A NEW ALGORITHM FOR GENERATING QUADRILATERAL MESHES AND ITS APPLICATION TO FE-BASED IMAGE REGISTRATION

S. Ramaswami[1] M. Siqueira[2,3] T. Sundaram[2] J. Gallier[2] J. Gee[2]

[1]*Rutgers University, Camden, NJ 08102, USA, rsuneeta@camden.rutgers.edu*
[2]*University of Pennsylvania, Philadelphia, PA 19104, USA, {marcelos,tessa}@seas.upenn.edu*
[3]*Universidade Federal de Mato Grosso do Sul, Campo Grande, MS 79070-900, Brazil.*

ABSTRACT

The use of finite element (FE) analysis in the simulation of physical phenomena over the human body has necessitated the construction of meshes from images. Despite the availability of several tools for generating meshes for FE-based applications, most cannot deal directly with the raw pixel-wise representation of image data. Additionally, some are optimized for the construction of much simpler shapes than those encountered within the human body. In this work, we introduce a new algorithm to obtain strictly convex quadrilateral meshes of bounded size from triangulations of polygonal regions with or without polygonal holes. We present an approach to construct quadrilateral meshes from segmented images using the aforementioned algorithm, and a quantitative analysis of the quality of the meshes generated by our algorithm with respect to the performance of a FE-based image registration method that takes image meshes as input.

Keywords: Quadrilateral mesh, mesh generation, finite element method, image registration

1. INTRODUCTION

Finite element analysis (FEA) is a powerful tool for numerically solving differential equations of variational problems that arise during structural modeling in engineering and the applied sciences. An essential prerequisite for the use of FEA is the availability of a mesh over the problem domain. If the problem domain is a subset of the Euclidean plane, triangular or quadrilateral meshes are typically employed. The accuracy of a problem's solution and the efficiency with which it is obtained using a particular FE implementation are highly dependent on a variety of mesh parameters, including element number and shape, and mesh regularity, directionality and grading, [1].

Triangular meshes have been extensively investigated by the meshing community, and their theoretical properties are now well understood. Algorithms for generating provably good triangular meshes of polygonal domains have been proposed, [2]. On the other hand, the generation of good quadrilateral meshes is not as well understood. A few algorithms exist to generate quadrilateral meshes of bounded size, [3, 4, 5, 6], bounded maximum angle, [4], and controlled density and directionality, [7, 8], for polygonal domains. However, there are no known algorithms to generate quadrilateral meshes of arbitrarily complex polygonal domains that are provably guaranteed to simultaneously meet several quality criteria. Yet good quality quadrilateral meshes may be more desirable for certain FE-based applications, such as planar stress-strain analysis, [9].

The use of FE analysis in biomedical research has led researchers to explore the construction of meshes from images, [10]. In two dimensions, image data is represented as a discrete collection of pixels. Although several algorithms have been proposed for generating meshes of arbitrary geometric domains, most of them have been developed and optimized with structural engineering applications in mind. As a result, they are not directly applicable to discrete data, and they may not perform very well when presented with geometrically complex shapes such as the ones encountered in biology.

In this paper we are primarily concerned with the generation

of *strictly convex* quadrilateral meshes from images, and the influence of their quality on the accuracy and performance of a particular implementation of a FE-based image registration method. Strictly convex quadrilateral meshes are meshes in which each of the four angles of every quadrilateral is strictly less than 180^o, and they are the only desirable quadrilateral meshes for FE-based applications.

The main contributions of our work are two-fold. First, we describe a new algorithm for generating strictly convex quadrilateral meshes of provably small size for polygonal domains with or without polygonal holes. In particular, we show that the interior of a polygonal region with holes triangulated with t triangles can be converted into a quadrilateral mesh with at most $\lfloor \frac{3t}{2} \rfloor + 2$ strictly convex quadrilaterals. We also use this algorithm to show that an arbitrary constrained triangulation (i.e., a triangulation in which some edges, called constraint edges, must be included) of t triangles can be converted into a constrained quadrilateral mesh with at most $\lfloor \frac{3t}{2} \rfloor + 3h$ quadrilaterals, where h is the number of connected components in the dual graph of the triangulation. These results improve on previously known bounds on mesh size. Our algorithm runs in $O(t)$ time and space. Second, we provide a quantitative comparison of the accuracy and performance of FE-based image registration when presented with distinct types of meshes from magnetic resonance (MR) images of the human brain (including the ones generated by our aforementioned algorithm).

The remainder of this paper is organized as follows. In Section 2 we introduce some basic concepts related to our work and review related work. In Section 3 we describe the details of our new algorithm for generating quadrilateral meshes of polygonal domains. In Section 4 we discuss an approach to generate meshes from images using algorithms for meshing polygonal domains. In Section 5 we define image registration, briefly describe the FE-based image registration method used here, and present the aforementioned comparative and quantitative analysis. In Section 6 we summarize our results and discuss future work.

2. BACKGROUND AND RELATED WORK

The problem of generating a quadrilateral mesh of a polygonal region $\mathcal{R}$ is more complex than that of producing a triangular mesh. For one thing, if we require the set of vertices of the mesh to be the set of vertices of $\mathcal{R}$, a quadrilateral mesh may not even exist, and the problem of deciding whether or not it exists has been shown to be **NP**-complete for $\mathcal{R}$ with one or more holes, [11]. In addition, the theoretical properties to generate good quality quadrilateral meshes are not as well understood as the ones for producing good quality triangular meshes. These facts have led several researchers to adopt an *indirect approach* to obtaining quadrilateral meshes: the polygonal domain is first triangulated and then the triangulation is converted into a quadrilateral mesh, [3, 12, 5, 7, 13, 8]. This approach relies on the premise that a good quality quadrilateral mesh could be more easily generated from an existing triangulation of the problem domain.

Let $\mathcal{R}$ be a polygonal region with n vertices and k polygonal holes, and let $\mathcal{T}$ be any triangular mesh of $\mathcal{R}$ such that the set of vertices $V_{\mathcal{R}}$ of $\mathcal{R}$ is contained in the set of vertices $V_{\mathcal{T}}$ of $\mathcal{T}$, $V_{\mathcal{R}} \subseteq V_{\mathcal{T}}$. From Euler's relation, we know that $\mathcal{T}$ has $t = 2m + 2k - 2 - m_b$ triangles, where m is the number of vertices of $\mathcal{T}$ and m_b is the number of vertices of $\mathcal{T}$ on its boundary, with $n \leq m_b \leq m$. A very simple algorithm for converting such a triangulation into a strictly convex quadrilateral mesh was proposed by de Berg, [3]. His algorithm runs in $O(t)$ time, produces $3t$ quadrilaterals, and inserts exactly $5m + 5k - 5 - 2m_b$ extra vertices inside $\mathcal{R}$. It is simple and fast, but the size of the output quadrilateral mesh may prevent its practical use in the presence of large input triangular meshes. Everett et al., [3], introduced another linear time algorithm to convert triangular meshes into strictly convex quadrilateral ones that generates at most $\lfloor \frac{8t}{3} \rfloor$ quadrilaterals. However, the size of the output quadrilateral mesh may still be prohibitive in practice. An interesting feature of this algorithm, which is also present in de Berg's algorithm, is the preservation of the input mesh grading. Johnston et al. proposed another indirect approach-based algorithm that uses several heuristics to obtain a strictly convex quadrilateral mesh from a triangulation, [12]. Their algorithm runs in $O(t^2)$ time, and selectively combines adjacent triangles to obtain quadrilaterals. However, it is not clear from the description in [12] that the heuristic procedures are always successful in producing an all-quadrilateral mesh.

Shimada et al., [7], proposed an algorithm for generating quadrilateral meshes that takes into account mesh regularity, directionality and grading as well as element shape. Their algorithm employs a physically-based relaxation process, called cell packing, that fills in the problem domain with squares, whose size and direction are controlled by user-defined, scalar density and vector functions. Mesh vertices are placed at the center of every square and then connected to generate a triangulation of the entire domain. Finally, the triangulation is converted into a strictly convex quadrilateral mesh. Later, Viswanath et al., [8], modified this algorithm by using rectangular cells instead of square cells, which enabled them to generate anisotropic quadrilateral meshes. Both algorithms produce nearly regular quadrilateral meshes with well-shaped elements and precise control over their direction and size distribution. However, if precise control of directionality is not critical and the problem domain has complex geometry, neither algorithm may be very attractive due to the expense of the cell packing approach. Furthermore, conversion of a triangular mesh into a quadrilateral one may leave isolated triangles, which may cause the algorithm to undergo an extra subdivision step to obtain an all-quadrilateral mesh.

Owen et al., [13], presented another quadrilateral meshing algorithm that takes into account directionality and element shape. It converts a triangular mesh into a strictly convex quadrilateral one using advancing fronts initially defined by the boundary edges of the input mesh. Quadrilaterals are

generated by combining and transforming triangles as the fronts move from the boundary to the interior of the input mesh. Local smoothing and topological clean-up, commonly performed as post-processing steps, are part of the conversion process. One limitation of this algorithm is that directionality cannot be arbitrarily specified as in [7, 8]. Although the algorithm in [13] and the ones in [7, 8] do not provide any theoretical bounds on mesh size nor mesh element shape, they can generate very good quality quadrilateral meshes in practical applications.

The algorithms proposed in [3, 12, 13] take a triangular mesh as input, while the ones in [7, 8] take a polygonal region as input and build a triangulation for it. Most of the algorithms for generating triangular meshes, as well as the algorithms in [7, 8], cannot deal directly with image data represented by a discrete collection of pixels. However, using image processing techniques, it is possible to identify several distinct structures within an image and then build polygonal approximations for their boundaries, [14]. By using polygonal approximations rather than pixel-wise representations, we are able to employ any algorithm for generating triangular and quadrilateral meshes of polygonal domains to separately mesh the polygonal approximation of each individual structure on the image.

3. THE ALGORITHM

In this section, we present a new algorithm to convert an arbitrary triangulation, possibly with constrained edges, into a quadrilateral mesh of bounded size. The conversion allows unconstrained edges to be deleted, but does not allow deletion of input points (vertices). New points, called *Steiner points*, may be inserted along with new edges between Steiner points and/or input points, in order to construct the quadrangulation[1]. The mesh produced by our algorithm consists of strictly convex quadrilaterals. First, we show that the interior of a triangulated polygonal region with holes can be quadrangulated with at most $\lfloor \frac{3t}{2} \rfloor + 2$ quadrilaterals by inserting at most $t+2$ Steiner points, where t is the number of triangles. All Steiner points, except possibly one, lie in the interior of the polygonal region. Next, we show how to extend this algorithm to convert a constrained triangulation into a strictly convex quadrangulation satisfying the given constraints. The resulting number of quadrilaterals is at most $\lfloor \frac{3t}{2} \rfloor + 3h$, obtained by using at most $t + 2h$ Steiner points, where h is the number of connected components in the dual graph of the triangulation.

3.1 Polygonal Regions with Holes

The idea behind our algorithm is to quadrangulate a small group of triangles at a time until the input triangulation is converted into a quadrilateral mesh. The group of triangles represents the triangulation of small, simple polygonal regions (the ones with a small constant, ≤ 7, number of vertices). This triangulation is converted into a partial or complete quadrangulation of the polygonal region. By using a spanning tree of the dual graph of the triangulation and processing it in a bottom-up fashion, our algorithm systematically groups triangles together so that no isolated triangles remain in the resulting decomposition.

Let $\mathcal{R}$ and $\mathcal{T}$ be as defined in Section 2. Let $m \geq n$ be the number of vertices of $\mathcal{T}$ and t the number of triangles of $\mathcal{T}$. The first step of our algorithm is to build a rooted spanning tree T of the dual graph G of $\mathcal{T}$. The *dual graph* of $\mathcal{T}$ is the graph that contains a node for every triangle of $\mathcal{T}$ and an edge between two nodes if the corresponding triangles share an (unconstrained) edge. T is built as a breadth-first search (BFS) tree. The root of T is any node corresponding to a triangle containing a boundary edge of $\mathcal{T}$. Note that T is a binary tree. After constructing T, the algorithm computes the set V_l of all nodes of T at level l, for every $l \in \{0, 1, \ldots, d\}$, where d is the depth of T and the root node is the singleton node at level 0. Next, the algorithm processes the nodes of T one level at a time in decreasing order of depth, i.e., $V_d, V_{d-1}, \ldots, V_0$. Let $par(v)$ denotes the parent of $v \in V$, $sib(v)$ the sibling of v, and $ele(v)$ the triangle of $\mathcal{T}$ corresponding to v. Note that $ele(v)$ and $ele(par(v))$ necessarily share an edge of $\mathcal{T}$. For each vertex $v \in V_k$ $(1 < k \leq d)$ we consider the subtree rooted at $par(v)$ or at $par(par(v))$. We denote this subtree by T_v and its root by r_v. Let G_v denote the subgraph of G induced by T_v. We show that in the original triangulation $\mathcal{T}$, the subgraph G_v corresponds to a triangulated polygonal region $\mathcal{T}_v$ consisting of $4, 5, 6,$ or 7 vertices. This triangulation is then converted into a partial or complete quadrangulation by adding Steiner points within the boundary of $\mathcal{T}_v$. If the result is a complete quadrangulation of the domain of $\mathcal{T}_v$, the subtree T_v is eliminated from T. If the quadrangulation is not complete, there will be only one leftover triangle within the boundary of $\mathcal{T}_v$. The root node r_v now represents this triangle and the remaining nodes of T_v are eliminated from T. The sets $V_k, V_{k-1},$ and V_{k-2} are updated accordingly[2].

The algorithm runs in phases. Each phase of the algorithm examines nodes from the three deepest levels and eliminates some of them from T and from the appropriate V_i. We show that for every two nodes eliminated from T during a given phase, at most three quadrilaterals are created by using at most two Steiner points. In other words, we allow the creation of "one and a half" quadrilaterals per triangle in the pruning process. Furthermore, at the end of each phase, the depth of T decreases by at least one. The tree T is thusly pruned until all nodes are eliminated and the underlying triangulation $\mathcal{T}$ is converted into a strictly convex quadrangulation.

[1] Throughout this paper, we use the terms "quadrangulation" and "quadrangulate" to mean "quadrilateral mesh" and "decompose into quadrilaterals", respectively. We also use the term "quad" to refer to a quadrilateral.

[2] This general idea of pruning the dual tree was also used in [5] to convert triangulations into quadrangulations consisting of quads that are not necessarily convex.

Before describing the details of our algorithm, however, we discuss two special situations. First, when T_v is a subtree of three nodes containing v, $r_v = par(v)$ and $sib(v)$, and there is a cross-edge between v and $sib(v)$ in G_v, the triangulated polygonal region $\mathcal{T}_v$ has a point in its interior, as shown in Figure 1. Second, when processing T_v, our algorithm may place a Steiner point s on the edge e between $ele(r_v)$ and $ele(par(r_v))$, as shown in Figure 2.

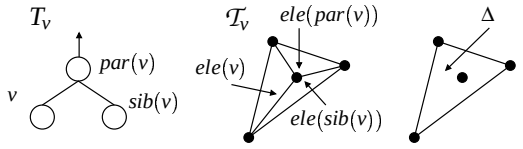

Figure 1: The non-empty triangle Δ.

In the first situation we eliminate v and $sib(v)$ from T, and $par(v)$ now represents *the non-empty triangle* Δ with its interior point. Note that if v is at level k, then the node corresponding to Δ is a leaf at level $k-1$. In the second situation the element $ele(par(r_v))$ corresponds to a degenerate quadrilateral or a degenerate pentagon (if $sib(r_v)$ also adds a Steiner point on the edge it shares with $ele(par(r_v))$) rather than a triangle. We have the following important observations:

Observation 3.1.1. *Since T_v gets eliminated from T when the Steiner point is placed on edge e, degenerate pentagons are leaves of T, and degenerate quadrilaterals are either leaves or internal nodes of degree 2. Furthermore, if v is at level k, the nodes corresponding to these degenerate elements are at level $k-2$ or level $k-3$ (refer to cases 3d, and 4c(ii) in the algorithm description).*

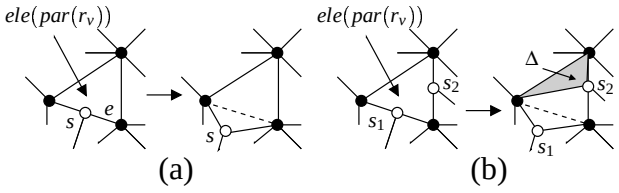

Figure 2: (a) Degenerate quadrilateral. (b) Degenerate pentagon.

Observation 3.1.2. *In all cases when such degenerate elements are created, the number of nodes in T_v is odd. Since we are allowed "one and a half" quadrilaterals per node but only create whole numbers of quadrilaterals, this implies that we have a "credit" of at least "half" a quadrilateral for every Steiner point s on the boundary of a degenerate element.*

Observation 3.1.3. *Since all quadrilaterals in the quadrangulation of the domain of $\mathcal{T}_v$ are strictly convex, there must be an edge of the quadrangulation incident on s and lying outside $ele(par(r_v))$, as shown in Figure 2. We can slightly perturb s along this edge without destroying the strict convexity of the quadrilaterals.*

We now describe the steps involved in each phase of the algorithm. During the course of the description, we refer to various lemmas pertaining to quadrangulations of small polygonal regions, which are stated formally later in Section 3.3. Let $1 < l \leq d$ be the current deepest level of T. We first eliminate all leaves v of T such that $ele(v)$ is a degenerate quadrilateral, degenerate pentagon, or a non-empty triangle. Note that the first two types of leaves will be at levels $l, l-1$, or $l-2$, and the third type will all be at level l.

Step 1. *Eliminate all $v \in V_l \cup V_{l-1} \cup V_{l-2}$ such that v is a leaf and $ele(v)$ is a degenerate quadrilateral.* Let s be the Steiner point of $ele(v)$, and let e_s be the edge of the quadrangulation (constructed thus far) incident on s. We convert $ele(v)$ into a strictly convex quadrilateral by perturbing s along the edge e_s, as shown in Figure 2a, and then we remove v from T.

Step 2. *Eliminate all $v \in V_l \cup V_{l-1} \cup V_{l-2}$ such that $ele(v)$ is a degenerate pentagon.* Let s_1 and s_2 be the two Steiner points of $ele(v)$ and let e be the shared edge of $ele(v)$ and $ele(par(v))$. It is straightforward to convert $ele(v)$ into a strictly convex quadrilateral and a leftover triangle Δ as shown in Figure 2b. Now v represents the leftover triangle, i.e., $ele(v) = \Delta$. Note that we have created one convex quadrilateral, but have not eliminated any nodes from T. However, from Observation 3.1.2, we know that each of s_1 and s_2 has a credit of half a quadrilateral. Thus, the number of quads produced in this case remains within the stated bounds.

Step 3. *Eliminate all $v \in V_l$ such that $ele(v)$ is a non-empty triangle.* Let T_v be the subtree of T rooted at $par(v)$. We consider the following cases (refer to the illustrations in Figure 3 and Figure 4):

3a. *$par(v)$ is a node of degree 2, and $ele(par(v))$ is a degenerate quadrilateral.* Let s be the Steiner point of $ele(par(v))$. By connecting s to the vertex of $ele(par(v))$ that is not adjacent to s, we decompose $\mathcal{T}_v$ into a triangle Δ and a quadrilateral with a point in its interior (see Figure 3a). The latter can be quadrangulated into five convex quads with three Steiner points in its interior by Lemma 3.2.2. We then remove v from T. A total of three nodes (the nodes that gave rise to v) have been eliminated from T. Once again, since s has a credit of half a quadrilateral, the five convex quads created in this step keep us within the stated bounds. The node $par(v)$ now corresponds to the triangle Δ.

3b. *$par(v)$ is a node of degree 2, and $ele(par(v))$ is a triangle.* In this case, the domain of $\mathcal{T}_v$ is a quadrilateral and this quadrilateral contains a vertex of $\mathcal{T}_v$ in its interior (see Figure 3b) . Again, by Lemma 3.2.2, the

region can be quadrangulated into five convex quads with three Steiner points in its interior. We then remove v and $par(v)$ from T. Four nodes have been eliminated from T.

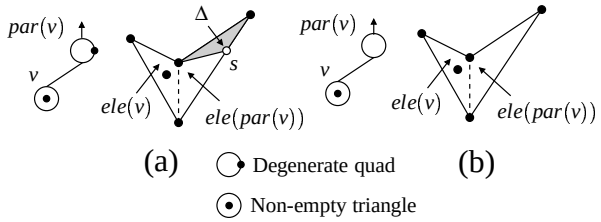

Figure 3: Cases 3a and 3b of Step 3.

3c. *$par(v)$ is a node of degree 3, and $ele(sib(v))$ is a triangle.* If $G_v = T_v$, then the domain of $\mathcal{T}_v$ is a pentagon and this pentagon contains a vertex of $\mathcal{T}_v$ in its interior (see Figure 4a). Then by Lemma 3.2.4, this region can be decomposed into at most six convex quads and one triangle Δ by using at most four Steiner points in the interior. Remove v and $sib(v)$ from T, and let r_v now represent Δ. If G_v contains a cross-edge between v and $sib(v)$, then $ele(v)$ and $ele(sib(v))$ form a quadrilateral with a point inside (see Figure 4a), which can be decomposed into five convex quadrilaterals by using three Steiner points. Eliminate v and $sib(v)$ from T. Four nodes have been eliminated from T in either case.

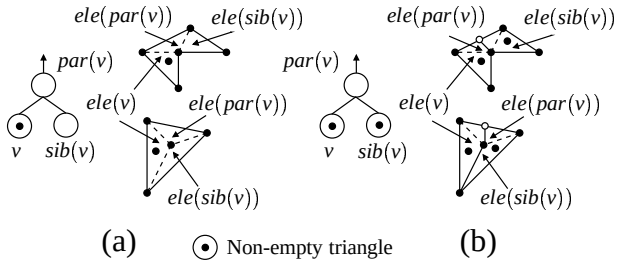

Figure 4: Cases 3c and 3d of Step 3.

3d. *$par(v)$ is a node of degree 3, and $ele(sib(v))$ is a non-empty triangle.* If $G_v = T_v$, then the domain of $\mathcal{T}_v$ is a pentagon and this pentagon contains two vertices of $\mathcal{T}_v$ in its interior. If G_v contains a cross-edge between v and $sib(v)$, then the domain of $\mathcal{T}_v$ is a triangle and this triangle contains three vertices of $\mathcal{T}_v$ in its interior. In either case, $\mathcal{T}_v$ can be decomposed into two quadrilaterals, each with a point in its interior as follows: add a Steiner point on the edge shared by $ele(r_v)$ and $ele(par(r_v))$ and connect it to the vertex of $ele(r_v)$ that is not adjacent to it (see Figure 4b). By Lemma 3.2.2, each quadrilateral can be decomposed into five convex quads using three Steiner points. Remove all nodes of T_v from T. Hence a total of seven nodes were eliminated and ten convex quadrilaterals were created using seven Steiner points. In the next phase of the algorithm, $ele(par(r_v))$ will be a degenerate quadrilateral or pentagon.

Observation 3.1.4. *After steps 1-3 are carried out, for every node $v \in V_l \cup V_{l-1}$, $ele(v)$ is either a triangle or a degenerate quadrilateral. In the latter case, v is a node of degree 2.*

Step 4. The last step eliminates all remaining $v \in V_l$. From Observation 3.1.4, and the fact that T is a BFS tree, it follows that the only possible configurations for T_v are those described in the sub-steps below.

4a. *Eliminate all $v \in V_l$ such that $ele(par(v))$ is a degenerate quadrilateral.* Let T_v be the subtree of T rooted at $r_v = par(v)$. Perturb the Steiner point s of $ele(r_v)$ along the quadrangulation edge incident to it. A Steiner point s' placed in $ele(r_v)$ decomposes the region $\mathcal{T}_v$ into two convex quads and a triangle Δ adjacent to the shared edge of $ele(r_v)$ and $ele(par(r_v))$ (see Figure 5a). Eliminate v from T and let r_v now represent the triangle Δ. The credit of half a quadrilateral on s keeps the number of convex quads within the stated bounds.

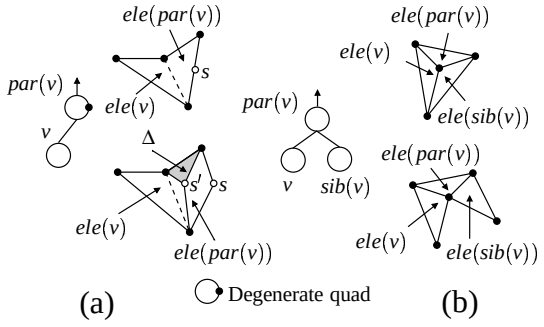

Figure 5: Cases 4a and 4b of Step 4.

4b. *Eliminate all $v \in V_l$ such that $par(v)$ is a node of degree 3.* Again, let T_v be the subtree of T rooted at $r_v = par(v)$ and refer to Figure 5b. If G_v contains an edge between the nodes v and $sib(v)$, then we remove v and $sib(v)$ from T_v. The node $par(v)$ now is the non-empty triangle corresponding to the boundary of $\mathcal{T}_v$, with the fourth vertex of $\mathcal{T}_v$ in the interior of its domain (see Figure 1). If there is no cross-edge between v and $sib(v)$, then by Lemma 3.2.3, the domain of $\mathcal{T}_v$ can be subdivided into two convex quadrilaterals and one triangle Δ (adjacent to the edge shared by $ele(r_v)$ and $ele(par(r_v))$) by adding one Steiner point. Remove v and $sib(v)$ from T. The node r_v now represents Δ.

4c. Finally, *eliminate all $v \in V_l$ such that $par(v)$ is a node of degree 2.* Let T_v be the subtree of T rooted at $r_v =$

$par(v)$. If the domain of the triangulated quadrilateral $\mathcal{T}_v$ formed by $ele(v)$ and $ele(par(v))$ is strictly convex, then we remove v and $par(v)$ from T. Otherwise, let T_v be the subtree of T rooted at $r_v = par(par(v))$ and consider the following sub-cases (refer to Figure 6, Figure 7, and Figure 8):

i. *$ele(r_v)$ is a degenerate quadrilateral.* Note that $G_v = T_v$ for this case because T is a BFS tree. Perturb the Steiner point of $ele(r_v)$ along the quadrangulation edge incident to it so that the domain of $\mathcal{T}_v$ is now a hexagon. By Lemma 3.2.1, this region can be subdivided into at most four convex quadrilaterals by using at most three Steiner points in its interior. Eliminate all the nodes of T_v from T.

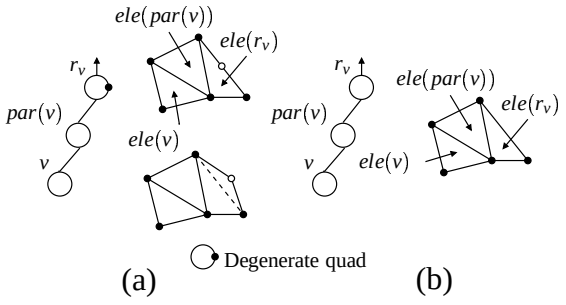

Figure 6: Cases i and ii of Step 4c.

ii. *r_v is a node of degree 2 and $ele(r_v)$ is a triangle.* Once again, $G_v = T_v$ for this case. Therefore, the domain of $\mathcal{T}_v$ is a pentagon. We apply Lemma 3.2.3, where the shared edge of $ele(r_v)$ and $ele(par(r_v))$ is designated as the "outgoing" edge. Then, we have two situations. First, the domain of $\mathcal{T}_v$ is subdivided into three convex quads and one triangle Δ adjacent to the outgoing edge by using two Steiner points. Second, the domain of $\mathcal{T}_v$ is subdivided into four convex quads by using three Steiner points, one of which lies on the outgoing edge. In the first situation, we remove v and $par(v)$ from T, and let r_v now represent Δ. In the second situation, we remove all nodes of T_v from T, and $ele(par(r_v))$ becomes a degenerate quad or a pentagon in the next phase of the algorithm.

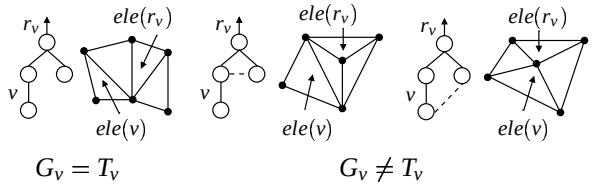

Figure 7: Case iii of Step 4c (dashed edges are cross-edges).

iii. *r_v is a node of degree 3, and $sib(par(v))$ is a leaf.* If $G_v = T_v$, the domain of $\mathcal{T}_v$ is a hexagon, to which we apply Lemma 3.2.1. If $G_v \neq T_v$, then the domain of $\mathcal{T}_v$ is a quadrilateral that contains a vertex of $\mathcal{T}_v$ in its interior, to which we apply Lemma 3.2.2. In either case, at most five convex quadrilaterals are created by using at most three Steiner points in the interior of $\mathcal{T}_v$. Remove all nodes of T_v from T.

iv. *r_v is a node of degree 3, and $sib(par(v))$ is a node of degree 2.* The different possibilities for the graph G_v are derived from the fact that T is a BFS tree. All cases are illustrated in Figure 8: Cases (a)–(c) correspond to a pentagon with a point in its interior, to which we apply Lemma 3.2.4. In cases (d)–(e), the non-root nodes of T_v (v, v_1, v_2 and v_3 in the figure) correspond to a quadrilateral with a point inside, to which we apply Lemma 3.2.2. Finally, case (f) corresponds to a septagon, to which we apply Lemma 3.2.5. In all cases, we remove all four non-root nodes of T_v and at most six convex quadrilaterals are created by adding at most four Steiner points.

After all $d-2$ phases have been carried out, the sets $V_d, V_{d-1}, \ldots, V_2$ are all empty, and the sets V_0 and V_1 are possibly non-empty. If V_0 and V_1 are non-empty, apply Steps 1–3 and Step 4a to V_1 and then V_0. It is easy to show that now the only remaining nodes in T are either the singleton root node, or the root node and one child, or the root node and two children. The domain of $\mathcal{T}$ is thus either a triangle, a quadrilateral, a pentagon, or a triangle with an interior point. In the case of the quadrilateral, four internal Steiner points decompose it into five convex quads. In the other cases, we add one Steiner point outside the boundary of $\mathcal{R}$ (this is unavoidable because the boundary of $\mathcal{R}$ has odd parity) to obtain one, three, or five convex quads using zero, one, or three internal Steiner points, respectively.

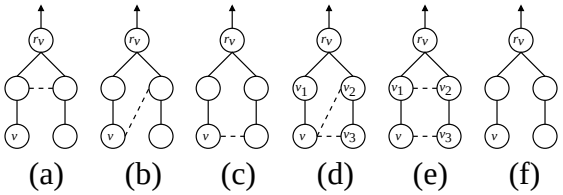

Figure 8: G_v for case iv of Step 4c (dashed edges are cross-edges).

Theorem 3.1.1. *Given a polygonal region $\mathcal{R}$, possibly with one or more polygonal holes, and a triangulation $\mathcal{T}$ of $\mathcal{R}$ with t triangles, the algorithm described before converts $\mathcal{T}$ into a strictly convex quadrangulation of $\mathcal{R}$ with at most $\lfloor \frac{3t}{2} \rfloor + 2$ quadrilaterals by using at most $t+2$ Steiner points, all except one of which lie within the boundary of $\mathcal{R}$. The algorithm runs in $O(t)$ time and space.*

Proof. In each step of phase l of the algorithm described above, at most k Steiner points are added for every k nodes eliminated from T. At the very last step, either one more Steiner point is added just outside the boundary of $\mathcal{R}$, or two more are added within $\mathcal{R}$. Therefore, the total number of Steiner points added is at most $t+2$. Furthermore, for every two nodes eliminated from T, at most three strictly convex quadrilaterals are constructed. At the very last step, two additional quads may be constructed. This gives us the upper bound of $\lfloor \frac{3t}{2} \rfloor + 2$ for the number of strictly convex quadrilaterals in the quadrangulation. At the end of phase l of the algorithm, all nodes of the set V_l have been eliminated, and hence the depth of the tree decreases by at least one. Furthermore, phase l of the algorithm examines only the nodes at levels $l, l-1,$ and $l-2$. In other words, each node in T gets examined only a constant number of times. Therefore, the algorithm runs in $O(t)$ time. The space requirements are clearly $O(t)$ as well. □

3.2 Small Polygonal Regions

In this section, we list several lemmas pertaining to strictly convex quadrangulations of small and simple polygonal regions, i.e., regions consisting of 4, 5, 6, or 7 boundary edges and no holes. As seen in the previous section, these facts are necessary to prove the correctness of our algorithm. We merely state the lemmas here. Proofs are omitted due to lack of space (see details in [6] and [15]).

Lemma 3.2.1. *[6] A hexagon can be decomposed into at most four convex quadrilaterals by using at most three Steiner points in its interior.*

Lemma 3.2.2. *[6] A quadrilateral with a point in its interior can be decomposed into at most five convex quadrilaterals by using at most three Steiner points in its interior.*

For polygonal regions bounded by an odd number of edges, one of the boundary edges is designated as an *outgoing edge*. (The outgoing edge is simply the triangulation edge between the root of subtree T_v and its parent in the algorithm described in the previous section.) When quadrangulating this region, all Steiner points except one are placed in the interior of the polygon, and one Steiner point may be placed on the outgoing edge. The resulting quadrangulation consists of strictly convex quadrilaterals, possibly with one leftover triangle adjacent to the outgoing edge. The following lemmas state the relevant facts formally:

Definition 3.2.1. *Let P be a pentagon and let e be an edge of P. Given a triangulation $\mathcal{T}$ of P such that $V_{\mathcal{T}} = V_P$, $\mathcal{T}$ necessarily consists of three triangles, two of which are ears of $\mathcal{T}$. Each of these ears shares two vertices and a distinct diagonal of $\mathcal{T}$ with the third triangle, denoted by center triangle. Furthermore, the edge e is said to be of type 1 with respect to $\mathcal{T}$ if it is the edge of P shared with the center triangle of $\mathcal{T}$. If e is not of type 1 and e is adjacent to the type 1 edge, it is said to be of type 2 with respect to $\mathcal{T}$. If e is neither of type 1 nor of type 2, then e is incident to the common vertex of all triangles of $\mathcal{T}$ and is said to be of type 3.*

Lemma 3.2.3. *[15] Let P be a pentagon and let e be the outgoing edge of P. Then, given any triangulation $\mathcal{T}$ of P such that $V_{\mathcal{T}} = V_P$, we have the following: (1) If e is of type 1 with respect to $\mathcal{T}$, the pentagon P can be decomposed into two convex quadrilaterals and one triangle adjacent to e by adding one Steiner point inside P. (2) If e is of type 2 with respect to $\mathcal{T}$, then P can be decomposed into three convex quadrilaterals and one triangle adjacent to e by adding two Steiner points inside P. (3) If e is of type 3 with respect to $\mathcal{T}$ then, P can be decomposed into four convex quadrilaterals by adding two Steiner points inside P and one more on the edge e.*

Lemma 3.2.4. *[15] Let P be a pentagon with a point in its interior and let e be the outgoing edge of P. Then, the pentagon P can be decomposed into at most six convex quadrilaterals and one triangle adjacent to e by adding at most four Steiner points inside P.*

In our algorithm, a polygonal region S bounded by seven edges is obtained when the subtree T_v is a path of five nodes, with the middle node as the root of T_v. This is the only case that results in a septagon, and the outgoing edge is always the edge of S belonging to the element corresponding to the middle node of T_v.

Lemma 3.2.5. *[15] Let S be a septagon such that S admits a triangulation $\mathcal{T}$, with $V_{\mathcal{T}} = V_S$, whose dual graph is a path. Let the edge of S contained in the middle triangle of $\mathcal{T}$ be the outgoing edge e. Then, S can be decomposed into six convex quadrilaterals and one triangle adjacent to e by adding at four Steiner points inside S.*

3.3 Constrained Quadrilateral Meshes

Our algorithm for generating quadrilateral meshes of polygonal regions with holes can be extended in a straightforward manner to work with arbitrary constrained triangulations. The ability to handle such input is critical for our image registration application described in Section 5, as we need to construct quadrilateral meshes of polygonal approximations of brain structures that are modeled as nested polygonal regions (see Figure 10). Both the interior and the exterior of each polygonal region (except the exterior of the outermost one) is to be meshed and it is important to respect the boundaries of the polygonal regions in the quadrangulation.

Let $\mathcal{T}$ be a given constrained triangulation with t triangles. Let G be the dual graph of $\mathcal{T}$ (G does not include a dual edge when the corresponding triangulation edge is a constraint), and let h be the number of connected components of G. In order to construct a convex quadrangulation that satisfies the given constraints, we build the spanning forest $T = \{T_1, T_2, \ldots, T_h\}$ of G and run our algorithm on each T_i. The root node of each T_i represents a triangle adjacent to

a boundary (constraint) edge e_i of the underlying triangulation $\mathcal{T}_i$. Let t_i denote the number of nodes in T_i. From the algorithm and Theorem 3.1.1, it follows that $\mathcal{T}_i$ can be quadrangulated with at most $\lfloor \frac{3t_i}{2} \rfloor + 2$ quadrilaterals using at most $t_i + 2$ Steiner points. Note that if t_i is odd, one Steiner point is placed on the constraint e_i and the adjacent triangles are modified accordingly, so that the number of nodes for every T_i becomes even. In either case, we show that the total number of strictly convex quads in the constrained quadrangulation is at most $\lfloor \frac{3t}{2} \rfloor + 3h$, obtained by using at most $t + 2h$ Steiner points.

3.4 Implementation and Results

We implemented our algorithm using C++ and the open source *Computational Geometry and Algorithms Library* (CGAL) class library (`http://www.cgal.org`). Figure 9a shows a triangular mesh with 3346 triangles generated by *Triangle*, which is a constrained Delaunay triangular mesh generator with quality constraints, [16]. Figure 9b shows a quadrilateral mesh with 1762 quads obtained from the mesh in Figure 9a using our algorithm. This reduction in mesh size of about 60% has been observed in almost all our test cases. Figure 9b highlights the input triangular mesh grading preservation that is also present in [3, 13]. Our algorithm does not provide any theoretical guarantee on mesh element shape, and it can indeed generate meshes with a few poorly-shaped quadrilaterals. We can further improve mesh quality by using post-processing methods at the expense of runtime and mesh size. Figure 9c illustrates the result of post-processing the mesh in Figure 9b using angle-based smoothing, [17], and topological clean-up, [18].

4. MESHES FROM IMAGES

Polygonal approximations of structures in a two-dimensional image can be obtained by performing two operations on the input image: image segmentation and boundary approximation. *Image segmentation* is the process of subdividing an image into its constituent parts, [14]. This operation makes it possible to extract the collection of all pixels corresponding to a particular structure of an image. The boundary approximation operation takes as input the collection of pixels of a particular segmented structure, searches for the pixels on the structure boundary, extracts the closed polygonal curves (polygons) defined by the exterior vertices and edges of the boundary pixels, and then simplifies these polygonal curves. The simplification can be carried out by using a polygonal curve simplification algorithm, [19].

The result of applying segmentation and boundary approximation operations to an image is a hierarchy of (nested) polygons, or *contours*, such that the contours of any two consecutive hierarchy levels define one or more polygonal regions. Each polygonal region is an approximation of a distinct segmented structure of the image, so any meshing algorithm for polygonal regions can be used to generate a mesh of the entire image from its set of contours. Figure 10 shows the set of contours obtained from a two-dimensional slice of a segmented human brain image volume.

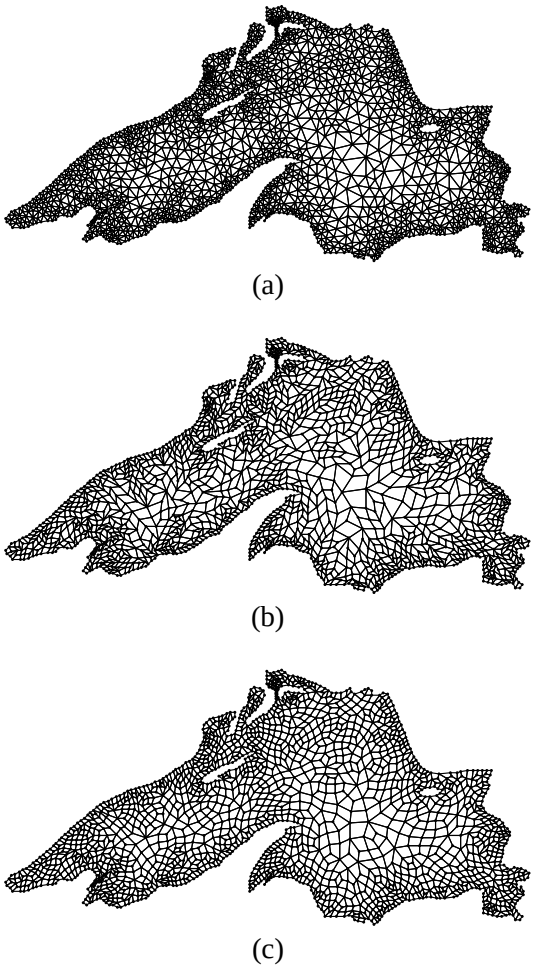

Figure 9: (a) Triangular mesh of "Lake Superior". (b) Quadrilateral mesh of Lake Superior. (c) Mesh in (a) after post-processing.

5. AN APPLICATION

Image registration is the process of finding a spatial alignment between two images so that corresponding features can easily be related, [20]. This spatial alignment is usually achieved by a combination of rigid and non-rigid image transformations. The former is used to globally align two images and it accounts for translational and rotational differences, while the latter is used to maximize the regional similarity between corresponding structures. Image registration has become a very important tool for image analysis, understanding and visualization in medical applications.

Broit, [21], developed a method for non-rigid image registration in which one image, modeled as an elastic continuum, is warped to match the appearance of another. Later,

Gee and Bajcsy, [22], proposed a variational and probabilistic framework to numerically compute a finite element-based solution for the registration problem using Broit's method. An implementation of Gee and Bajcsy's framework is available in the open source Insight Segmentation and Registration Toolkit (ITK), `http://www.itk.org`, sponsored by the National Library of Medicine (NLM). This implementation takes as input a pair (A, B) of images and a mesh for image B, and outputs an image A' resulting from warping image A to match the appearance of image B. Figure 11 illustrates the FE-based image registration method implemented in ITK. Here, we adopt the ITK implementation of Gee and Bajcsy's FE-based image registration framework to evaluate the quality of quadrilateral meshes produced by our algorithm in Section 3, as well as the quality of their triangular counterparts and regular rectangular grids automatically generated by the registration software.

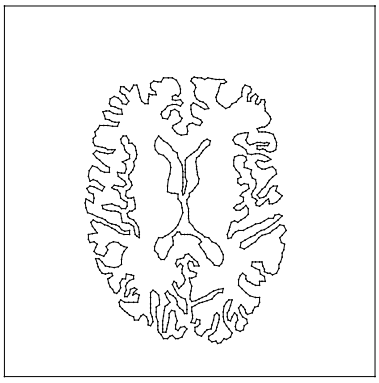

Figure 10: Contours of a human brain image.

We set up an experiment for registering a pair (A, B) of 2D images of the human brain in which A was a coronal MR image with dimensions equal to 256×256 pixels and B was the result of applying a cubic polynomial warp to A. Next, we used the approach in Section 4 to compute a polygonal approximation for the structures of interest in image B, and generated several triangular and quadrilateral meshes for this polygonal approximation. We used Triangle, [16], to produce four triangular meshes whose triangles have minimum angles of 20^o, 25^o, 30^o, and 33^o. Then we produced quadrilateral meshes from these triangular meshes using the algorithm in Section 3. Next, we obtained four more quadrilateral meshes by smoothing and improving the topology of the previous four meshes. We also used an internal procedure within ITK to generate four regular grids of 8×8, 4×4, 2×2, and 1×1-pixel elements. Table 1 shows the number of elements, vertices, and edges of all meshes used in our experiment. Meshes 1–4 are triangular meshes with minimum angle 20^o, 25^o, 30^o, and 33^o, respectively. Meshes 5–8 are quadrilateral meshes generated from meshes 1–4. Meshes 9–12 are the result of post-processing meshes 5–8. Meshes 13–16 are rectangular grids of 8×8, 4×4, 2×2, and 1×1-pixel elements, respectively.

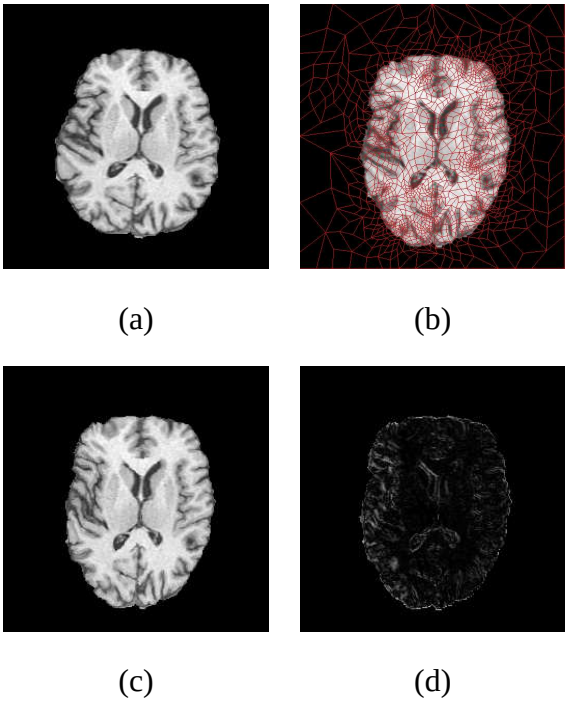

Figure 11: (a) Source image. (b) Target image and its associated mesh. (c) Image resulting from warping image in (a). (d) Subtraction of (c) from image in (b).

Finally, we registered image A to image B 20 times. Each registration used the same pair (A, B) of images and a mesh from Table 1. Meshes 1–4 were used twice each, and meshes 5–16 were used only once. All triangles and quadrilaterals were 3-noded and 4-noded linear elements, respectively. For the registrations using quadrilaterals, four integration points were used by the FE numerical integration procedure. For the registrations using triangles, we used 1 and 3 integration points. All registrations ran for 20 iterations on a PC with an Intel Pentium III processor and 256MB of RAM running Windows 98. We evaluated the results of the registrations by calculating the root-mean squared (RMS) difference between the intensity values of the corresponding pixels over the entire domain of images A' and B. Table 2 summarizes the results obtained from the 20 registrations.

Examination of the results in Tables 1 and 2 demonstrates that the larger the size of a particular type of mesh, the smaller the associated RMS. In addition, quadrilateral meshes 5–8 have less than 60% of the number of elements of their triangular counterparts, meshes 1–4. The use of post-processing techniques to obtain meshes 9–12 from meshes 5–8 produced meshes with approximately 10% more quadrilaterals. Even so, the number of quadrilaterals in meshes 9–12 is still less than 61% of the number of triangles in corresponding meshes 1–4. Despite this reduction in mesh size, the RMS associated with the quadrilateral meshes and their counterparts are comparable. The RMS associated with

the quadrilateral meshes is bounded from above and from below by the RMS associated with their triangular counterparts sampled with 1 and 3 integration points, respectively. Note that post-processing techniques in general improved the quadrilateral meshes. The runtime associated with the quadrilateral meshes, however, is higher than that of the triangular counterparts. This is due to the fact that the procedures to compute the finite element solution using 4-noded quadrilaterals are more expensive than the simple ones associated with 3-noded triangles.

Mesh	#Elements	#Vertices	#Edges
1	2921	1472	4392
2	3549	1790	5338
3	4914	2481	7394
4	8254	4173	12426
5	1645	1657	3301
6	1941	1957	3897
7	2581	2605	5185
8	4318	4364	8681
9	1773	1785	3557
10	2073	2089	4161
11	2747	2771	5517
12	4499	4545	9043
13	1024	1089	2112
14	4096	4225	8320
15	16384	16641	33024
16	65536	66049	131584

Table 1: Size of the meshes 1–16.

Quadrilateral meshes 7 and 11 have about 75% as many quadrilaterals as the regular rectangular grid 14, yet both meshes have a smaller RMS. This is an example of a case in which mesh regularity and well-shaped elements alone were not enough to provide a better result. The registration method is very sensitive along the boundary of distinct structures (see Figure 11d), and appropriately graded meshes can provide a better result. Figure 12 shows triangular mesh 3. Figures 13a and 13b show the quadrilateral mesh 7 produced from mesh 3 by the algorithm in Section 3, and quadrilateral mesh 11 that is the smoothed and topologically improved version of mesh 7, respectively.

6. CONCLUSIONS

We presented an algorithm to convert triangulations of polygonal regions with or without polygonal holes into strictly convex quadrangulations. Our algorithm has a runtime linear in the number of triangles of the input triangulation, offers better bounds than similar algorithms ([3]) that also produce strictly convex quadrilateral meshes of bounded size, and is simpler and faster than algorithms that produce better quality meshes (in terms of element shape, regularity and directionality control) at the expense of runtime, [7, 13, 8]. We also evaluated the quality of the meshes generated by our algorithm, their triangular counterparts and regular rectangular grids with respect to the performance of a FE-based image registration method. Our evaluation demonstrated that our quadrilateral meshes lead to slightly more accurate registrations when compared with those obtained using rectangular grids of similar size, and also lead to solutions comparable with the ones obtained using their denser triangular counterparts.

Mesh	Int. Pts.	Runtime (s)	RMS
1	1	10	17.95
1	3	17	17.25
2	1	12	17.35
2	3	21	16.98
3	1	17	17.20
3	3	30	16.81
4	1	32	16.78
4	3	52	16.62
5	4	21	17.67
6	4	24	17.12
7	4	33	16.93
8	4	59	16.68
9	4	23	17.44
10	4	27	17.47
11	4	35	16.92
12	4	62	16.66
13	4	12	18.56
14	4	46	16.99
15	4	205	16.11
16	4	1001	15.93

Table 2: Summary of the registration results.

Future work will focus on the investigation and formalization of the relationship between the quality of the input triangulation and the quality of the corresponding output quadrangulation obtained by our algorithm with respect to quality measures such as angle bounds, etc. We also intend to extend the experiment in Section 5 to include an algorithm that is known to generate well-shaped quadrilateral meshes, such as the ones in [7, 13, 8], as well as to investigate an extension of our meshing algorithm to produce hexahedral meshes between planar cross-sections of image volumes, a problem directly motivated by applications in medical imaging.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their comments. The first author's research is partially supported by NSF Grant No. CCR-0204293, and the second author is partially supported by CNPq, Brazil.

References

[1] Shewchuk J. "What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures." *Proc.*

11^{th} *IMR*, pp. 115–126. 2002

[2] Bern M., Eppstein D. "Mesh Generation and Optimal Triangulation." F. Hwang, D.Z. Du, editors, *Computing in Euclidean Geometry*. World Scientific, 1992

[3] Everett H., Lenhart W., Overmars M., Shermer T., Urrutia J. "Strictly Convex Quadrilateralizations of Polygons." *Proc.* 4^{th} *Can. Conf. Comp. Geom.*, pp. 77–82. 1992

[4] Bern M., Eppstein D. "Quadrilateral Meshing by Circle Packing." *Proc.* 6^{th} *IMR*, pp. 7–19. 1997

[5] Ramaswami S., Ramos P., Toussaint G. "Converting Triangulations to Quadrangulations." *Computational Geometry: Theory and Applications*, vol. 9, 257–276, 1998

[6] Bremner D., Hurtado F., Ramaswami S., Sacristán V. "Small Convex Quadrangulations of Point Sets." *Proc. Intl. Sym. on Algorithms and Computation (ISAAC)*, LNCS 2223, pp. 623–635. Spriger-Verlag, 2001

[7] Shimada K., Liao J.H., Itoh T. "Quadrilateral Meshing with Directionality Control through the Packing of Square Cells." *Proc.* 7^{th} *IMR*, pp. 61–76. 1998

[8] Viswanath N., Shimada K., Itoh T. "Quadrilateral Meshing with Anisotropy and Directionality Control via Close Packing of Rectangular Cells." *Proc.* 9^{th} *IMR*, pp. 217–225. 2000

[9] Malanthara A., Gerstle W. "Comparative Study of Unstructured Meshes Made of Triangles and Quadrilaterals." *Proc.* 6^{th} *IMR*, pp. 437–447. 1997

[10] Hartmann U., Kruggel F. "A Fast Algorithm for Generating Large Tetrahedral 3D Finite Element Meshes from Magnetic Resonance Tomograms." *Proc. IEEE WBIA*, pp. 184–192. 1998

[11] Lubiw A. "Decomposing polygonal regions into convex quadrilaterals." *Proc. ACM Symp. Comp. Geom.*, pp. 97–106. 1985

[12] Johnston B., Sullivan J., Kwasnik A. "Automatic Conversion of Triangular Finite Meshes to Quadrilateral Meshes." *Intl. J. for Num. Methods in Eng.*, vol. 31, no. 1, 67–84, 1991

[13] Owen S., Staten M., Cannan S., Saigal S. "Q-Morph: An Indirect Approach to Advancing Front Quad Meshing." *Intl. J. for Num. Methods in Eng.*, vol. 9, no. 44, 1317–1340, 1999

[14] Gonzalez R., Woods R. *Digital Image Processing*. Addison-Wesley, 2^{nd} edition, 2002

[15] Ramaswami S., Siqueira M., Sundaram T., Gallier J., Gee J. "Quadrilateral Meshes for FE-Based Image Registration." Tech. Rep. MS-CIS-03-16, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA, 2003

[16] Shewchuk J. "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator." *1st Workshop on Applied Comp. Geom.*, pp. 124–133. Philadelphia, PA, USA, May 1996

[17] Zhou T., Shimada K. "An Angle-Based Approach to Two-Dimensional Mesh Smoothing." *Proc.* 9^{th} *IMR*, pp. 373–384. 2000

[18] Kinney P. "CleanUp: Improving Quadrilateral Finite Element Meshes." *Proc.* 6^{th} *IMR*, pp. 449–461. 1997

[19] Hershberger J., Snoeyink J. "Speeding Up the Douglas-Peucker Line-Simplification Algorithm." *Proc. 5th Intl. Symp. on Spatial Data Handling*, vol. 1, pp. 134–143. 1992

[20] Hajnal J., Hawkes D., Hill D. *Medical Image Registration*. CRC Press, 2001

[21] Broit C. *Optimal Registration of Deformed Images*. Ph.D. thesis, Department of Information and Computer Science, University of Pennsylvania, Philadelphia, PA, USA, 1981

[22] Gee J., Bajcsy R. "Elastic Matching: Continuum Mechanical and Probabilistic Analysis." A. Toga, editor, *Brain Warping*. Academic Press, 1999

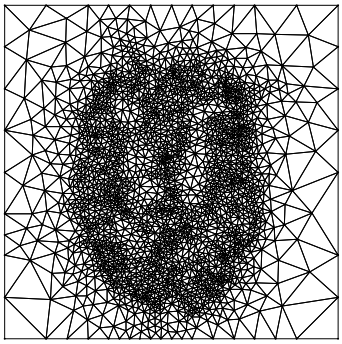

Figure 12: Triangular mesh with minimum angle 30^o.

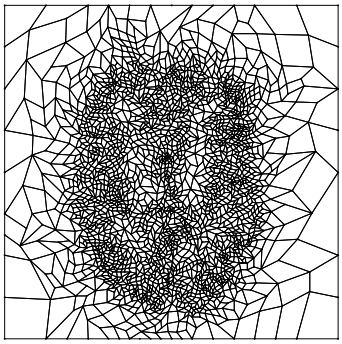

Figure 13: Mesh 7.

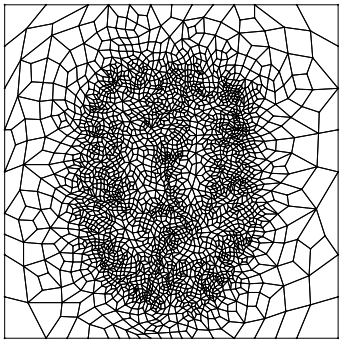

Figure 14: Mesh 11.