1B.3

Well-centered Planar Triangulation – An Iterative Approach

Evan VanderZee[1], Anil N. Hirani[2], Damrong Guoy[3], and Edgar Ramos[4]

[1] vanderze@uiuc.edu, Department of Mathematics
[2] hirani@cs.uiuc.edu, Department of Computer Science
[3] guoy@uiuc.edu, Center for Simulation of Advanced Rockets
[4] eramosn@cs.uiuc.edu, Department of Computer Science

University of Illinois at Urbana-Champaign

Summary. We present an iterative algorithm to transform a given planar triangle mesh into a well-centered one by moving the interior vertices while keeping the connectivity fixed. A well-centered planar triangulation is one in which all angles are acute. Our approach is based on minimizing a certain energy that we propose. Well-centered meshes have the advantage of having nice orthogonal dual meshes (the dual Voronoi diagram). This may be useful in scientific computing, for example, in discrete exterior calculus, in covolume method, and in space-time meshing. For some connectivities with no well-centered configurations, we present preprocessing steps that increase the possibility of finding a well-centered configuration. We show the results of applying our energy minimization approach to small and large meshes, with and without holes and gradations. Results are generally good, but in certain cases the method might result in inverted elements.

1 Introduction

A *well-centered* mesh is a simplicial mesh in which each simplex contains its circumcenter. A 3D example is a tetrahedral mesh in which the circumcenter of each tetrahedron lies inside it, and circumcenter of each triangle face lies inside it. In this paper we address the case of planar well-centered triangulations, i.e., triangle meshes in which each triangle is acute angled. Typical meshing algorithms do not guarantee this property. For example, a Delaunay triangulation is not necessarily well-centered. We present an iterative energy minimization approach in which a given mesh, after possible preprocessing, is made well-centered by moving the internal vertices while keeping the boundary vertices and mesh connectivity fixed. The preprocessing step may add vertices or change the mesh connectivity, as this is sometimes necessary to permit a well-centered mesh.

A well-centered (primal) mesh has a corresponding dual mesh assembled from a circumcentric subdivision [14]. For an n-dimensional primal mesh, a k-simplex in the primal corresponds to an $(n-k)$-cell in the dual. In a well-centered planar triangle mesh, the dual of a primal interior vertex is a convex polygon with bound-

ary edges that are orthogonal and dual to primal edges. This orthogonality makes it possible to discretize the Hodge star operator of exterior calculus [1] as a diagonal matrix which simplifies certain computational methods for solving partial differential equations. Some numerical methods that mention well-centered meshes in this context are the covolume method [17] and discrete exterior calculus [8, 14]. Well-centered meshes are not strictly required for these or other related methods. However, some computations may be easier if such meshes were available. Another example from scientific computing is space-time meshing. When tent-pitching methods for space-time meshing were first introduced, the initial spatial mesh was required to be well-centered [19]. More recently, this requirement has been avoided, although at the expense of some optimality in the construction [12].

2 Previous Results

We are mostly concerned with planar triangulations where the domain is specified by a polygonal boundary or, more general, a *straight line graph*. In addition to the triangulations being acute, we are also mostly interested in *quality triangulations* in which a lower bound on the triangle angles is achieved. Relevant work can be divided into constructive and iterative approaches.

Constructive approaches start with the specified input boundary/constraints and generate additional points and a corresponding triangulation. Normally a point is committed to a position and never moved afterwards. An algorithm for non obtuse triangulations based on circle packings is described in [3], and more recent works describe improved constructions while also describing how to derive an acute triangulation from a non obtuse one [15, 20]. In fact, these algorithms aim to achieve a triangulation of size linear in the input size, and so the smallest angle can be arbitrarily close to zero. It is not straightforward to extend this class of algorithms to acute quality triangulations. There are also algorithms that achieve acute quality triangulations for limited domains for point sets [4], or non obtuse quality triangulations [16]. Also relevant is an algorithm that, given a constraint set of both points and segments, finds a triangulation that minimizes the maximum angle [11], without adding points. If an acute triangulation exists for the input constraints, the algorithm will find it, otherwise it fails. Thus, there appears to be a no complete constructive solution for the generation of acute quality triangulations.

On the other hand, there are *iterative* or *optimization* approaches which allow an initial triangulation (possibly the canonical Delaunay) and then move the points while possibly changing the connectivity. This is the class of algorithms in which we are primarily interested: there are well-known algorithms to generate quality triangulations [10, 18] for which reliable implementations exist and so they are good candidates as starting points for iterative approaches that seek to achieve acute angles. In this class there are optimization approaches like centroidal Voronoi diagrams [9] and variational triangulations [2]. Each approach has a global energy function that it attempts to minimize through an iterative procedure that alternates between updating the location of the mesh vertices and the triangulation of those vertices. The energy functions optimized in these approaches are designed to optimize certain qualities of the mesh elements, but they do not explicitly seek well-centered simplices. Though

these methods appear to produce nice experimental results, there is no guarantee that they construct quality triangulations, much less acute ones. In fact, in Sect. 5 we show examples in which the converging triangulation is not acute. Also, only limited convergence results are known (there are indeed local minima that can be reached). In addition to the optimization approaches that work directly with a mesh, there are several algorithms that generate circle packings or circle patterns by optimizing the radii of the circles. In particular the algorithms for creating circle patterns that were proposed in [7] and [5] can be adapted to create triangulations. These algorithms produce circle patterns that have specified combinatorics but they do not permit a complete specification of the domain boundary. Thus they are not appropriate to our purpose.

3 Iterative Energy Minimization

Given a simplicial mesh of a two-dimensional planar domain, we iteratively modify the mesh guided by minimizing a cost function defined over the mesh. We'll refer to the cost function as *energy*. Our method is somewhat similar to the methods of [9] and [2] in that it uses an iterative procedure to minimize an energy defined on the mesh, but it differs in that the mesh connectivity and boundary vertices remain fixed as the energy is minimized. We also minimize a different energy, one designed to achieve well-centeredness. We next describe this energy, which is the main component of our method. In Sect. 4 we describe preprocessing steps that may add vertices and change the connectivity of the mesh before energy minimization is begun.

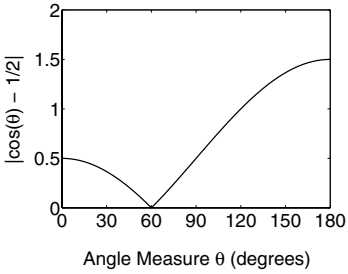

Fig. 1. Graph of $|\cos(\theta) - 1/2|$

Given a simplicial mesh $\mathcal{M}$ of a planar region, the mesh is well-centered if and only if every angle θ in every triangle in the mesh has angle measure strictly less than 90°. This suggests minimizing the maximum angle in the mesh. The cosine of an angle, which can be computed with a dot product and division by vector norms, is easier to compute from a mesh than the actual angle, so a good alternative to directly minimizing the maximum angle is to minimize the maximum $|\cos(\theta) - 1/2|$ over all angles θ. As the graph of $|\cos(\theta) - 1/2|$ in Fig. 1 shows, if any angle of the mesh is obtuse, then the largest angle of the mesh will dominate the expression, but when the mesh is well-centered and all angles are acute, the expression penalizes small angles as well as large ones. This is an added benefit, since small angles may be considered poor quality in some scientific computing applications. We also note the nice feature that a mesh composed of entirely equilateral triangles achieves the minimum energy, taking a value of 0. The key feature, though, is that the energy maintains the property that any acute triangulation is of lower energy than any nonacute triangulation.

As stated, this energy faces several difficulties, in particular some problems with differentiability. In many cases the gradient is not well-defined in areas of the mesh away from the current maximum angle. This problem could probably be addressed by using techniques of nonsmooth analysis such as the taking the minimum norm member of a generalized gradient [6], but a simpler alternative is to note that

$$\lim_{p\to\infty}\left(\sum_i |\cos(\theta_i) - 1/2|^p\right)^{1/p} = \max_i |\cos(\theta_i) - 1/2|\,,$$

and make a reasonable modification of the energy to use, for some finite p, the energy

$$E_p(\mathcal{M}) = E_p(\mathcal{V}, \mathcal{T}) = \sum_{\theta\in\mathcal{M}} |\cos(\theta) - 1/2|^p\,. \tag{1}$$

The mesh $\mathcal{M}$ is defined by the collection of vertex locations $\mathcal{V}$ and collection of triangles $\mathcal{T}$. The sum here is taken over every angle θ of every triangle $T \in \mathcal{T}$.

Unfortunately, in taking a fixed p we lose the property that every well-centered mesh is of lower energy than any mesh with an obtuse angle. On the other hand, using E_p we gain some sense of the quality of a triangle in regions of the mesh away from the maximum angle. Moreover, we know that for any particular domain and mesh connectivity that admit a well-centered configuration of the vertices, there is a finite p such that a given well-centered configuration is of lower energy than any non-well-centered arrangement of the vertices. It is computationally infeasible to compute E_p as p approaches infinity, but globally minimizing E_4, E_6, E_8 or some combination of them has sufficed in our experiments to date. Note that an even power p is preferable, since that makes it unnecessary to explicitly take the absolute value.

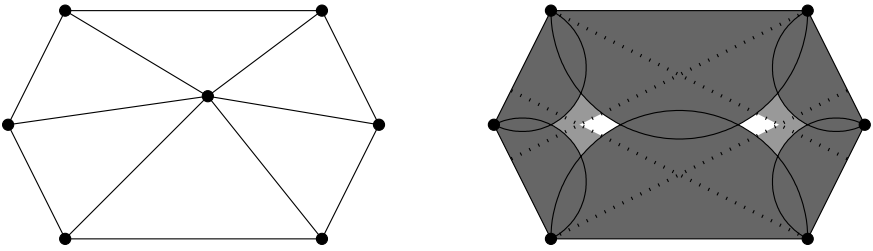

Fig. 2. Necessity of a Nonconvex Energy

The energy we have defined has the undesirable feature of being nonconvex. For energy minimization, one might hope to develop an energy that is convex or at least has a unique minimum. It is not possible, though, to define an energy that accurately reflects the goals of well-centered meshing and also has a unique minimum. Consider the mesh shown on the left in Fig. 2 where only the interior vertex may move. We want to relocate the interior vertex and obtain a well-centered mesh. The right side

of Fig. 2 shows the constraints on where the vertex can be placed to produce a well-centered mesh. The lighter gray regions are forbidden because placing the interior vertex there would make some boundary angle nonacute. (The dotted lines show the four boundary angles that are most important in defining this region.) The darker gray regions, which overlay the lighter gray regions, are forbidden because placing the interior vertex there would make some angle at the interior vertex nonacute.

If the interior vertex is placed in either of the two small white regions that remain, the mesh will be well-centered. We see that the points permitted for well-centeredness form a disconnected set in $\mathbb{R}^2$. Moreover, the mesh is radially symmetric, so there is no way to create an energy that prefers one white region over the other unless we violate the desired property that the energy be insensitive to a rotation of the entire mesh. Any symmetric energy that has minima in only the white regions must have at least two distinct global minima and is not convex.

In most planar meshes there is an interior vertex v that has exactly six neighbors, all of which are interior vertices. If all interior vertices are free to move, as we assume in the method we propose, then the six neighbors of v can be rearranged to match a scaled version of the boundary of the mesh in Fig. 2. Moving v around when its neighbors have such a configuration should exhibit nonconvexity in whatever energy for well-centeredness we might define. One certainly can define convex energies on a mesh, but for the mesh in Fig. 2, no symmetric strictly convex energy will have a minimum at a well-centered mesh.

4 Neighborhood Improvement

In some cases the mesh connectivity or the fixed boundary vertices are specified in a way that no well-centered mesh exists. A simple example of this is an interior vertex with fewer than five neighbors. Any such vertex has some adjacent angle of at least 90°. Similarly, boundary vertices need to have enough interior neighbors to divide the boundary angle into pieces strictly smaller than 90°. We will refer to a vertex that does not have enough neighbors as a *lonely* vertex.

To address the problem of lonely vertices we propose a preprocessing step involving three simple operations for increasing the valence of vertices in the mesh. By applying these three operations – edge flip, edge split, and triangle subdivision – one can guarantee that no mesh vertex is lonely. The authors plan to include a formal proof of that in a sequel paper, though a thoughtful consideration of the double triangle subdivision (Sect. 4.3) may make it clear to the reader. The condition that each vertex have enough neighbors is not sufficient to guarantee that a well-centered arrangement of vertices exists. In our experience this situation – in which there is no lonely vertex and no well-centered configuration possible – appears to occur rarely in meshes we encountered in practice. This issue, too, will be addressed in a future paper. We next describe the three preprocessing steps and an algorithm for applying these steps in a way that limits the number of new vertices introduced.

4.1 Edge flip

Figure 3 is a graphical illustration of the well-known edge flip operation. The initial mesh is shown on the left, with the edge that will be deleted drawn as a dotted line

rather than a solid one. The initial mesh has exactly one lonely vertex. The final mesh is displayed at right, with the edge that replaced the deleted one shown as a dotted line. The final mesh has no lonely vertices, and is, in fact, a well-centered mesh.

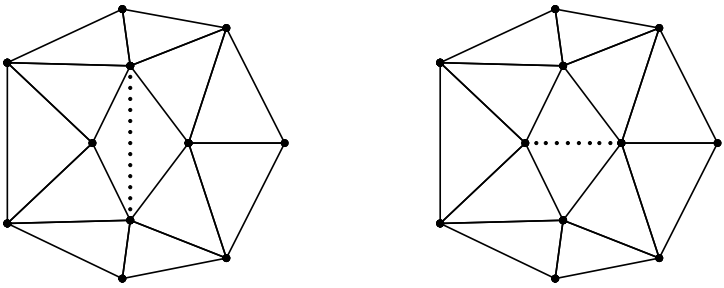

Fig. 3. Edge Flip

In the edge flip operation there are two vertices whose valence decreases and two vertices whose valence increases. We permit an edge flip for increasing valence of a lonely vertex only if no new lonely vertices are introduced. For example, if the vertices whose valence decreases are both interior vertices, they must initially have valence at least six. We also require that the edge-flip not introduce an inverted triangle. Triangle inversion is rarely a problem, but it is possible.

4.2 Edge split

The edge split operation shown in Fig. 4 is a more versatile operation than the edge flip. In the initial mesh shown at left, the edge that will be deleted through the edge split operation is drawn as a dotted line. There is exactly one lonely vertex in the initial mesh. In the final mesh shown on the right, there are three new edges (the dotted lines), and one new vertex (the empty circle). The final mesh is not well-centered, but it has no lonely vertices, and it is quite easy to find a well-centered configuration of the mesh by relocating the new vertex. Note that we do not permit an edge flip operation in this case since all the candidate edges for flip have an endpoint with valence five.

For definiteness of location, the new vertex is introduced at the midpoint of the edge. The edge being split and the one being deleted both have to be interior edges (we do not want to split a boundary edge this way, because the new vertex would be lonely, having only one interior neighbor). Thus the new vertex is always an interior one, and it always gets exactly five neighbors, so it is not a lonely vertex. Of the vertices that appeared initially there are two whose valence increases and one whose valence decreases. We permit the edge split only if it will not introduce a new lonely vertex and will not introduce an inverted triangle. Again, triangle inversion is rare but possible.

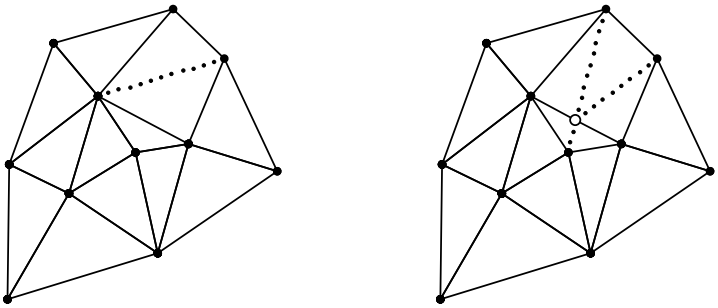

Fig. 4. Edge Split

4.3 Triangle subdivision

The edge split operation is quite versatile, since in the interior of the mesh there is usually a choice of both which edge to split and which endpoint of the edge will decrease in valence. There are some infrequent cases, though, when something more than the edge split may be needed. Except for the small example meshes we created to illustrate these cases, we have not needed anything else in our experiments, but for completeness we propose a third operation, triangle subdivision, that will work in all cases.

Figure 5 shows how a triangle subdivision can be used to increase the valence of a vertex in the mesh. The initial mesh (left) has exactly one lonely vertex. The final mesh (right) has six new edges, shown as dotted lines, and three new vertices, shown as empty circles. The final mesh has some angles that are much larger than the largest angle of the initial mesh, but it has no lonely vertices, and one can obtain a well-centered configuration of the mesh by relocating all of the interior vertices appropriately.

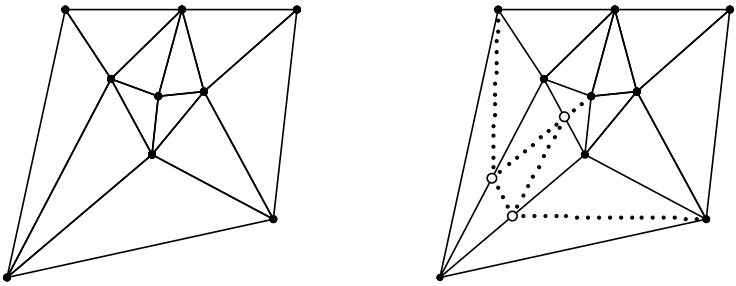

Fig. 5. Triangle Subdivision

If the mesh shown in Fig. 5 were a submesh of a larger mesh, it might be possible to do an edge split to increase the valence of the lonely vertex, but if this submesh occurs with the two edges at the top along the boundary, then no edge split will be

permitted that might increase the number of neighbors of the lonely vertex. No edge flip will be permitted in any case, since every edge we might want to flip has an endpoint that is an interior vertex with valence five.

In the triangle subdivision operation we introduce three new vertices, each at the midpoint of some edge of the initial mesh. We do allow the subdivision of a boundary triangle. Any new interior vertex introduced by the subdivision has valence five, so it is not lonely, and any new boundary vertex will have valence four (two interior neighbors) and a boundary angle of 180°, so it is not lonely either. In the case of meshing a domain with a curved boundary, one might move a new boundary vertex onto the actual boundary with some rule, but unless the boundary is not well resolved, the boundary angle will still measure much less than 270°, so a new boundary vertex will not be lonely.

There is a case when a single triangle subdivision is not sufficient to increase the valence of a lonely vertex. In that case, one can perform a pair of triangle subdivisions to increase the number of neighbors of the lonely vertex. This double triangle subdivision can be performed within a single triangle, so we see that we can increase the valence of any lonely vertex. The double subdivision is illustrated in Fig. 6, where the initial mesh on the left is taken to be the entire mesh, i.e., a mesh of the square with four triangles. Again we see that the final mesh has some very large angles. Increasing the valence of lonely vertices is only a preprocessing step, though, and we note that there is a well-centered mesh with the same connectivity and boundary vertices as the final mesh in Fig. 6.

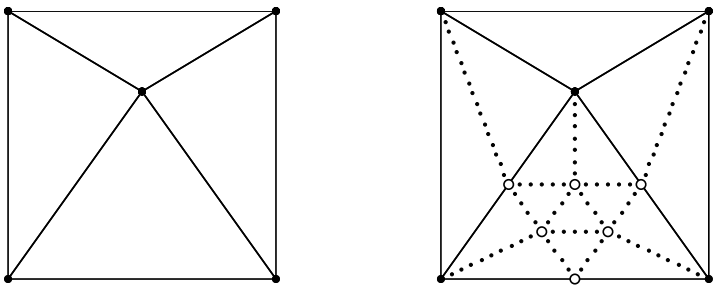

Fig. 6. Double Triangle Subdivision

4.4 Preprocessing algorithm

The preprocessing algorithm in Fig. 7 guarantees that the resulting mesh has no lonely vertices. The result of the preprocessing step is a mesh that can be passed to the energy minimization method. The repeat-until loop of the algorithm is necessary because a mesh may have an interior vertex with only three neighbors. For such a vertex we would need to increase the valence twice to keep the vertex from being lonely.

Using an operation increases the valence of vertices other than the targeted lonely vertex. This can make it possible to perform an operation at another vertex where

```
for each lonely vertex v
   repeat
      perform first permitted operation among:
         edge flip,
         edge split,
         triangle subdivision,
         double triangle subdivision;
   until v is not lonely
end
```

Fig. 7. Preprocessing Algorithm

previously that operation was not permitted. For example, sometimes flipping one edge lets us flip another edge that we were not initially allowed to flip. For this reason it may be advantageous to implement the algorithm in stages. Each stage would look like the algorithm in Fig. 7, but the earlier stages would have a shorter list of operations. For example, one might allow only edge flips in the first stage, then allow edge flips or edge splits in the second stage, and allow any operation in a third stage. In the earlier stages, the until condition would need to be modified to detect the possibility that vertex `v` was unchanged, i.e., that no operation was permitted among the list of allowed operations.

Repeating stages in the algorithm can also be advantageous. Our current implementation is the three-stage example just mentioned, with the first stage repeated twice before moving on to the second stage. We implement the algorithm this way in an attempt to limit the number of new vertices. To limit the number of new vertices even more, one could implement the preprocessing step to continue flipping edges until no more edge flips were allowed, then split an edge and return to flipping edges, etc., using the more versatile operations (and adding vertices) only when the simpler operations cannot be applied anywhere in the mesh. On the other hand, if addition of vertices is not a concern, one can perform a triangle subdivision for each lonely vertex and be done with the preprocessing.

5 Experimental Results

In this section we give some experimental results of applying our energy minimization to a variety of meshes. The algorithm for the preprocessing step has been already described in Section 4 and it was implemented in MATLAB.

We also used MATLAB for the energy minimization, implementing the conjugate gradient method with the Polak-Ribiere formula for modifying the search direction [13]. If $\mathbf{g}_k$ is the gradient at step k and $\mathbf{s}_k$ is the search direction at step k in the conjugate gradient method, then $\mathbf{s}_{k+1} = -\mathbf{g}_{k+1} + \beta_{k+1}\mathbf{s}_k$ for some scalar β_{k+1}. In the Polak-Ribiere formula, we have $\beta_{k+1} = ((\mathbf{g}_{k+1} - \mathbf{g}_k) \cdot \mathbf{g}_{k+1})/(\mathbf{g}_k \cdot \mathbf{g}_k)$ instead of the more common Fletcher-Reeves formula, $\beta_{k+1} = (\mathbf{g}_{k+1} \cdot \mathbf{g}_{k+1})/(\mathbf{g}_k \cdot \mathbf{g}_k)$. For a problem with n free variables, we reset the search direction to the negative gradient after every n iterations. We discovered that MATLAB's `fminbnd` function was not

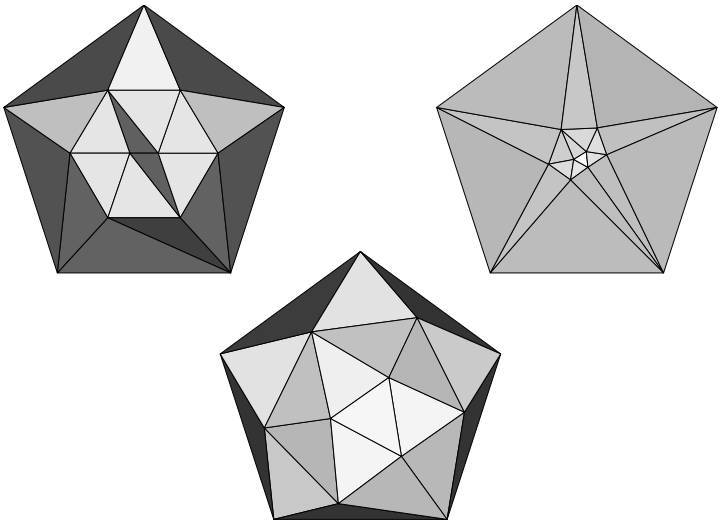

Fig. 8. Regular pentagon on top left is not well-centered. On top right is the well-centered mesh obtained by our method by applying 14 steps of conjugate gradient minimization of E_4. This particular well-centered mesh is not ideal for the finite element method (which is not our target application anyway), but some well-centered meshes produced by our method are appropriate for that application. See for example, the mesh in Fig. 10. We have not yet studied the effects of our algorithm on element aspect ratios. The second row shows the result of applying variational triangle meshing to the initial mesh. We used a boundary fixing variant of the 2D version of the variational tetrahedral meshing algorithm [2]. The resulting mesh shown is not well-centered.

performing a good line search, so we implemented our own (rather expensive) line search. Our line search takes samples of the function on a log scale to determine where the function decreases and where it increases. Then near the minimum we take evenly spaced samples of the function to get better resolution of where precisely the minimum lies. The line search should eventually be replaced by something more efficient. In what follows, *number of iterations* refers to the number of iterations of the conjugate gradient method.

In all the experiments, our MATLAB code can successfully decrease maximum angle below 90° with little or no degradation of minimum angle of the mesh. Some experiments show impressive improvement in both maximum and minimum angle.

Shading scheme: For all the meshes shown hereafter the shading indicates triangle quality with regard to well-centeredness. The shade of a triangle is based on the cosine of the largest angle of the triangle. Darker shade indicates greater largest angle and there is a noticeable jump at 90° so that well-centered triangles can be distinguished from those that are not. For example, the ten triangles that are not well-centered in the initial mesh on the left in Fig. 8 should be easily identifiable.

5.1 Small Meshes

The top row of Fig. 8 shows a test involving a small mesh of a regular pentagon and the well-centered mesh we obtained. Fourteen iterations using the energy E_4, results in the well-centered mesh shown. The final mesh on the right in top row of Fig. 8 has some long, thin isosceles triangles and a rather abrupt change from small triangles in the center to large triangles along the boundary. These features may be unusual compared to an intuitive idea of a nice mesh, but they are permitted in a well-centered planar triangulation, and, in this case, essential to getting a well-centered triangulation with the given boundary vertices and connectivity. We acknowledge that this particular well-centered mesh would not be of high quality for the finite element method, but in some cases, such as the mesh in Fig. 10, making a mesh well-centered improves the aspect ratio of the majority of the mesh elements. We have not yet systematically studied the effects of our algorithm on element aspect ratios, since the finite element method is not the primary motivation of the work on well-centered triangulations.

In Sect. 2 we mentioned variational triangulations. These are based on an iterative energy minimization algorithm introduced in [2] for tetrahedral meshing. We adapted it for comparison with our method. Our adaptation keeps boundary vertices fixed, but is otherwise analogous to the algorithm given in [2]. The bottom row of Fig. 8 shows the result of applying variational triangle meshing to the initial mesh on top left. The result is shown after 10 iterations, which is quite near convergence. The vertices are spread out, and the triangles in the middle of the mesh are nice, but the boundary triangles are all obtuse. The energy used for variational triangle meshing does not detect a benefit of clustering the interior vertices near the center of the pentagon. We tried variational triangle meshing for several of our meshes, sometimes obtaining good results and sometimes not. In most of the failures the method converged to a mesh containing at least one lonely vertex.

We also tried Centroidal Voronoi Tessellation (CVT) [9] for several of our meshes. For our implementation of CVT, we kept boundary vertices fixed, along with any other vertices that had unbounded Voronoi cells. Vertices in the interior of the mesh with bounded Voronoi cells were moved to the centroid of their full Voronoi cells (not clipped by the domain). The preliminary CVT experiments were inconclusive regarding an explanation for why CVT produces a well-centered mesh in some cases and not in others and more analysis is required in this direction. The mesh shown in top left of Fig. 9 is one of the cases for which centroidal Voronoi tessellation did not work. The actual initial mesh is shown on the left in middle row of Fig. 9, but CVT depends on only the vertex positions and uses the Delaunay triangulation of the point set, so we show the Delaunay triangulation and the bounded Voronoi cells in Fig. 9. We see that this mesh is, in fact, a fixed point of our implementation of CVT while being far from well-centered. Note also that the pattern of the mesh can be extended, and it will remain a fixed point of CVT.

There are other extensible patterns that are not well-centered but are fixed points of the CVT variant that clips Voronoi cells to the domain and allows boundary vertices to move within the boundary. Our method yields a well-centered mesh with 30

iterations of energy minimization using E_4 (Fig. 9). That figure also shows how the energy and the maximum angle evolve. The graphs show that the method is nearing convergence at 30 iterations and that decreases in the energy E_4 do not always correspond to decreases in the maximum angle of the mesh.

5.2 Larger Meshes

These first two examples shown in Figures 8 and 9 are for small toy meshes. We have also done experiments with larger meshes. For the largest of these meshes it is difficult to see detail in a small figure, hence for this paper we include only midsize meshes such as the meshes in Fig. 10 and Fig. 11. The largest mesh we have used in our experiments has 2982 vertices and 5652 triangles (about four times as many as the mesh in Fig. 11). That mesh and several others of similar size that we tried all became well-centered without complications.

The initial mesh in the top row of Fig. 10 has some lonely vertices, and the result of preprocessing the mesh with the operations described in Sect. 4 is shown in middle row. The well-centered mesh obtained by 30 iterations minimizing E_4 appears in the bottom row of Fig. 10. Histograms of the minimum and maximum angles of each triangle are included beside each mesh. We see that the preprocessing step introduces several new large angles, but the energy minimization takes care of these, finding a mesh with maximum angle approximately $82.38°$, and having most triangles with maximum angle in the range [62, 76].

5.3 Meshes Requiring Retriangulation

Next, we show a mesh for which our energy could not find a well-centered configuration. However, when we applied our method after a retriangulation of the same set of vertices, we did obtain a well-centered mesh. The initial mesh is show on the top left in Fig. 11. The mesh has no lonely vertices, so we apply the energy minimization directly. After 500 iterations using the energy E_4, we obtain the mesh shown at top center in Fig. 11. The shading shows that the general quality is much improved, but there is a problem. In the top right corner of the result mesh there are some inverted triangles. A zoom on that portion of the mesh is displayed at top right in Fig. 11. Inversion of triangles is rare, since it requires some angle of the mesh to reach 180°, but for the same reason, when inversion does occur, the inverted triangles tend to stay inverted.

The standard method does not work, but there are other ways to get a well-centered mesh. One way is to try a completely different connectivity for the same vertex set. The middle row of Fig. 11 shows the Delaunay triangulation of the two holes mesh after preprocessing has been applied. Along with more than twenty edge flips, the preprocessing step included eight edge splits, which produced the eight groups of bad triangles along the inner boundary. Five hundred iterations with energy E_4 produce a fairly good result with several slightly obtuse triangles, so we follow that with 500 iterations using the energy E_8. The E_8 energy focuses more on the largest angles of the mesh and less on the general quality of the triangles, producing a well-centered result, which appears at middle right in Fig. 11.

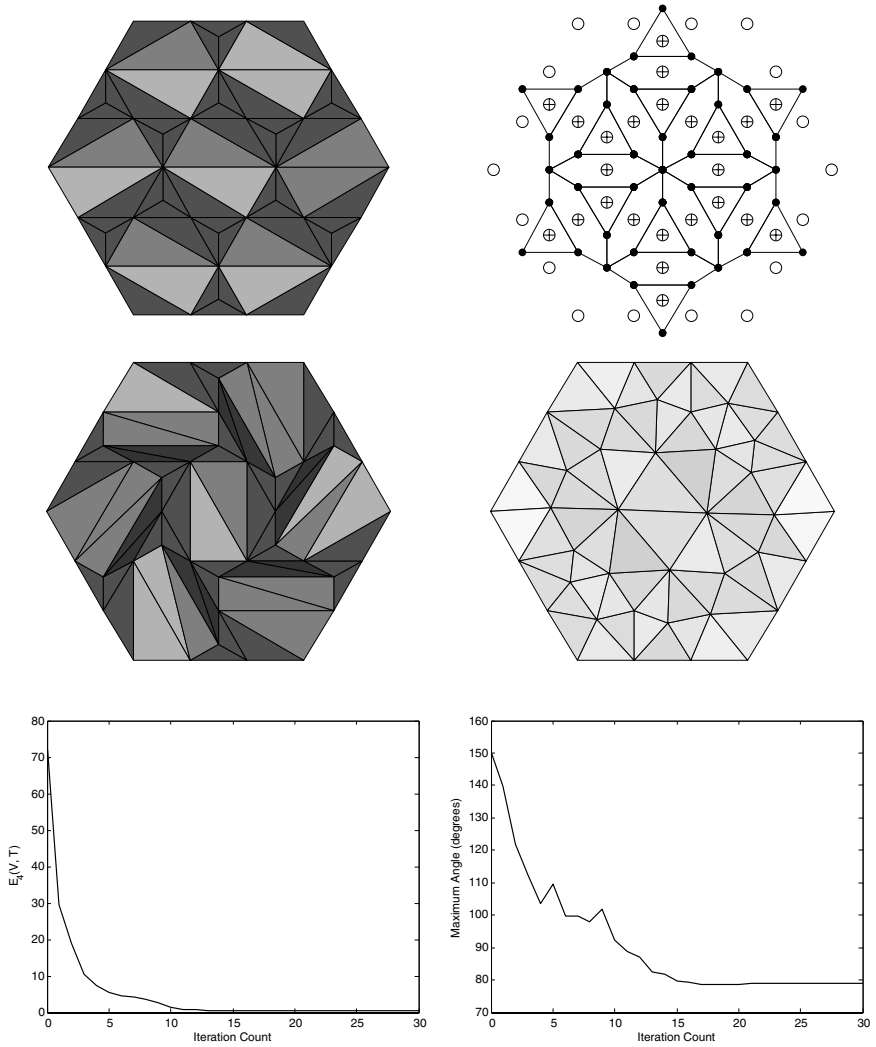

Fig. 9. Mesh shown on top left is a fixed point of the Centroidal Voronoi Tessellation algorithm [9] but it is far from being well-centered. It is the Delaunay triangulation corresponding to the initial mesh which is shown in middle left (CVT uses Delaunay triangulation). Bounded Voronoi cells are shown in top right figure, with vertices denoted by empty circles and centroids of Voronoi cells by plus symbols. Starting with initial mesh in middle left, 30 iterations of our energy minimization using E_4 yields the well-centered mesh in middle right. The evolution of energy and maximum angle observed during energy minimization is shown in the bottom row.

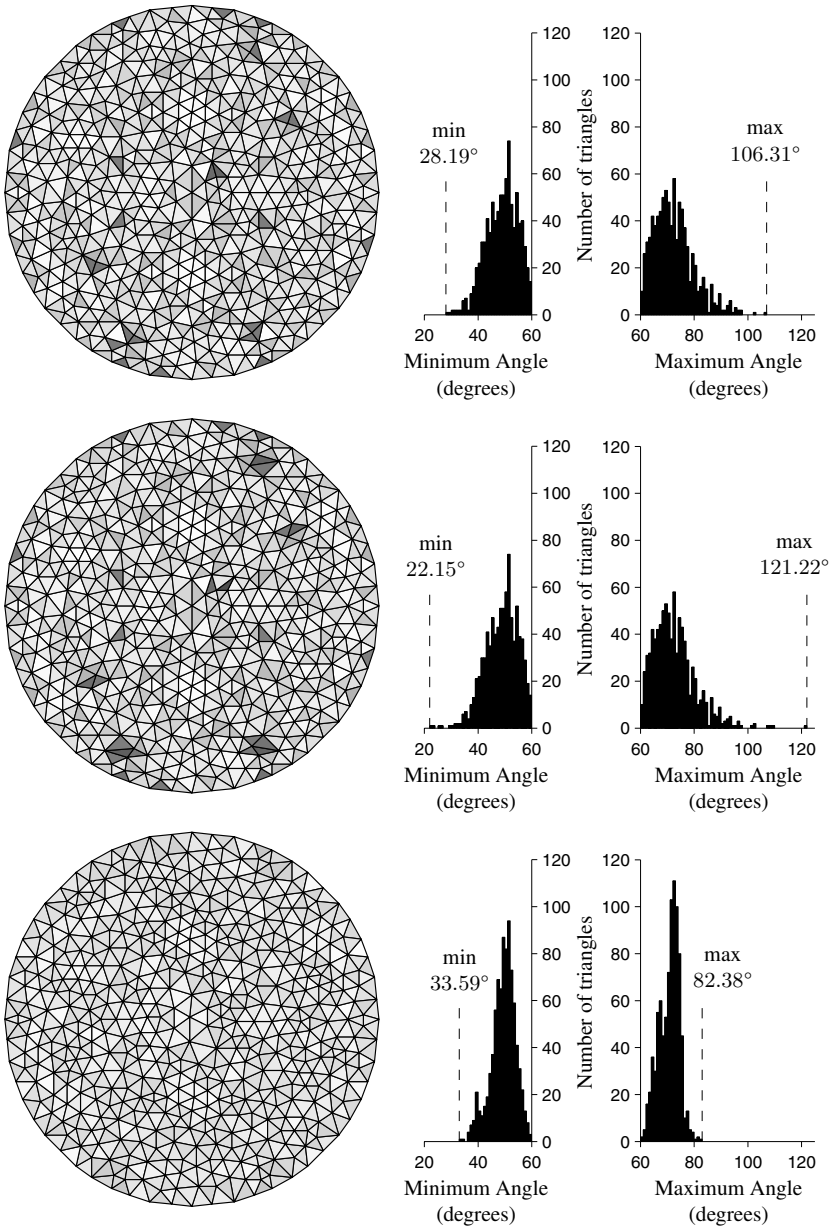

Fig. 10. The initial mesh in the top row has some lonely vertices which are removed by the preprocessing algorithm described in Section 4. The result of preprocessing is shown in middle row. The well-centered mesh resulting from 30 iterations of E_4 minimization is shown in the bottom row. Histograms of the minimum and maximum angles are shown next to each mesh.

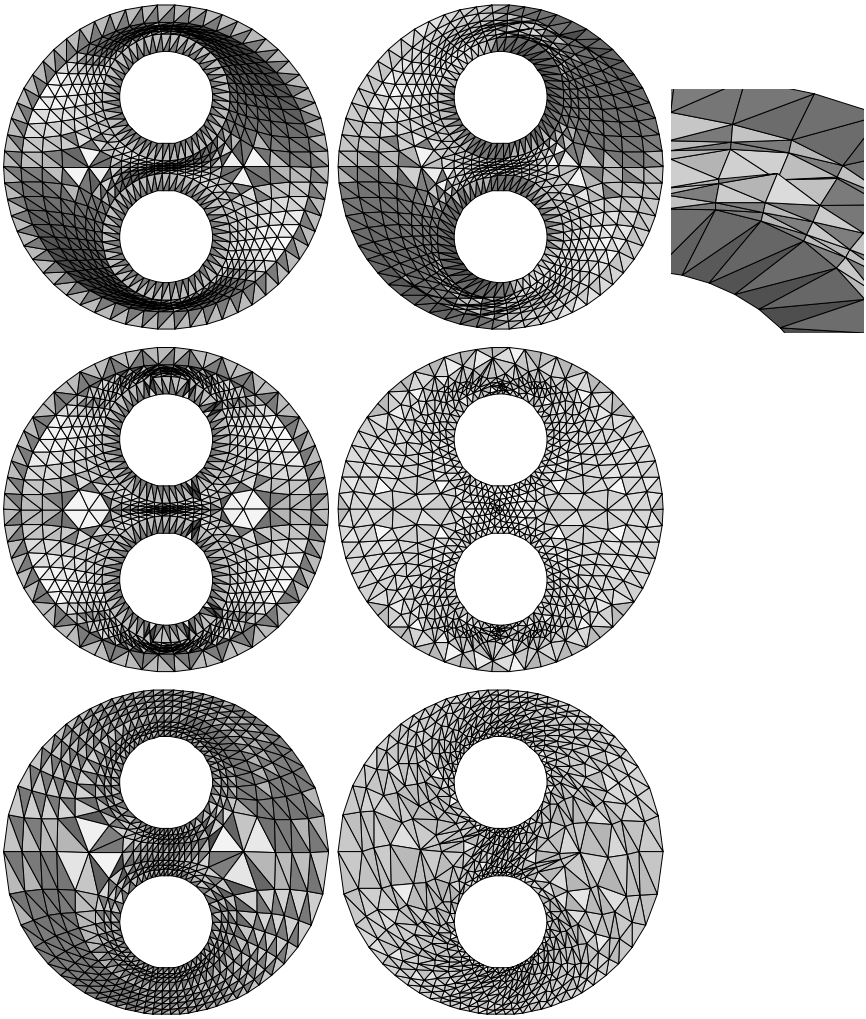

Fig. 11. Energy minimization applied to the two holes mesh on top left does not yield a well-centered mesh. Result after 500 iterations of E_4 minimization is shown at top center. This resulting mesh has some inverted triangles which are shown in the close-up at top right. With a different connectivity for the same vertex set, our minimization does produce a well-centered mesh. This is shown in the middle row. Middle left shows a Delaunay triangulation of the original vertex set after preprocessing. Using this mesh as the initial mesh and applying 500 iterations of E_4 followed by 500 iterations of E_8 minimization yields the well-centered mesh shown in middle right. The bottom row shows the two holes mesh with a different boundary. The mesh has the same mesh connectivity as the initial mesh at top left, but the vertices along the boundary have been moved. The well-centered mesh at bottom right was obtained by applying 100 iterations of E_6 minimization to the mesh at bottom left.

Another way to get a well-centered mesh of the two holes domain is to change the location of the boundary vertices. The mesh on at bottom left in Fig. 11 has the same connectivity as the initial two holes mesh at top left in Fig. 11, but the vertices along the boundary have moved. Instead of being equally spaced, the vertices on the outer boundary are more concentrated at the north and south and more spread out along the east and west. The vertices along the inner boundary have also moved slightly. For this mesh we use the energy E_6, reaching a well-centered configuration by 100 iterations. This mesh appears at bottom right in Fig. 11. The converged result with E_6 actually has one slightly obtuse triangle (90.27°), but there are many iterations during the minimization for which the mesh is well-centered.

5.4 Graded Meshes

The two holes mesh of Fig. 11 is graded. However, the gradation was controlled partly by the presence of the internal boundaries (of holes) and the geometry of the mesh. As a final result we show a mesh obtained by applying energy minimization to a square mesh with an artificially induced gradation. The initial mesh (after a preprocessing step that used only edge flips) appears at left in Fig. 12. The nearly converged result of 50 iterations minimizing E_4 is displayed to its right. From the other experimental results that we have shown, it is clear that the initial size of the triangles of a mesh is not always preserved well. We expect, however, that the energy will generally preserve the grading of an input mesh if the initial mesh is relatively high quality. This hypothesis stems from the observation that the energy is independent of triangle size, the idea that the connectivity of the mesh combined with the property of well-centeredness somehow controls the triangle size, and the supportive evidence of this particular experiment.

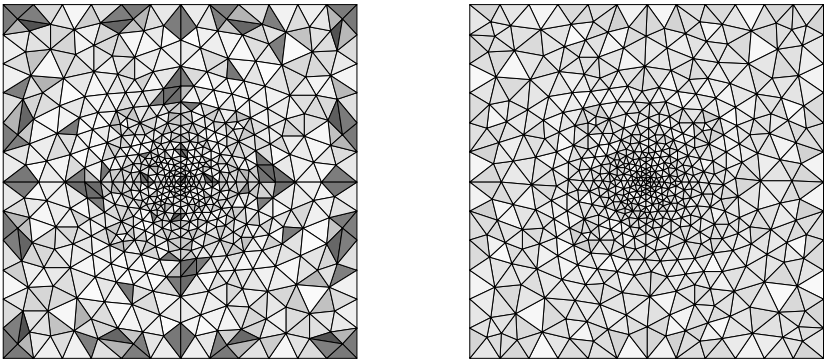

Fig. 12. A graded mesh of the square, in which the gradation is not due to internal boundaries. The initial mesh on left becomes the well-centered mesh on right after 50 iterations of E_4 minimization.

6 Conclusions and Future Work

This paper introduces a new energy function that measures the well-centeredness of planar triangulations. The authors are preparing a paper that describes an extension of this energy to three and higher dimensions. We also intend to do theoretical analysis and proofs of when and how quickly the energy minimization converges to a well-centered configuration. As for optimization, in our experiments shown here we have used relatively simple optimization techniques. It would be good to investigate other options for optimization. In addition, implementation in some language faster than MATLAB may significantly change the execution time of the algorithm, so collecting meaningful data about efficiency remains future work as well.

The current paper also discussed lonely vertices and the general problem that some meshes have no well-centered configuration that preserves both the mesh connectivity and the positions of the boundary vertices. The paper proposes a preprocessing algorithm that eliminates lonely vertices from the mesh. A complete characterization of which meshes permit a well-centered configuration is still lacking, however. Moreover, there may be a more efficient or effective way to perform preprocessing. In particular, the preprocessing algorithm proposed here employs only refinement operators, but it is possible that coarsening operators, such as the edge-collapse operator, might be helpful. An analysis of the number of points added during preprocessing would be interesting as well.

Our experiments shown here demonstrate that the proposed energy function can be effective in finding a well-centered configuration of a mesh. Unfortunately, inverted elements may be introduced and avoiding or handling inversions is a topic for future research. Additional experiments with larger, more complex meshes are also planned. A systematic study of the change in aspect ratio would be worthwhile as well. Although the finite element method is not a primary motivation for the work on well-centered meshes, it is possible that the method could be used effectively to improve finite element meshes in some cases.

Acknowledgment

The work of Evan VanderZee was supported by a fellowship from the Computational Science and Engineering Program, and the Applied Mathematics Program of the University of Illinois at Urbana-Champaign. The work of Anil N. Hirani was supported in part by NSF grant DMS-0645604. The authors thank the reviewers for their comments and suggestions.

References

[1] ABRAHAM, R., MARSDEN, J. E., AND RATIU, T. *Manifolds, Tensor Analysis, and Applications*, second ed. Springer–Verlag, New York, 1988.

[2] ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. Variational tetrahedral meshing. *ACM Transactions on Graphics 24*, 3 (2005), 617–625.

[3] BERN, M., MITCHELL, S., AND RUPPERT, J. Linear-size nonobtuse triangulation of polygons. In *Proceedings of the tenth annual ACM Symposium on Computational Geometry* (New York, 1994), ACM Press, pp. 221–230.

[4] BERN, M. W., EPPSTEIN, D., AND GILBERT, J. Provably good mesh generation. *J. Computer and Systems Sciences 48*, 3 (June 1994), 384–409. Special issue for 31st FOCS.

[5] BOBENKO, A. I., AND SPRINGBORN, B. A. Variational principles for circle patterns and Koebe's theorem. *Trans. Amer. Math. Soc. 356* (2004), 659–689.

[6] CLARKE, F. H. *Optimization and Nonsmooth Analysis*. Society for Industrial and Applied Mathematics (SIAM), 1990.

[7] COLLINS, C. R., AND STEPHENSON, K. A circle packing algorithm. *Computational Geometry: Theory and Applications 25*, 3 (2003), 233–256.

[8] DESBRUN, M., HIRANI, A. N., LEOK, M., AND MARSDEN, J. E. Discrete exterior calculus. *arXiv:math.DG/0508341* (August 2005). Available from: `http://arxiv.org/abs/math.DG/0508341`.

[9] DU, Q., FABER, V., AND GUNZBURGER, M. Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review 41*, 4 (1999), 637–676.

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

[11] EDELSBRUNNER, H., TAN, T. S., AND WAUPOTITSCH, R. An $O(n^2 \log n)$ time algorithm for the minmax angle triangulation. In *SCG90: Proceedings of the sixth annual ACM Symposium on Computational Geometry* (New York, 1990), ACM Press, pp. 44–52.

[12] ERICKSON, J., GUOY, D., SULLIVAN, J., AND UNGOR, A. Building space-time meshes over arbitrary spatial domains. In *Proceedings of the 11th International Meshing Roundtable* (2002), pp. 391–402.

[13] HEATH, M. T. *Scientific Computing: An Introductory Survey*, second ed. McGraw-Hill, 2002.

[14] HIRANI, A. N. *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology, May 2003. Available from: `http://resolver.caltech.edu/CaltechETD:etd-05202003-095403`.

[15] MAEHARA, H. Acute triangulations of polygons. *European Journal of Combinatorics 23*, 1 (2002), 45–55.

[16] MELISSARATOS, E. A., AND SOUVAINE, D. L. Coping with inconsistencies: A new approach to produce quality triangulations of polygonal domains with holes. In *SCG '92: Proceedings of the Eighth Annual Symposium on Computational Geometry* (New York, NY, USA, 1992), ACM Press, pp. 202–211.

[17] NICOLAIDES, R. A. Direct discretization of planar div-curl problems. *SIAM J. Numer. Anal. 29*, 1 (1992), 32–56.

[18] RUPPERT, J. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms 18*, 3 (1995), 548–585.

[19] ÜNGÖR, A., AND SHEFFER, A. Pitching tents in space-time: Mesh generation for discontinuous Galerkin method. *International Journal of Foundations of Computer Science 13*, 2 (2002), 201–221.

[20] YUAN, L. Acute triangulations of polygons. *Discrete and Computational Geometry 34*, 4 (2005), 697–706.