of tool radius $R$, length $L$ and holder radius $R_h$ (see Fig. 4).

The first step for calculating the accessible region is to triangulate the blade geometry into a triangular mesh body. Owing to the particular geometry of the blisk, only the two neighboring blades can post as obstacles to the blade in between. Let these three blades be labeled as LB, MB, and RB respectively, as shown in Fig. 5. All the three blades are triangular meshes.

Next, for a given CL point $CLp_i$, we calculate the normal direction $n_i$ of the reference plane $RP_i$ by Eq. (2.1); if $n_i$ is facing backward, then the potential obstacle should be RB, otherwise, LB.

---

**Fig. 3.** Determine the reference plane. 

**Fig. 4.** Difference between the accessible and visible region of a tool. 

**Fig. 5.** Calculate the accessible region: in 3D (left) and in the plane (right).
Intersecting the reference plane with MB and the potential obstacle will result in two loops of points (the intersection with the potential obstacle could be null for some special cases, as to be explained).

Fig. 5 illustrates the accessible region in both the workpiece coordinate system (WCS) and the introduced plane coordinate system (PCS). Initially, the left and right extreme tool position are assigned to be \( \pi \) and 0 in PCS, which makes the accessible region to be 180° in the first place. After that, we search for the critical point \( C_p \) on \( \text{Loop}^0 \) which the tool reaches from the right, and, similarly, the critical point \( C_p \) on \( \text{Loop}^1 \) which the tool reaches from the left. In some special positions where the reference plane has no intersection with the neighboring blade, \( \text{Loop}^1 \) will be null; e.g., referring to Fig. 6, in such a case, the right extreme tool position will keep its initial one, which is 0° in PCS.

So far, we have established the reference plane for every CL point and identified the embedded accessible region. In the next chapter, the optimal tool orientations will be selected among the calculated planar accessible regions, targeting at a smooth tool motion along the CL curve.

### 3. Tool orientation optimization

#### 3.1. Overview

The previous chapter has discussed about the scheme to derive constrains for tool orientation selection. The tool orientations selected under these constrains are guaranteed to be collision-free. Fig. 7(a) gives an illustration of the reference-plane-restricted tool orientations along a smooth CL curve.

For the simplicity of discussion, the tool orientations are defined by the tool axial unit vectors \( TO = [t_{o1}, t_{o2}, t_{o3}, ..., t_{on}] \), where \( n \) is the total number of CL points on the CL curve. A planar tool accessible range is then represented by a fan area in the reference plane's local 2-D system. As can be seen from Fig. 7(b), for each tool accessible range, an angle \( \phi_i \) would uniquely define a tool orientation \( t_{oi} \). So the task here is to find an appropriate \( \phi_i \) for each CL point.

The objective of this tool orientation selection and the ensuing optimization is to obtain a set of gradually and smoothly changing tool orientations along the entire CL curve. In five-axis machining, the tool movement is often represented by a ruled surface whose rulings correspond to the tool axes. So the “smoothness” of the tool orientations can be represented by the geometrical features of the corresponding ruled surface.

#### 3.2. Tool orientation representation

The unknown variables are thus \( TO = [t_{o1}, t_{o2}, t_{o3}, ..., t_{on}] \), which denote the unit vectors of the tool orientation. Originally, a unit vector in 3-D space is defined by two real numbers. In this paper, since the tool orientations are confined in a plane, the tool axial vector's degree of freedom is effectively reduced to one.

Let \( TOC^1 = [toc^1_1, toc^1_2, toc^1_3, ..., toc^1_n] \) and \( TOC^2 = [toc^2_1, toc^2_2, toc^2_3, ..., toc^2_n] \) be the two sets of unit vectors representing the boundary of the \( n \) tool accessible ranges. Any selected \( t_{oi} \) thus can only fall in the planar fan delimited by \( toc^1_i \) and \( toc^2_i \). It is obvious that any vector in this fan falls on the great arc defined by \( toc^1_i \) and \( toc^2_i \) on the unit sphere. We make use of Spherical Linear Quaternion Interpolation (SLERP) [15] which guarantees that all the interpolated quaternions are on the unit sphere. SLERP can be independent of quaternion. The geometric relationship between the two boundary unit vectors \( \overrightarrow{OA}, \overrightarrow{OB} \) and any arbitrary unit vector \( \overrightarrow{OH} \) whose endpoint falls on the great arc \( \sim AB \) of the unit sphere is shown in Fig. 8:

where \( \Phi \) is the angle between \( \overrightarrow{OA} \) and \( \overrightarrow{OB} \), \( \phi \) the angle between \( \overrightarrow{OA} \) and \( \overrightarrow{OH} \), and \( \overrightarrow{OE} \) is perpendicular to \( \overrightarrow{OA} \).

Obviously, we have:

\[
\| \overrightarrow{OE} \| = \sin \Phi \tag{3.1}
\]

\[
\| \overrightarrow{OF} \| = \sin \phi \tag{3.2}
\]

\[
\| \overrightarrow{OC} \| = \| \overrightarrow{OF} \| \| \overrightarrow{OE} \| \tag{3.3}
\]
Submit Eqs. (3.1) and (3.2) into Eq. (3.3):

\[
\|\mathbf{OC}\| = \frac{\sin \Phi}{\sin \phi} \quad \mathbf{OC} = \frac{\sin \Phi}{\sin \phi} \mathbf{OB}
\]

Similarly,

\[
\mathbf{OD} = \frac{\sin (\Phi - \phi)}{\sin \phi} \mathbf{OA}
\]

So finally, we have:

\[
\mathbf{OH} = \frac{\sin (\Phi - \phi)}{\sin \phi} \mathbf{OA} + \frac{\sin \phi}{\sin \phi} \mathbf{OB}
\]

Normalizing the angular interval \(\phi \in (0, \Phi)\) to \(x \in (0, 1)\), and denoting \(\mathbf{OA}, \mathbf{OB}\) and \(\mathbf{OH}\) by \(\text{toc}^1\), \(\text{toc}^2\) and \(\text{tore}\) respectively, Eq. (3.6) can then be written as:

\[
\mathbf{to} = \text{Slerp}(\text{toc}^1, \text{toc}^2, x) = \frac{\sin (\phi - \Phi)}{\sin \phi} \text{toc}^1 + \frac{\sin (\phi)}{\sin \phi} \text{toc}^2, x \in (0, 1)
\]

Now that the tool orientations are properly represented in a normalized space, the tool orientation selection and optimization problem can then be expressed as follows:

Selecting/optimizing \(X = \{x_1, x_2, x_3, ... x_n\}, x_i \in (0, 1)\), so that the overall smoothness of the tool trajectory rule surface is the best.

### 3.3. Optimization problem formulation

In five-axis machining, the concept of “smoothness” is inherently complicated and often inconsistent. Intuitively, “smoothness” is more or less a local property. In this paper, the tool axial trajectory ruled surface is represented by a series of discrete tool axes. So, naturally, the “local smoothness” should be calculated by two or more adjacent tool axes.

The most obvious intuition is that smoothness means that the tool axis sweeps as little area as possible along the tool path. This definition of smoothness is illustrated in Fig. 9.

Since the tool orientations are represented by a set of discrete vectors, the area of the tool trajectory surface can be approximated by the area of the sum of area of the bilinear patches formed by every two consecutive tool axial vectors on the tool path. A useful measure for the bilinear patch is given by Chen and Pottmann in [16] as:

Let \(l_1\) and \(l_2\) be two line segments in \(R^3\) space, with \(p_1, p_2\) and \(q_1, q_2\) being their end points respectively. Establish a mapping between the two line segments:

\[
(1 - \lambda)p_1 + \lambda q_1 \rightarrow (1 - \lambda)p_2 + \lambda q_2, \lambda \in [0, 1]
\]

Connecting the associated points gives the bilinear patch formed by \(l_1\) and \(l_2\). The area measure of this patch is given as:

\[
A(l_1, l_2) = 3 \int_{0}^{1} [(1 - \lambda)(p_1 - p_2) + \lambda(q_1 - q_2)]^2 d\lambda
\]

For our application, to define a tool axial vector \(\mathbf{to}_i\), we need two end points of which one is known and fixed, i.e., the CL point \(c_i\), while the other, to be denoted by \(t_i\), is subject to modification for optimization (note that \(t_i = c_i + t_{oi}\)). The \(i\)th bilinear patch area in Eq. (3.8) is then given by:

\[
A_i = [(t_i - t_{i+1})^2 + (c_i - t_{i+1})^2 + (t_i - t_{i+1})(c_i - c_{i+1})]
\]

The optimal smoothness of the tool trajectory surface can then be achieved by minimizing the sum of all the \(n-1\) bilinear patches’ areas.

On the other hand, however, minimizing the tool axis’s swept area alone does not warrant the most desirable tool orientation changing pattern. As illustrated in Fig. 10, the two sets of tool orientations (for a same CL curve) have approximately the same tool axial trajectory surface areas. The former accelerates and also decelerates its change substantially along the CL curve, whereas the latter changes much more mildly and exhibits a more rotational (desirable) pattern.
As we are dealing with discrete data, the local angular velocity can be approximated by mean angular velocity in each interval between two neighboring tool orientations. Assuming that the tool center moves in a steady feed rate, the mean angular speed of the tool orientation change in the $i$th interval can be expressed as:

$$
\omega_i = \frac{\theta_i}{l(c_i,c_{i+1})} \quad (3.10)
$$

where $\theta_i$ is the angle between the $i$th and $(i+1)$th tool axial vectors $t_{oi}$ and $t_{oi+1}$, and $l(c_i,c_{i+1})$ stands for the arc length between the two CL points. In real situation, usually two consecutive CL points are close, and $\theta_i$ is small, so we have: $l(c_i,c_{i+1}) \approx ||c_i-c_{i+1}||$ and $\theta_i \approx \sin \theta_i \approx ||t_{oi+1}-t_{oi}||$; therefore, we have:

$$
\omega_i = \frac{||t_{oi+1}-t_{oi}||}{||c_i-c_{i+1}||} \quad (3.11)
$$

The smoothness of angular velocity can be achieved by minimizing the harmonic mean of $\omega_i$ or the largest $\omega_i$ along the tool path.

The final tool orientation smoothing problem can then be formulated as the following constrained optimization problem (refer to Eq. (3.7)):

$$
\min_{X = \{x_1,x_2,\ldots,x_n\}} \sum_{i=1}^{n-1} A_i
$$

subject to:

$$
0 \leq x_i \leq 1 \quad (3.12)
$$

Or:

$$
\min_{X = \{x_1,x_2,\ldots,x_n\}} \sum_{i=1}^{n-1} \omega_i
$$

subject to:

$$
0 \leq x_i \leq 1 \quad (3.13)
$$

### 3.4. Tool orientation selection and optimization schemes

Local process determines one tool position at a time, and once a tool position is determined, it will not be modified in the future process. Greedy based algorithms are suitable for this task owing to the fact that they focus only on finding the optimal choice at hand.

Greedy based algorithms have clear advantages of easy implementation and relatively good computational efficiency. Their biggest deficiency is the “short sightedness”. Under most circumstances, a local optimal solution does not ensure a global optimal one, and may even lead to a worst one. This predicament gets particularly worse when the decision chain grows longer.

In general, whether a local optimum is near the global optimal solution depends largely on how the “greediness” is defined, i.e., it depends on the local optimal sub-problem’s formulation, and the lay-out of tool accessible range. In our particular case, it is very likely that local optimal solutions may tend to swarm to one side of the (planar) tool accessible ranges. To avoid that, a penalty factor is introduced:

$$
\min F_i + \mu \frac{1}{s(1-s)}
$$

subject to:

$$
0 \leq x_i \leq 1
$$

where $\mu$ is a weight to control the influence of the penalty factor, $F_i$ is one of the optimization goals as described in Eqs. (3.12) and (3.13), or the combination of the two.

As mentioned earlier, local optimal solutions do not ensure a global optimal one. In our case, smoothing the tool orientation at one CL point may lead to unsmooth tool orientations at other CL points. In addition, the introduced penalty factor in Eq. (3.14) has the tendency to “drag” the tool orientation to the center of the tool accessible range; the unsmooth alignment of tool accessible range boundary vectors would then have negative impact on the smoothness of the final result.

Therefore, it is plausible to apply global optimization to achieve overall smoothness of the tool trajectory. As the optimization functions and constrains described in Eqs. (3.12) and (3.13) are three times continuously differentiable, the popular SQP (sequential quadratic programming) method [17], which is more efficient in dealing with large scale problems, can be used for solving this global optimization problem.

The basic SQP algorithm for solving our problem is described below.

**SQP algorithm:**

**Step 1:** choose an initial solution $X_0$; it can be obtained by the greedy based algorithm mentioned above, or simply take the middle points of the tool accessible ranges; let $t = 0$.

**Step 2:** approximate the optimization problem described in Eq. (3.14) with a quadratic programming problem at $X'$.:

$$
\min_{X = \{x_1,x_2,\ldots,x_n\}} \left[ \mathbf{V} \sum_{i=1}^{n} F_i(X') \right] d + \frac{1}{2} d^T H d
$$

subject to:

$$
\mathbf{V} g(X') d \leq -g(X')
$$

(3.15)
In which, $H'$ is a symmetric matrix whose $ij$th element is

$$H'_{ij} = \frac{\partial^2 F}{\partial x_i \partial x_j} \quad (3.16)$$

and $g(X) \leq 0$ denotes the constrains $0 \leq x_i \leq 1$.

**Step 3:** solve the quadratic programming sub-problem for the optimal $d^t$.

**Step 4:** if $\|d^t\| \leq \varepsilon$, with $\varepsilon$ being the threshold error given by the user, stop; otherwise, let $X_t+1 = X_t + d^t$, $t = t + 1$, and return to Step 2.

The above algorithm is guaranteed to converge (the proof can be found in [17]).

There is one remaining issue for this SQP scheme: the algorithm often converges at a local minimum solution rather than the global one. This feature of the SQR algorithm makes the choice of initial solution especially important. The algorithm would likely fail to provide a satisfying result if the initial tool orientations are scrambled. This problem gets more serious when the size of the problem, i.e., the number of tool orientations, gets larger. There are a few possible solutions to this problem:

1. provide an effective and universal way to select a relatively high quality initial solution;
2. provide a heuristic mechanism so that the algorithm would escape from a local optimum;
3. reform the optimization problem to convex problem so that it guarantees that the algorithm would converge to the global optimum.

These three solutions, however, have their own drawbacks. For solution 1, the machine components and obstacles may come in all kinds of geometrical patterns. It is a hard task to design a universal scheme that always produces a satisfying initial solution. For the second solution, it would make the algorithm computationally more expensive and at the same time less robust. For the last one, new constrains may be introduced and some of the “smoothness” intuitions may be lost during the reformation of the optimization problem.
Fig. 13. (a) Tool orientation selected by the greedy based algorithm; (b) Tool orientation selected by the greedy based algorithm shown with the neighboring blades; (c) Tool orientation optimized by the SQP algorithm; (d) tool orientation optimized by the SQP algorithm shown with the neighboring blades.

Fig. 14. Comparison of the solutions obtained by the greedy based algorithm and the SQR algorithm.
4. Evaluation and conclusion

The presented tool orientation optimization algorithm has been implemented using C++ and tested on a PC with an average configuration. A model blisk is used to test the algorithm, as shown in Fig. 11. It can be seen from this CAD model that the cavities between the blades are narrow with a complicated shape and hence a good test example for proposed algorithm. The tool path (CL point data) is assumed to be generated from the upper stream procedures.

First, a set of reference planes are generated using the method as described in Chapter 2. The result is illustrated in Fig. 12(a). The generated planes have visually satisfying smoothness, which ensures the overall smoothness of the tool orientations at an early stage. Planar tool accessible ranges are then calculated accordingly. As shown in Fig. 12(b), every planar tool accessible range is represented by its two boundary lines.

Next, we apply our tool orientation selection and optimization operation as introduced in Chapter 3. First, a greedy search scheme is performed. The result is shown in Fig. 13(a) and (b). Although there is no obstacle interference, abrupt changes in tool orientation can be easily seen, due to the drawback mentioned in Chapter 3. We then apply the SQP algorithm to solve the global optimization problem of Eq. (3.13) with result obtained by a greedy based algorithm serve as the initial solution, obtaining a much smoother result, as shown in Fig. 13(c) and (d). The figure also clearly confirms that no interference occurs anywhere along the tool path. The comparison of angular speeds (assuming the tool has a constant feed rate) obtained by the greedy based algorithm and the SQR algorithm is shown in Fig. 14. The mean angular speeds obtained by the greedy based algorithm and the SQR algorithm are 0.6706 and 0.6300 respectively; while the maximum angular speeds are 1.7149 and 0.9267, respectively. It can be seen that the solution given by the SQR method ensures a much lower angular speed. But it is also seen that the overall distribution of SQR's solution is highly correlative with that of the greedy based algorithm. This confirms that the SQR algorithm is sensitive to the initial solution.

The method proposed in this paper formulates the problem in a way such that the collision-free requirement is guaranteed while at the same time plenty of room is left for tool orientation optimization. Although a lot of information might be lost by dividing the tool orientation optimization into two independent phases (i.e., determining the reference planes and then selecting from the planar accessible ranges), the proposed tool orientation determination/optimization method is believed to be useful in applications where good or preferred reference planes for the tool orientation are already known beforehand, such as in our case of blisk machining. In addition, even if such luxury is not there, by iteratively trying different reference planes and then applying the proposed optimization procedure, the user is provided a useful computational tool for finding a globally near-optimal solution.

References