Triangular Prism Element Optimization for Mesh Visualization of Printed Circuit Boards

A. Karen Daniels[1] **and B. Shu Ye**[2]
[1,2]Computer Science Department, University of Massachusetts Lowell, Lowell, MA, USA

Abstract— *Prism elements arise in some printed circuit board modeling contexts, such as visualization and electromagnetic field modeling. Here we consider prisms built by extruding from triangular bases which result from constrained $2d$ Delaunay triangulation. The goal is to partition each extruded prism into sub-prisms of high quality that fit within the given printed circuit board layers. A prism quality measure is introduced and, from it, optimal prism height is derived given a triangular base. Given a printed circuit board's layer heights and optimal prism heights, we provide a method for determining the height of each prism element. The overall prism mesh quality is evaluated, which examines the tradeoff of prism element quality versus the number of elements. The new method also compares favorably with respect to a prior prism mesh generation method that does not involve optimizing prism heights.*

Keywords: Meshing, Visualization

1. Introduction

In recent years, the interconnect modeling on Printed Circuit Board (PCB) and in packaging has become a bottleneck for successful high-speed circuit design [1] and visualization. The signal integrity issues, such as the signal propagation time, the digital pulse distortion, and the cross-talk, all effect the quality of the digital signal and can cause integrated circuit gate misswitching and introduce large bit rate error [2]. Therefore, simple physical constraints on the routing rules are no longer sufficient. For critical nets, accurate circuit simulation is needed, which requires accurate electromagnetic (EM) characterization on interconnects. The finite element based full-wave EM field solver can be applied to perform such tasks which, in turn, rely heavily on the quality of the finite element mesh generation [3]. Figures 1 and 2 (both images courtesy of Cadence Design Systems) provide meshing examples for two common PCB structures: coupled serpentine lines and coupled vias. The PCB has a layered structure. A serpentine line is a transmission line, embedded in a single layer, containing turns to control signal propagation time over a line segment. A via is a vertical signal connection between layers. In this paper we focus on prism mesh generation, as discussed in Section 1.1 and illustrated in Figures 1 and 2. The goal is to provide a high-quality prism mesh which can be used for mesh visualization as well as techniques such as finite element modeling. Part of this process involves creating sublayers of layers, where appropriate. One important goal of mesh visualization is verification of the model structure.

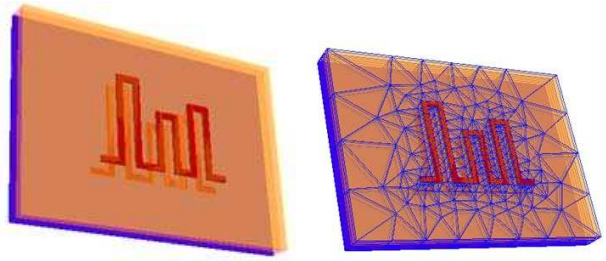

Fig. 1: PCB coupled serpentine line feature (left) with mesh (right).

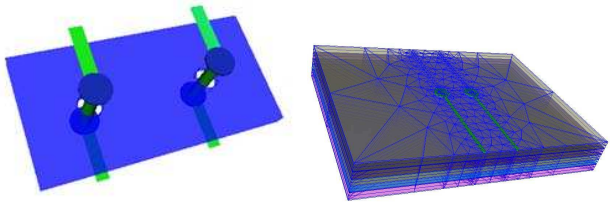

Fig. 2: PCB coupled via feature (left) with mesh (right).

1.1 Prism Mesh Generation

Mesh generation for finite elements has been widely studied (see [4] for a survey). The following mesh definition is given in [4]. T_h is a mesh of Ω if:

- $\Omega = \cup_{K \in T_h} K$.
- The interior of every element K in T_h is non-empty.
- The intersection of the interior of two elements is empty.

Techniques for mesh generation in general have been studied extensively in the geometric modeling and computational geometry communities [4], [5], [6], [7]. Geometric and topological underpinnings of mesh generation are explored in [8]. In some cases, mesh generation is tightly coupled with the EM simulation method (e.g. in [9] there is tight coupling of mesh generation with Finite Element Method (FEM) simulation).

Here we perform mesh generation separately from the EM simulation in order to allow our mesh results to be used as a starting point for mesh visualization or other techniques

such as FEM. We primarily focus on mesh visualization, but we use quality measures that should be valuable across multiple contexts.

Due to the layered specialty of those interconnects on PCB and in packaging, a mesh consisting of triangular prisms is very efficient and sufficient to meet our meshing needs (each prism has triangular top and bottom and three rectangular sides.) On each layer, the vertical height of the interconnect is thin and the shape is irregular, which is very suitable for triangular prism meshing. Along the vertical direction in the layer stack, triangles can be extruded up or down to build prism volume elements which satisfy FEM needs. Due to this special property, a triangular prism is the basic element for our mesh generation [10]. As aforementioned, the quality of triangulation directly effects the accuracy of EM computation for FEM. This is true especially for full-wave EM modeling [11]. The best triangles are the ones that have equal angles, so a Delaunay triangulation algorithm is best utilized (see Section 2.1). Furthermore, PCB feature boundaries must be included in the triangulation, so a *constrained* Delaunay triangulation is used (see also Section 2.1).

The fundamental prism generation strategy that we build upon here is commonly used, as noted in [12]. It is employed in [9], and is detailed in our prior work [13] and summarized in Section 3. The first step is to project all PCB features' line segments from $3d$ orthogonally down onto the $2d$ x-y plane. There a $2d$ constrained Delaunay triangulation is produced using the Computational Geometry Algorithms Library (CGAL) [14] (see Section 2.2). In [9] the Triangle software by Shewchuck [15], [16] is used to generate a constrained Delaunay triangulation. While this is a strong approach, we find that CGAL is adept at handling segment intersection other than at their endpoints. In both our work and [9] edges of the triangulation are extruded up through the layers to construct prisms. In [9] each prism is subdivided into tetrahedra. In contrast, we subdivide the prisms horizontally to preserve the prism mesh structure. Within a given sublayer all of the prisms must have the same height. This is because if prisms have different heights within a sublayer, the generated prisms may not produce a conformal prism mesh. For us, the height of the prisms is a key decision and a crucial contributor to the quality of the overall mesh. The focus of this paper is producing a high-quality mesh by optimizing prism height and relating it to the selection of quality criteria fed as inputs to CGAL.

Other prism meshing research includes [12], [17], [18]. Motivated by problems in biology and medicine, Whitaker *et al.* [12] use iterative relaxation of point samples to create thin layers of triangular prisms. Their triangular quality measure is discussed in Section 4.1. In [17] Yamakawa and Shimada transform a tetrahedral mesh into a hybrid prism-tetrahedral mesh, motivated by FEM applications. They use prisms to reduce the number of elements and provide more accurate FEM analysis. Layers of prisms are created. Prism height can be governed by user input or can be derived "from the average edge length of the triangular faces." They provide a prism quality measure that we discuss in Section 4.1. For thin-walled solids, Yamakawa and Shimada [18] create prisms as an intermediate step in a process that begins with a tetrahedral mesh and ends with a hexahedral mesh. A layer of prisms is added to the boundary of the tetrahedral mesh. Some prisms are converted to hexahedral elements to form a mixed mesh. Finally, midpoint subdivision of the mixed mesh generates a hexahedral mesh. No pyramid elements are generated by this process; they can be a FEM concern.

1.2 Contribution and Overview

In [13] we based prism height inside each PCB layer solely on the thinnest height among the layers. Our triangle quality criteria for input to CGAL were related to the length of a triangle's longest edge and its aspect ratio. We used a triangle quality measure from [19] to evaluate the results. In [13] the focus was on triangle quality for successive refinements of an elliptical pad with a circle as a special case. Prism mesh quality was not evaluated. In this paper we present several contributions beyond the basic prism meshing algorithm. The first idea maximizes prism quality by formulating a prism quality measure based on the regular prism; this is based on the triangle quality measure from [19]. Next CGAL provides a $2d$ constrained Delaunay triangulation. Then we show how to, given a triangle as the prism's base, find the prism height that maximizes prism quality. The optimal height of the individual prism elements can then be used to determine a common prism height. A variety of strategies can be applied to derive common prism height. We give 5 choices and provide guidance on how to select a strategy. The next step produces sub-prisms within each PCB layer guided by the common prism height. Note that this approach is semi-automatic and involves optimizing prism quality. In contrast, the prism height selection method of Yamakawa and Shimada [17], while also semi-automatic, does not appear to select prism height to optimize prism quality. We compare their approach of the average triangle edge length measure with our strategy.

Finally, we perform the following post-processing step. We supply the common prism height to CGAL as a maximum edge length constraint. Thus, CGAL again creates a constrained Delaunay triangulation. We show that our approach compares favorably with respect to the results in [13]. We also examine the trade space of prism element quality versus number of prism elements. Some applications may need to apply further post-processing of the prism elements that we construct using quality criteria. For example, in FEM for PCB structures, the shape functions describing the field (e.g. EM) can further influence the number of prism sublayers required.

The remainder of the paper is structured as follows. First, Section 2 gives background on constrained triangle

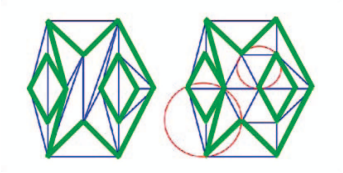

Fig. 3: A constrained triangulation (left) with a constrained Delaunay triangulation (right) (modified from [14]).

meshing, including the definition of a constrained Delaunay triangulation, and brief introduction to CGAL. Section 3 gives the original $3d$ triangular prism meshing algorithm, which uses constrained Delaunay triangulation, as in [13], and its choice of CGAL triangle quality criteria, as well as the triangle quality measure from [19].

Our new algorithm's primary goal is to provide good prism mesh quality. Section 4 presents our prism quality measure and shows how to maximize this measure by deriving optimal prism height given the base triangle. We compare this with the strategy used in [17]. Section 5 describes our revised prism meshing algorithm, including the post-processing step of using prism height to supply CGAL with revised quality criteria. Section 6 presents some results of our revised prism meshing algorithm on PCB examples and shows that it compares favorably with the standard approach. Section 7 concludes the paper and outlines future work.

2. Constrained Triangle Meshing

Here we give background for the prism meshing algorithms in this paper. We define Delaunay and constrained Delaunay triangulation in Section 2.1. Then we discuss CGAL's support for this functionality in Section 2.2.

2.1 Constrained Delaunay Triangulation

Delaunay triangulation ([3], [6], [8], [10]) has the empty circle property: each triangle's circumcircle's interior contains no vertices. The Delaunay triangulation also maximizes the minimum triangle angle size, which supports our $2d$ quality criteria.

Because we must include edges of structural features in the triangulation, we use a constrained Delaunay triangulation [8]. "It is convenient to think of constrained edges as blocking the view. Then, a triangulation is constrained Delaunay if and only if the circumscribing circle of any facet encloses no vertex visible from the interior of the facet" [14]. Among all constrained triangulations of a given set of vertices, the constrained Delaunay triangulation maximizes the minimum angle [8]. Figure 3 illustrates the constrained empty circle property of a constrained Delaunay triangulation, where thick segments are the constrained edges.

2.2 CGAL

CGAL's $2d$ constrained refined meshing [14] is utilized to respect input line segments and control the size of the triangles [15]. A CGAL constrained Delaunay triangulation satisfies the constrained empty circle property stated above in Section 2.1. The CGAL provides easy access to efficient and reliable geometric algorithms in a C++ library. For Delaunay triangulation and mesh generation, we find that CGAL can handle segment intersection other than at their endpoints better than other similar software, such as [16], which is used in [9]. CGAL does not yet support $3d$ constrained Delaunay triangulations (although it does support basic $3d$ triangulation), so we use their $2d$ constrained Delaunay functionality.

CGAL uses shape criterion lower bound B and size criterion of longest edge length to control triangle elements. The lower bound B is the ratio between the circumradius and the shortest edge length. The size criterion is an upper bound on the longest edge length of a triangle element. This criterion can allow users to define small triangles. In Section 3.2 we describe the choices available in the basic prism meshing algorithm.

3. Basic Prism Meshing Algorithm

Section 3.1 summarizes the algorithmic starting point for this paper. Section 3.2 discusses the CGAL triangle quality criteria used in this algorithm. Section 3.3 introduces the chosen triangle quality measure.

3.1 Algorithm

BASIC_PRISM_MESHING_ALGORITHM

1: $E_{xy} \leftarrow$ Initial set of structural feature edges, projected orthogonally onto the x-y plane
2: $l \leftarrow$ number of layers
3: $quality\_criteria \leftarrow 2d$ triangle quality criteria
4: $T_{xy} \leftarrow$ 2D_CONSTRAINED_TRIANGULATION $(E_{xy}, quality\_criteria)$
5: **for** $i = 1$ to l **do**
6: Extrude and create prisms for layer i using $2d$ triangles in T_{xy}. Thinnest height among the layers is found and used to divide each existing layer into sub-layers.
7: **end for**

The mesh examples of Figures 1 and 2 use this approach. The idea, which we employed in [17], comes from Lee's thesis [9] for FEM analysis. Even prior to that, the idea of extruding triangles to form prisms has appeared in the literature as a common approach to prism meshing [12]; in some cases the offset direction comes from surface normals. The components and layers are projected to a $2d$ surface, on which a triangle mesh is generated based on the certain mesh control criteria such as edge length and angle of the mesh triangle elements. Then, the triangle mesh is extruded vertically back to the original layers of $3d$ structure to form prism elements. Prisms are built by connecting vertically adjacent triangles vertically. The results of this strategy are

examined in Section 6 as a baseline for comparison with our new approach. Examples are introduced there for those experiments.

3.2 CGAL Triangle Criteria

In the basic algorithm we used the CGAL default angle bound of 20.7 degrees, which guarantees termination of the constrained Delaunay triangulation algorithm [14]. Good mesh quality in FEM simulation for a PCB context not only relies on the shape of mesh elements, but also closely depends on the wavelength λ. Wavelength has the relationship $\lambda = \delta/f$, where δ is the speed of light and f is the frequency. Consequently, we model the longest edge of a mesh element to be close to or less than one third of the wavelength, (1/3)λ. In today's high-speed design, if we take 50GHz as an example, the longest edge of mesh elements should be at most 2mm. We initially use this criterion, motivated by FEM considerations, as the longest CGAL edge length.

3.3 Triangle Quality

There are many possible ways to measure quality for triangular elements to assess the success of the above algorithm and the choice of CGAL criteria. For example, Whitaker *et al.* [12] use a radius ratio $Q = 3r/R$. Here "r and R are the radii of inscribing and circumscribing circles, respectively." From [4], in an optimal mesh of triangles the triangles are equilateral and "the elements in the mesh have a quality close to 1." One triangle quality formula offered there involves a ratio of longest edge length to inradius.

Here we present the method from [19] that we used in [13]. It forms a solid foundation for the $3d$ extension to the prism case in Section 4.1 and facilitates optimization. For the triangle case [19] the element quality q_t is:

$$q_t = \frac{4\sqrt{3}A}{h_1^2 + h_2^2 + h_3^2} \tag{1}$$

where A denotes the area, and h_1, h_2 and h_3 are the edge lengths. When it is an equilateral triangle $q_t = 1$. This agrees with the view expressed in [4].

4. Prism Mesh Quality

We begin our prism quality discussion by first generalizing Eq. 1 from Section 3.3 to the prism case in Section 4.1. Section 4.2 shows how to maximize prism height given a base triangle, and Section 4.3 examines sensitivity of prism quality to changes in prism height. Section 4.4 discusses overall prism mesh quality.

4.1 Prism Quality

Similar to [12] we start with a triangle quality measure that encourages equilateral triangles and the side faces of our prisms will be perpendicular to the triangular faces. We generalize generalize Eq. 1 to the prism case to accomplish this, and the result lends itself to optimization, which is useful in our context. To generalize Eq. 1 to the prism case the element quality q_p is:

$$q_p = \frac{\frac{32\sqrt{3}}{3}V}{(h_1^2 + h_2^2 + h_3^2 + h_4^2)^{\frac{3}{2}}}. \tag{2}$$

where V denotes the volume, the h's are the edge lengths, and the coefficient of V follows from the constraint that $q_p = 1$ for a regular prism. We assume that h_1, h_2, h_3 are the known edge lengths for the base triangle of the prism, and h_4 is the unknown height which we would like to optimize.

4.2 Prism Quality Maximization

Starting with Eq. 2, for notational convenience let $\beta = h_1^2 + h_2^2 + h_3^2 \geq 0$. This can be calculated for a given base triangle. Our goal is to solve for positive h_4 which maximizes quality q_p. The result using calculus is:

$$h_4 = \sqrt{\beta/2} \geq 0. \tag{3}$$

Although we designed q_p to equal 1 for a regular prism, 1 is not the maximum value of q_p. For an equilateral base triangle with sides all equal to 1, we obtain $q_p = 1.0264$ from Eq. 2 when using Eq. 3 for the height. Using Eq. 2 for q_p, we randomly generated 20,000 triangles to select h_1, h_2, h_3 and with h_4 from Eq. 3. Results appear in the top line of Table 1. Our experiments support the conjecture that $q_p = 1.0264$ is the maximum value of q_p.

Using the fact that h_4 maximizes q_p, one can use calculus to show that the equilateral case when $h_1 = h_2 = h_3 = 1$ does indeed optimize q_p. Since the roles of h_1, h_2, and h_3 in Eq. 2 are symmetric, one can show that the partial derivative of q_p with respect to h_1 equals 0 when $h_1 = h_2 = h_3 = 1$, so that this yields a critical point. These results suggest that the behavior of Eq. 2 makes it a useful prism quality measure.

Table 1: q_p for 20,000 randomly generated triangles using Eq. 3 for h_4 and using average triangle edge length.

method	$max\ q_p$	$min\ q_p$	$avg\ q_p$
Eq. 3	1.02638	$1.91483\ e-4$	.528858
Avg. edge length	.999977	$2.92361\ e-5$	.506697

Table 1 (bottom line) also shows 20,000 random trials for calculating prism height by averaging triangle edge lengths (an approach from [17]). Note that the average quality is smaller here than when using Eq. 3 for h_4. The maximum quality is also smaller and appears to have an upper limit of 1 rather than the upper bound of 1.0264 when using Eq. 3 for h_4. These differences can be justified algebraically.

4.3 Prism Quality Sensitivity

For the purpose of the revised prism meshing algorithm in Section 5, it is useful to examine the sensitivity of Eq. 2 to the value of h_4. A small random example is presented in Figure 4. In that example a triangle's edge lengths are

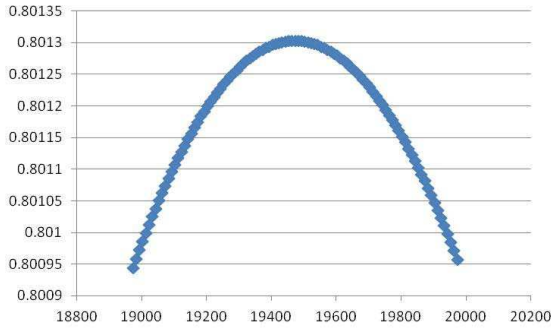

Fig. 4: Prism quality vs. h_4 for a randomly generated triangle.

randomly generated, h_4 is calculated using Eq. 3, and then prism quality is plotted using Eq. 2 for a range of h_4 values near the optimal value. Note the difference in scale for the two axes in the figure. This suggests that, for this sample triangle, prism quality is not very sensitive to changes in h_4. For a collection of triangles, it is possible to compare sensitivity by calculating the second derivative of q_p and then comparing the results across the triangles. This can be used to guide the overall prism quality policy choice in Section 5.

4.4 Overall Prism Mesh Quality

In assessing the results of Section 5's new algorithm in Section 6, we will evaluate the average prism quality across the mesh using Eq. 2. However, this is not the only criterion. From [4]: "An optimal mesh is that for which the chosen quality function is optimal while, at the same time, its number of vertices (elements) is minimal." Thus, we must take the number of prism elements into account in addition to the quality of the prism elements. There is a tradeoff between number of prism elements and prism element quality, which we explore in Section 6. The number of prism elements is addressed in some literature. Yamakawa and Shimada [17] give an example from computational fluid dynamics in which 42,304 elements provided good analysis results.

As noted in Section 1.2, some applications may need further post-processing of the prism elements that we construct. For example, in FEM for PCB structures, wavelength or choice of FEM shape functions may impose a lower bound on the number of prisms within a layer. This lower bound is often small and is satisfied in our work.

5. Revised Prism Meshing Algorithm

Here we introduce our new approach in REVISED_PRISM_MESHING_ALGORITHM. An important challenge is how to select sub-layer heights during the extrusion process so that overall high prism quality is achieved. Step 6 of the algorithm presented in Section 3 used the thinnest height among the layers to produce conformal prisms with a common height inside each sub-layer. Other choices of sub-layer height are possible, and here we use our prism quality maximization from Section 4.2 as the basis for 5 policy choices in CALCULATE_PRISM_HEIGHT. While this choice is user-defined, it can be guided by the sensitivity analysis suggested above in Section 4.3.

REVISED_PRISM_MESHING_ALGORITHM

1: $E_{xy} \leftarrow$ Initial set of structural feature edges, projected orthogonally onto the x-y plane
2: $l \leftarrow$ number of layers
3: $l_h \leftarrow$ layer heights
4: $c \leftarrow$ user's choice of height policy
5: $quality\_criteria \leftarrow 2d$ triangle quality criteria: $longest\_edge\_length$ and $angle\_bound$
6: $T_{xy} \leftarrow$ 2D_CONSTRAINED_TRIANGULATION $(E_{xy}, quality\_criteria)$
7: $h \leftarrow$ CALCULATE_PRISM_HEIGHT(T_{xy}, l, l_h, c)
8: **if** $c \neq 1$ and $h < longest\_edge\_length$ **then**
9: $\quad longest\_edge\_length \leftarrow h$
10: $\quad quality\_criteria \leftarrow 2d$ triangle quality criteria with adjusted triangle $longest\_edge\_length$
11: $\quad T_{xy} \leftarrow$ 2D_CONSTRAINED_TRIANGULATION $(E_{xy}, quality\_criteria)$
12: **end if**
13: **for** $i = 1$ to l **do**
14: $\quad$ Extrude and create prisms for layer i using $2d$ triangles in T_{xy} and height h.
15: **end for**

CALCULATE_PRISM_HEIGHT(T_{xy}, l, l_h, c)

1: **switch** (c)
2: **case** 1**:**
3: $h \leftarrow$ height calculated as in Step 6 of BASIC_PRISM_MESHING_ALGORITHM
4: **case** 2**:**
5: $h \leftarrow$ average optimal prism height in T_{xy} using Eq. 3
6: **case** 3**:**
7: $h \leftarrow$ maximum optimal prism height in T_{xy} using Eq. 3
8: **case** 4**:**
9: $h \leftarrow$ minimum optimal prism height in T_{xy} using Eq. 3
10: **case** 5**:**
11: $h \leftarrow$ minimum layer height
12: **end switch**
13: **return** h

This new method gives users an opportunity to tune prism mesh quality over the given model structure based on optimal heights of prism meshing. The difference between cases 1 and 5 in CALCULATE_PRISM_HEIGHT is that case 1 does not have the feedback into CGAL triangulation criteria that case 5 has. Cases 2 through case 5 provide feedback for $longest\_edge\_length$ criteria to refine triangle

Table 2: Resources for Prism Meshing

Resources	List
Operating System	Windows 7: 64-bit
Machine	PC
Programming Language	C++ (STL)
Development Environment	Microsoft Visual Studio 2005
Existing Libraries	Qt 4.6.2 Desktop Edition
	CGAL 4.0
	Boost Library 1_51
	VTK 5.8.0

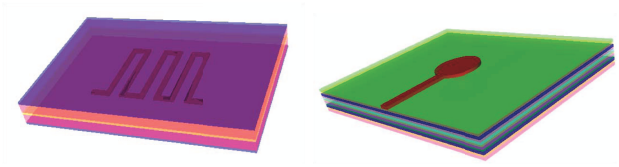

Fig. 5: PCB with single serpentine line (left) and single via (right), used examples in our experiments.

meshing again. Cases 2 through case 5 also give suggestion of more realistic heights for prism meshing. Later on, the prism quality tables show very positive results.

6. Quality Results

We begin by listing in Table 2 the resources that we use to implement and test our algorithm. This includes the visualization toolkit VTK. Section 6.1 describes our test cases and examines their layer heights. Section 6.2 discusses the optimal heights associated with the prisms formed by our new algorithm. Section 6.3 tabulates quality results for our test cases and illustrates visualization results.

6.1 Test Cases

To evaluate our new algorithm we use one single serpentine line PCB and one single via PCB (Figure 5, courtesy of Cadence Design Systems). Since layer height is the basis for cases 1-5 in CALCULATE_PRISM_HEIGHT, we briefly discuss layer height for these 2 examples. The serpentine line resides in a 5 layer structure. Top and bottom layers are the shield layers, while the middle layers are the dielectric layers. Of the 5 layers, the thickness of 3 of them is 1mm and the other 2 are each 4 mm thick. The via model is a 11 layer structure. The 11 layer structure has two shield layers in the middle of the layers; other layers are the dielectric layers. The via model itself has two traces, a drill, two pads and two anti-pads (void or hole features). We approximate a pad using a regular octagon. Two anti-pads are on the shield layers. Two traces are used to connect other devices, via models or transmission line models. The minimum of the layer heights for this via example is 2mm.

6.2 Optimal Prism Heights

The other key ingredient to the revised algorithm in Section 5 is the calculation of optimal prism element height

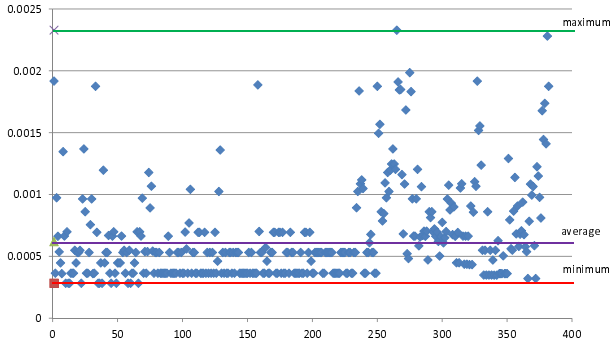

Fig. 6: Scatter plot of optimal prism heights using Eq. 3, in centimeters, for example in Figure 5, using 20.7 degree angle bound and 2mm longest edge length criteria in CGAL. There are 382 triangles (horizontal axis).

for cases 2-4. In our two examples the optimal prism heights are calculated using Eq. 3. For a 20.7 degree angle bound and 2mm longest edge length criteria in CGAL we discuss optimal prism height results. A scatterplot of optimal prism height for the triangles in the serpentine line case appears in Figure 6. In this case, there are 382 triangles. The minimum height is 0.282843 mm, the maximum height is 2.32849 mm and the average height is 0.659132 mm. 62.8% of optimal triangle heights in this case are smaller than the average. In the single via case, we obtain minimum optimal prism height of 0.019614 mm, maximum height of 1.18 mm, and the average is 0.196417 mm. Similar to the serpentine line case, the majority (68.8%) of optimal triangle heights in the single via example are smaller than the average. In the via case, the layer heights are all smaller than the average optimal prism height; this will influence the choice of case in CALCULATE_PRISM_HEIGHT.

6.3 Quality and Visualization Results

Mesh quality and visualization results for the 5 cases in CALCULATE_PRISM_HEIGHT for our serpentine line and via cases appear in Tables 3 and 4 and Figures 7 and 8 (both images courtesy of Cadence Design Systems). The last column is average prism quality across the mesh. In both of our examples the best case is always better than case 1. The percentage improvement is 94.4% and 87.5%, respectively. In the serpentine line example, the best triangle and prism quality is provided by case 4, which uses minimal optimal prism height across the triangles. Note that the number of triangles and prisms is of moderate size, which facilitates fast mesh visualization (144 seconds).

In the via example, the best triangle and prism quality is provided by case 5, which uses minimum layer height. Unfortunately, in this situation the number of triangles and prisms is quite large, which presents a challenge for mesh visualization. We plan to address this in future work.

Table 3: Single serpentine line results for example in Figures 5 and 7.

Case	Longest Edge (meters)	# Triangles	Triangle Quality	# Prisms	Prism Quality
1	0.002	382	0.845355	3056	0.359807
2	0.000659132	717	0.899001	2868	0.594523
3	0.00232849	382	0.845355	1524	0.521804
4	0.000282843	3350	0.907986	13400	0.699492
5	0.0001	27121	0.911653	108484	0.555146

Table 4: Single via results for example in Figures 5 and 8.

Case	Longest Edge (meters)	# Triangles	Triangle Quality	# Prisms	Prism Quality
1	0.002	324	0.8088	3888	0.3728
2	0.000196417	1097	0.8927	13164	0.3177
3	0.00118	324	0.8088	3888	0.3728
4	0.000019614	90376	0.9142	1084512	0.68273
5	0.00002032	84199	0.9143	1010388	0.699

7. Conclusion

The goal of new prism meshing approach is to improve the quality of meshing. Here, we obtain optimal heights of prism elements by maximizing prism quality. Our experimental results show significant quality improvement of the revised method. In the future we plan to expand our set of test cases beyond the single serpentine line and via feature cases to multiple serpentine lines and coupled vias. Also, as indicated in Section 6.3, in some cases the number of elements can be large enough to induce visualization difficulty with our use of the visualization toolkit VTK. To address this, in future work we plan to investigate using VTK in a parallel environment. The authors thank Michelle Daniels for helpful suggestions on prism quality sensitivity.

References

[1] R. Tummala, "SOP: What is it and why? A new microsystem-integration technology paradigm-moore's law for system integration of miniaturized covergent systems of the new decade," *IEEE Transactions on Advanced Packaging*, vol. 27, no. 2, pp. 241–249, 2004.

[2] J. Thompson and N. Weatherill, Eds., *Handbook of Grid Generation*. CRC Press, 1999.

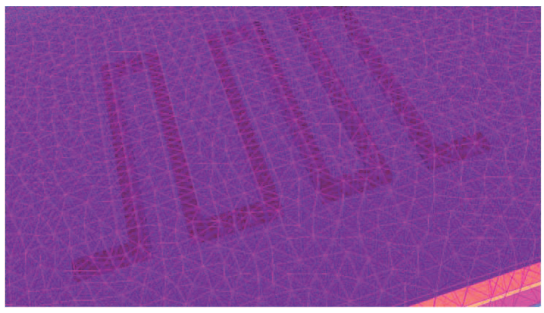

Fig. 7: PCB single serpentine line mesh result from Figure 5 with case 4 and zoomed in view.

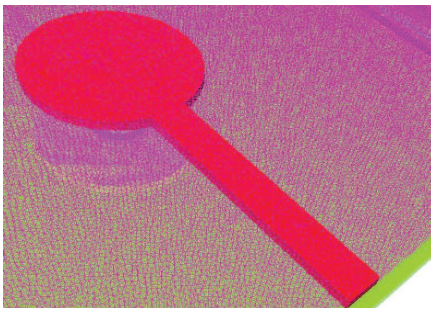

Fig. 8: PCB single via partial mesh result from Figure 5 with case 5 and zoomed in view. 30000 out of 1010388 prisms are supplied to VTK.

[3] I. Tsukerman, "A General Accuracy Criterion for Finite Element Approximation," *IEEE Transactions on Magnetics*, vol. 34, no. 5, pp. 1–4, 1998.

[4] P. Frey and P.-L. George, Eds., *Mesh Generation: application to finite elements*. Paris: Oxford and HERMES Science Publishing, 2000.

[5] M. Botsch and et al., "Course 23: Geometric modeling based on polygonal meshes," in *Proceedings of ACM SIGGRAPH*, 2007.

[6] J. Goodman and J. O'Rourke, Eds., *Handbook of Discrete and Computational Geometry, second ed.* CRC Press, 2004.

[7] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer, 2008.

[8] H. Edelsbrunner, *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001.

[9] S. Lee, "Efficient Finite Element Electromagnetic Analysis for High-Frequency/High Speed Circuits and Multiconductor Transmission Lines," Doctoral thesis, University of Illinois at Urbana-Champaign, Urbana Illinois, 2009.

[10] C.-T. Hwang and et al., "Partially Prism-Gridded FDTD Analysis for Layered Structures of Transversely Curved Boundary," *IEEE Transactions of Microwave Theory and Techniques*, vol. 48, no. 3, pp. 339–346, 2000.

[11] D. Rodger and et al., "Finite Element Modeling of Thin Skin Depth Problems using Magnetic Vector Potential," *IEEE Transactions on Magnetics*, vol. 33, no. 2, pp. 1299–1301, 1997.

[12] R. Whitaker, R. Kirby, and Z. Fu, "A Relaxation Method for Surface-Conforming Prisms," in *Proceedings of the 17th International Meshing Roundtable, Research Note*, 2008.

[13] S. Ye and K. Daniels, "Triangle-based Prism Mesh Generation for Electromagnetic Simulations," in *Research Note for the 17th International Meshing Roundtable*, Pittsburgh, Pennsylvania, 2008.

[14] M. Yvinec and et al., "CGAL User and Reference Manual, URL = http://www.cgal.org."

[15] J. Shewchuck, "Updating and constructing constrained delaunay and constrained regular triangulations by flips," in *19th Annual ACM Symposium on Computational Geometry*, San Diego, California, 2003, pp. 181–190.

[16] ——, "Triangle software. URL = http://www.cs.cmu.edu/∼quake/triangle.html."

[17] S. Yamakawa and K. Shimada, "Converting a Tetrahedral Mesh to a Prism-Tetrahedral Hybrid Mesh for FEM Accuracy and Efficiency," in *Proceedings of the ACM Solid Modeling and Physical Modeling Symposium*, 2008, pp. 287–294.

[18] ——, "Automatic All-Hex Mesh Generation of Thin-Walled Solids via a Conformal Pyramid-less Hex, Prism, and Tet Mixed Mesh," in *Proceedings of the 20th International Meshing Roundtable*, 2011.

[19] E. Holzbecher and H. Si, "Accuracy Tests for COMSOL - and Delaunay Meshes," in *Proceedings of the COMSOL Conference 2008. URL = http://cds.comsol.com/access/dl/papers/5436/Holzbecher.pdf*, Hanover, 2007.