Image-based Adaptive Meshing

Andrew C.E. Reid and Stephen A. Langer*

Information Technology Laboratory

National Institute of Standards and Technology

Gaithersburg, MD, 20899

Rhonald C. Lua and Valerie R. Coffman

Materials Science and Engineering Laboratory

National Institute of Standards and Technology

Gaithersburg, MD, 20899

R. Edwin García

School of Materials Engineering

Purdue University

West Lafayette, IN 47907-2044

(Dated: August 28, 2007)

INTRODUCTION

Most finite element packages require as input numerical representations of straight lines, flat planes, and simple curves that make up the boundaries of the structure to be simulated. For microstructures, such as the one shown in figure 1, converting the boundaries to a simple numerical representation by hand is a tedious task. In this paper, we will describe how the software package OOF (named for "Object Oriented Finite-Elements") [1, 2, 3] circumvents this difficulty by creating meshes directly from images – the mesh itself is the geometric description of the microstructure. Through utilizing a unique, image-based, adaptive meshing technique, OOF is capable of parsing experimental data relating polyphase, polydomain materials with complex geometries into a representation that is appropriate for use in a finite element simulation. [need to reword to emphasize how other packages make a distinction between the mathematical representation of the geometry and the mesh]

OOF is a unique tool for performing virtual experiments on microstructure geometries extracted *directly* from 2D experimental images. OOF has been used successfully in a wide

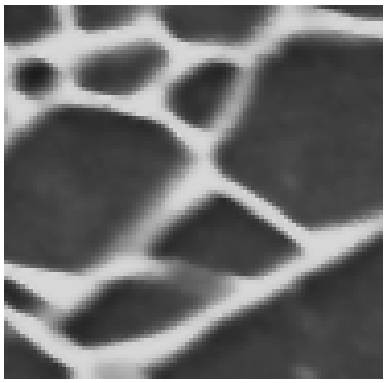


FIG. 1: **Microstructure Image Example**: A micron scale SEM image of plasma-etched Si_3N_4, provided by Chun-Hway Hsieh at Oakridge National Laboratory) [4].

variety of applications including studying the relationship between thermal properties and the structure of plasma-sprayed zirconia coatings [5], the effects of microstructure on the macroscopic mechanical properties of glass-matrix composites [6], the role of texture in the macroscopic response of polycrystalline piezoelectric materials [7], and the modeling and design of electrode microstructures in rechargeable lithium-ion batteries [8].

One common method of forming a finite element mesh from an image is to define a square element for each pixel [9, 10]. This has two major pitfalls. First, it usually creates too many elements – homogenous regions can adequately be described by larger elements. Second, it introduces jagged edges where the boundary in the real material is smooth, which can lead to artificial results such as stress concentrations at corners. Since the image is an imperfect representation of the actual material, and the mesh is an approximation of the image, it does not make sense to force the boundaries of the mesh to line up exactly with the image boundaries [1].

Most current meshing strategies involve triangulating a surface and tetrahedralizing the interior. This strategy suffers from the pitfall of creating large, un-meshable voids [more specific, name theorem, find references, relevant in 2D?]. OOF's meshing system has two advantages over other strategies. Because OOF's meshing method begins with a space-filling mesh that is only modified by methods that preserve the space-filling quality of the elements, there will be no unmeshable voids or (very rarely) illegal elements.

-briefly explain how to process image to prepare for meshing - threshold and assign materials? SAL: Mention pixel categories, goal is for element edges to follow boundaries of pixel categories.

Once the material regions and their properties have been assigned, OOF can create the mesh directly from the image.

The technique OOF uses is to create an initial mesh that is a regular, space-filling grid of rectangles or triangles, with user-specified dimensions. In order to optimize this mesh and align it with the boundaries of the microstructure, OOF provides a number of methods for mesh adaptation. Several methods move the nodes in order to improve homogeneity or shape quality. Other methods change the topology of the mesh by refining inhomogenous elements, merging homogenous elements, or correcting for badly-shaped elements.

Quantifying the Quality of the Mesh

[RCL: This is just my suggestion (it is wordy; I'm just thowing out ideas. Will pare it down): For an automated meshing procedure to work, a figure of merit, or a quantitative measure of the quality of the elements in the mesh, must be introduced that has as few components as possible. This quantitative measure(s) must reflect the degree to which the mesh represents the microstructure. It also must reflect the quality of the elements as far as being appropriate to a finite element calculation is concerned.

To this end, we define two element energies: the shape energy and the homogeneity energy. We use the word "energy" because the mesh is considered a good representation of the microstructure, or good for a finite element calculation, if the value of the energy is low. The mesh adaptation routines are oriented towards lowering the energy of a mesh. The analogy is all the more appropriate because many of the mesh adaptation routines work by moving the nodes in a mesh (see Figure 3), as if the nodes are particles under the influence of some fictitious potential or force field.]

One of the methods for mesh adaptation is the anneal method (section), which moves nodes randomly and accepts those that lower a measure of mesh quality [2, 3] analogous to an effective energy. The effective energy used by the anneal method and other mesh

adaptation algorithms in OOF conists of two parts: the shape energy, which quantifies the quality of the shapes of elements, and the homogeneity, which measures how well the mesh matches the pixel regions.

We want to minimize the shape energy because we want to avoid elements with high aspect ratios which lead to poor finite element convergence. Equilateral triangles and squares, which are ideal, have a shape energy of zero while thin and flat elements have higher energies [RCL: According to this article [11, 12] by the same guy who wrote "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain", thin elements may not be so bad. At least, it is bad in a different way from flat elements]. Degenerate elements with colinear corners, have the maximum shape energy, which is normalized to 1.0. For triangular elements, the shape energy is given by

$$E_{\text{shape}} = 1 - 4\sqrt{3}\frac{A}{L_0^2 + L_1^2 + L_2^2} \tag{1}$$

where A is the area of the triangle and L_i are the lengths of the sides. This function gives a value of 0 for equilateral triangles and higher values for narrow triangles. For quad elements, OOF calculates a quality measure for each corner [13]

$$q = 2\frac{A_{\parallel}}{L_0^2 + L_1^2} \tag{2}$$

where $A_{\parallel}$ is the area of the parallelogram formed by the two edges adjacent to the corner in question. L_0 and L_1 are the lengths of the adjacent sides. OOF calculates the shape energy for a quad by constructing a weighted average from the q's at the corner with the minimum q (q_{min}) and at the corner opposite it (q_{opp}).

$$E_{\text{shape}} = 1 - ((1 - w_{\text{opp}})q_{\text{min}} + w_{\text{opp}}q_{\text{opp}}). \tag{3}$$

The weighted average is taken to make sure that any node movement changes the energy, even if it doesn't shift the minimum. Without the weighted average, some node moves would not register a response, and would behave badly in some of the routines. [Word this better]

The homogeneity (see Figure 2) is a measure of how close an element is to enclosing a region that contains a single material:

$$H = \frac{\max\{a_i\}}{A} \tag{4}$$

where a_i are the areas within an element that belongs to each material phase. If an element is filled with only one material, $H = 1$. The corresponding homogeneity energy is defined

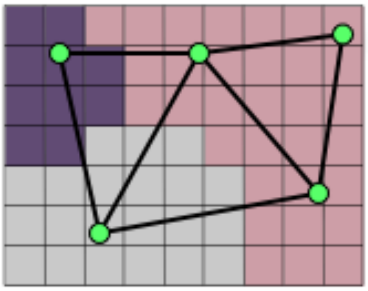


FIG. 2: If the three different pixel colors represent different materials, then the triangle in the upper right is completely homogeneous ($H = 1.0$), the central triangle is inhomogeneous ($H \approx 0.5$) and the leftmost triangle is even more inhomogeneous ($H \approx 0.3$).

by

$$E_{\text{hom}} = 1 - H \tag{5}$$

These two measures can be combined into an effective element energy

$$E = \alpha E_{\text{hom}} + (1 - \alpha) E_{\text{shape}} \tag{6}$$

where α is a tuneable parameter. When performing several of the mesh adaptation routines, a user may adjust α according to whether his or her priority is improving shape quality or homogeneity.

Adaptive Methods

The mesh adaptation routines used in OOF can be classified into two groups: (1) Routines that move nodes (Movers) and that maintain the connectivity or topology of the starting mesh (e.g. Anneal, Snap Nodes), and, (2) Routines that split elements and edges (Splitters) and that change the connectivity of the starting mesh (e.g. Refine, Snap Refine). Routines in both groups are used to improve overall element energy. The routines and their classifications are illustrated in Table I.

[VRC: Do we want to call them "splitters"? Merge triangles, for instance, doesn't split. Rationalize and swap edges also doen't necessarily split. If we wanted to be cute, we could call them "movers" and "shakers". But maybe "rearrangers" is more descriptive? Or maybe we shouldn't force a one word label on them... RCL: I also realized that Merge Triangles does not "split" an existing element. However, it does change the way elements are split

TABLE I: OOF Mesh Adaptation Routines

Group	Routine	Monte Carlo	Snapper	Fixer
Movers	Anneal	×		
(Connectivity of	Smooth	×		
nodes preserved)	Snap Anneal	×	×	
	Snap Nodes		×	
	Fix Illegal Elements			×
	Relax			
Splitters	Refine			
(Topology of	Snap Refine		×	
mesh changed)	Rationalize			×
	Split Quads			
	Merge Triangles			
	Swap Edges			

(it "unsplits" a quad made of two triangles into a solid quad). "Splitters" is just a single-word term I came up with to denote the group of routines that change the topology of the mesh. In general, I think the labels ("Movers", "Splitters", "Snappers", etc.) are helpful in introducing the various skeleton modification routines and it would not be harmful (as in politics) to put labels on a related group of routines. Yes, I think it is somewhat cute, but not an overload of cute, to introduce these labels. I put them in seriously to organize the routines into one neat table. We could of course vote at the end of the day on whether to not keep these labels, or expand the labels (e.g. "Movers" to "Preserves connectivity of the mesh") and I'd have no problem with it.]

Movers can improve the homogeneity or shape energy of the elements in the mesh without increasing the number of elements. Unless the initial grid is sufficiently fine, Movers usually cannot be used to resolve the small features in a microstructure. Splitters are used to achieve this task by subdiving existing elements and segments and spawning new elements and nodes. Once the desired element resolution is reached, Movers can then be used to further improve the mesh.

Most of these routines can be flexibly applied to an explicitly selected set of elements, segments or nodes. Most of the routines can also be targeted to elements or segments that satisfy certain criteria, such as having a homogeneity below a certain threshold.

Because of the similarity of Movers to steps in a molecular dynamics simulation, some of these routines (e.g. Anneal, Smooth, Snap Anneal) can be coupled to a type of Metropolis Monte Carlo method [14]. The factor ΔE in the exponent that enters the Boltzmann factor

$$P = \exp\left(-\Delta E/T\right) \tag{7}$$

is the change in the energy (6) of the affected elements. A change in the shape of some elements can be accepted even if it leads to an increase in the overall element energy, with a probability given by P. This analogy with molecular dynamics simulations can be pursued further by explicitly applying fictitious forces on the nodes that may drive the elements to improve their homogeneity or shape (e.g. the Relax routine).

One can also define two subcategories of mesh adaptation routines. The first (Fixer) can be characterized by its specialized use in correcting or improving an element or group of elements that were created as a result of repeated application of other mesh adaptation routines such as Refine or Snap Refine. [SAL: Is Corrector (RCL: or Fixer) really a well defined term? Can't Refine be a Corrector too? Split Quads can be used on its own, not just to correct previous modifications.] These routines (e.g. Rationalize, Fix Illegal Elements) can be Movers or Splitters. Fixers are more oriented toward improving the shape quality of the elements in the mesh. The second subcategory of routines (Snappers) places nodes precisely at the interface or boundary of two different pixel groups (Snap Nodes, Snap Refine, Snap Anneal). The nodes that get placed at these points along an interface, called transition points, may be existing nodes or new nodes created along an existing edge or within an existing element. Snappers are more oriented toward improving the homogeneity of the mesh. [VRC: I'm not sure the labels "Fixer", "Snapper", and "Monte Carlo" are helpful. I think the same information (the role/usefulness of each method) can be conveyed in the intros to each of the subsections which should explain what the role of each method is.]

In the following sections, we describe each mesh adaptation routine used in OOF. We also decribe a combination of these routines that seems effective in creating a mesh for a categorized microstructure. This sequence of routines is applied in OOF after providing just

a little input and once started requires no feedback from the user.

[SAL: Each of the following subsubsections needs a summary paragraph explaining the circumstances in which the skeleton modifier would be used, and the effect it has.]

Anneal

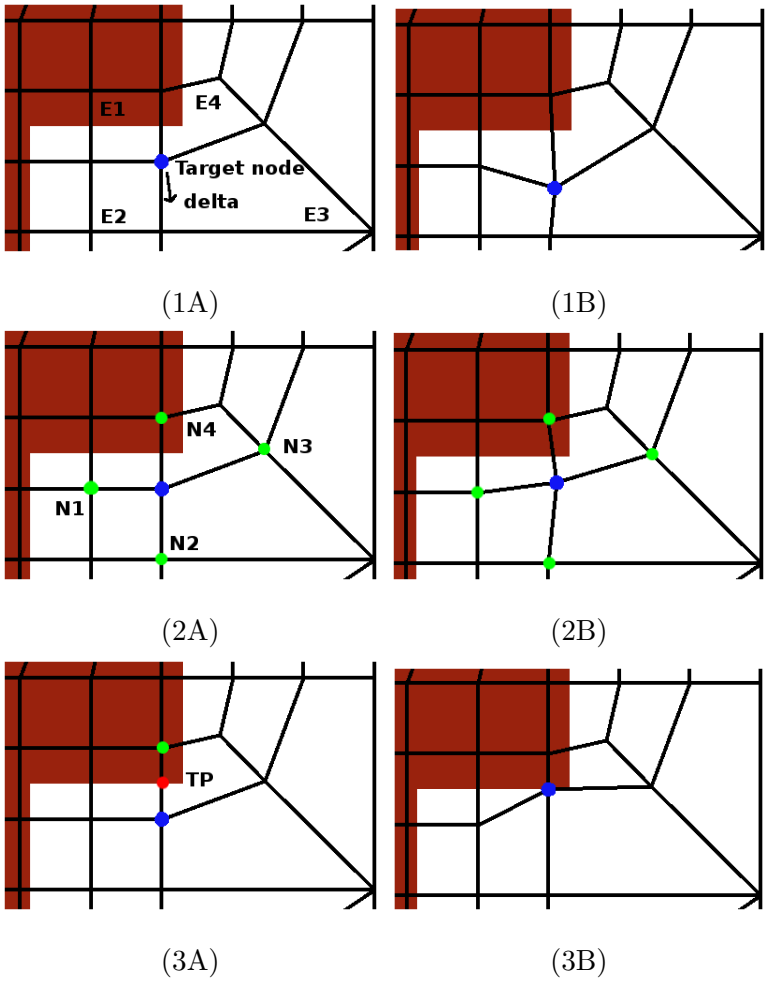


FIG. 3: Illustration of some node moves used in OOF. 1A and 1B show a trial move of the Anneal routine. A target node is displaced by a random vector whose components are taken from a Gaussian distribution. 2A and 2B show the trial move for the Smooth routine. The target node is placed at the center of the four neighboring nodes $N1$,$N2$,$N3$ and $N4$. 3A and 3B show the trial move for the Snap Anneal routine. The target node is shifted to the transition point (TP) along one of the edges incident at the target node.

Anneal moves nodes to random positions (see Figure 3-1A and -1B), accepting or rejecting

moves according to a given criterion. It is similar to a simulated annealing simulation in statistical mechanics, from which it gets its name. Instead of minimizing the free energy of a system of particles, it minimizes the effective energy of a mesh.

The general procedure for a single iteration of Anneal is as follows:

1. Collect target nodes and reorder randomly to remove any potential artifacts from the original ordering of nodes. This reordering is repeated at every iteration.

2. Give each node a single chance to move to a randomly assigned new position. OOF computes the new position from

$$x_{\text{new}} = x_{\text{old}} + \delta x$$
$$y_{\text{new}} = y_{\text{old}} + \delta y$$

   δx and δy are random numbers chosen from a Gaussian distribution of width delta and mean 0.0. Figure 3-1A shows a target node that is about to move to a new position. Before making the move, OOF computes the total effective energy of all the neighboring elements of the node ($E1$,$E2$,$E3$,$E4$).

3. After moving each node, the given acceptance criterion is used to decide whether the move is acceptable.

4. If the move is unacceptable according to the acceptance criterion, OOF may still accept the move if the annealing is being done at a non-zero temperature T. The parameter T sets the effective temperature of the annealing process. These moves are accepted with a probability P given in Equation (7), where ΔE is the difference in the effective energies of the new and old element configurations.

Successful annealing usually requires a number of iterations. On each iteration, OOF makes one attempt to move each node. OOF can be set to perform a fixed number of iterations of Anneal or to stop after some condition is satisfied. As with conventional simulated annealing or monte carlo simulations, if the change in energy or acceptance rate gets too small, the annealing process is improving the mesh effectively. [RCL: I think this statement, as well as the one it replaced, are both true. From the manual: "If the change in energy or the acceptance rate gets too small, the annealing process is not being effective at improving

the Skeleton." If the parameters were set such that the acceptance rate is too small, then the annealing may not be effective. It is possible though that the mesh is good to start with. Now if in the first few iterations of anneal the acceptance rate is significant, and later it got small, that could be a signal that the mesh was significantly improved by annealing.]

Smooth

Smooth moves nodes to the average position of their neighbors (see Figure 3-2A and -2B) in order to smooth out gradients in node density in a mesh. This tends to improve the shape of the elements. Typically, it takes about 3 iterations of Smooth to get a desirable result. The general procedure for a single iteration is very similar to that for Anneal, except that in step 2, each target node is given a single chance to move to the average position of its neighbors. For this purpose, neighbors are defined as nodes that share a segment with the target node ($N1$,$N2$,$N3$,$N4$).

The technique of moving nodes to the average position of its neighbors in the mesh is also used to fix illegal elements (section).

Snap Anneal

Snap Anneal is very similar to Anneal. In Snap Anneal, the trial move consists of placing the target node at a nearby pixel boundary, called a transition point, located along a segment that contains the target node as an endpoint (see Figure 3-3A and -3B).

Snap Nodes

Snap Nodes moves nodes to improve the elements' homogeneity energy. If an element edge crosses over regions of the microstructure belonging to different pixel categories, then Snap Nodes tries to move one of the element coreners to the crossing point. These points are precisely the transition points used in Snap Anneal and illustrated in Figure 3-3B. Snap Nodes differs from Snap Anneal by having correlated snaps of two nodes. That is, more than one node of an element may be moved at the same time in order to increase the homogeneity.

The general procedure for snapping nodes is as follows:

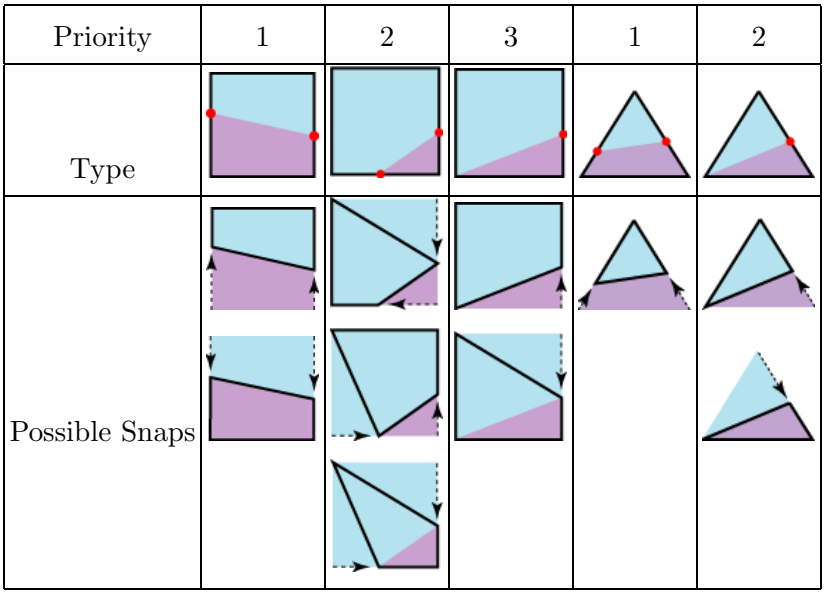


FIG. 4: Snapping Nodes.

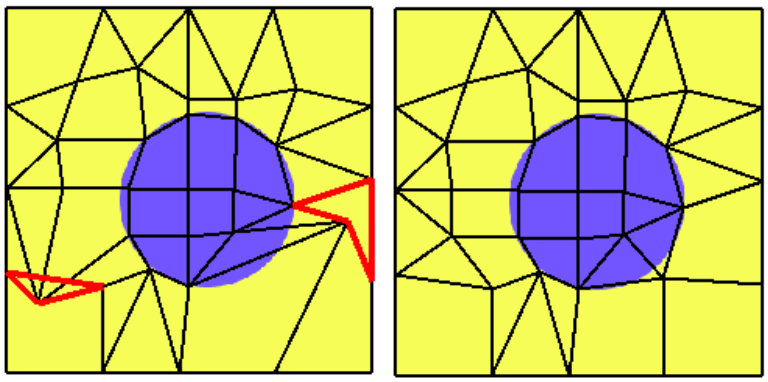
1. Scan the current mesh to find candidate nodes for snapping.

2. Loop over the elements containing snappable nodes, and identify the transition points along each segment of each element.

3. Find all possible snaps involving one or two nodes of a single element, assigning a priority to each according to the arrangement of its transition points. The priorities are listed in Figure 4.

4. Starting with the highest priority snaps (and choosing randomly from snaps with equal priorities), attempt to actually move the nodes. The move is accepted or rejected according to a given criterion. When more than one snap is possible for one element, all are attempted and the best is chosen. After successfully snapping the nodes of an element, the nodes of any neighboring snappable elements are attempted, regardless of their priority.

Snapping nodes tends to produce badly-shaped elements when the weighting in the element energy (α) favors homogeneity. However, Snap Nodes is really a homogeneity-oriented operation, and works best when given a large α. It is usually best to allow it to create badly-shaped elements, and to clean up afterwards by applying the Rationalize routine.



(A) (B)

FIG. 5: Before (A) and after (B) application of the Fix Illegal Elements routine on a mesh. In (A), the elements that are fixed include a flipped triangle (left), and a non-convex quadrilateral. Both were intentionally created using OOF's interactive graphical toolboxes to select and move nodes.

Illegal elements are elements having corners that subtend an angle greater than 180 degrees or less than 0 degrees. For quadrilateral elements, these include elements that have non-convex shapes. Fix Illegal Elements modifies and corrects these elements by moving one or more nodes in the element to the average position of its neighboring nodes, just like in Smooth. The move is accepted or rejected solely by whether or not it reduces the number of illegal elements in the vicinity of the node.

In many cases, it is sufficient to fix an element by applying this smoothing procedure to the nodes at the offending corners. However, this procedure may not improve the element when the smoothing is applied to a node located at a corner with an angle very close to zero. In this case, the nodes at the other corners should be tested and moved as well.

An example of the Fix Illegal Element procedure is illustrated in Figure 5. On one pass through the mesh, Fix Illegal Elements attempts to move each node in an illegal element once, choosing nodes in a random order. It may not be successful on the first pass, so it repeats the process until there are no more illegal elements, or when the routine has tried many number of times.

After an element has been fixed, it is possible that the mesh is no longer a good representation of the microstructure. This is because the Fix Illegal Elements routine does not pay attention to element homogeneity or shape.

Relax

Relax is like a deterministic version of Anneal. It works by giving each element an isotropic elastic modulus and an elastic driving force with two components. The first component is isotropic and proportional to the element's homogeneity energy. This causes inhomogeneous elements to shrink. The second part is anisotropic and proportional to the traceless part of the element's moment of inertia tensor (the masses are concentrated at the nodes). This drives the elements' shapes toward squares for quadrilaterals and towards equilateral triangles for triangles. The relative weight of the two terms can be adjusted [α?]. Using these two driving forces, the positions of the nodes are computed for a fixed number of time steps.

Refine

Refine is a routine that chops its target elements and their neighbors into smaller pieces (see Figure 6-B). This adds more degrees of freedom to the mesh, which allow it to adapt better to the microstructure. Refinement by itself is rarely sufficient to create an acceptable mesh – it must be combined with other routines that move nodes or place nodes at pixel boundaries, such as Anneal or Snap Refine.

The steps involved in refining a mesh are:

1. Find the element edges that need to be refined. These edges may be chosen from edges of heterogeneous elements, or from long edges of skinny elements.

2. Sometimes, the way in which the elements will be subdivided (for instance, whether the edge is to be bisected or trisected, or if a quadrilateral should only be refined into quadrilaterals) requires extra segments to be marked. In these cases a second pass is made through the mesh, marking additional segments.

3. After the segments are marked, each element is replaced by a suitable collection of smaller elements, such that each marked segment of the old mesh turns into a set number of segments of the new mesh. The rules sometimes allow more than one way of subdividing an element. In those cases, the total effective energy of the new elements is computed, and the configuration with the lowest energy is used.

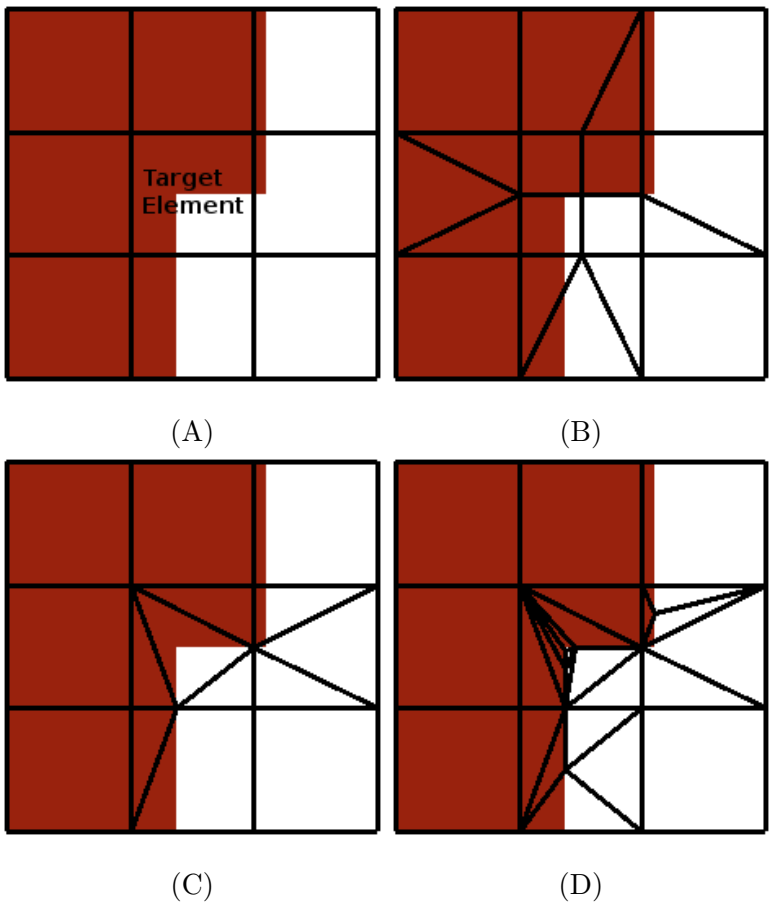


FIG. 6: Illustration of the refinement of a target element using Refine and Snap Refine. (A) The target for refinement is the region defined by the central element. (B) One application of the Refine routine splits elements and bisects edges. (C) One application of Snap Refine subdivides elements and splits edges at transition points. (D) Four successive applications of Snap Refine improves the homogeneity but produces thin elements.

[SAL: mention that the advantage of having two passes, one to mark edges and one to replace elements, is that the replacement step is completely local.]

Snap Refine

Snap Refine attempts to combine the most desirable features of Refine and Snap Nodes into a single method. The routine operates like Refine, except that the element segments are split by points located at the interface of two different pixel categories (see Figure 6-C). These points become the corners of new elements that subdivide the original elements.

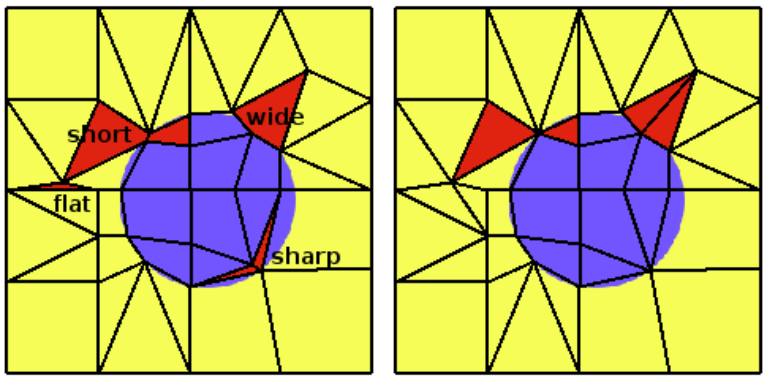
Snap Refine can yield a mesh with a comparable homogeneity to a mesh produced by

Refine, but often with significantly fewer elements. Snap Refine can also be more flexible than Snap Nodes in fitting a mesh to a microstructure, because Snap Refine introduces new nodes and edges that can be made to follow the boundaries of pixel categories and resolve smaller features in the microstructure.

It may seem obvious that one should always use Snap Refine instead of Refine. Unfortunately, Snap Refine can lead to the creation of very thin elements (see Figure 6-D). The creation of such badly-shaped elements may be prevented in some cases by telling Snap Refine to not place a new node within a certain distance of one of the corners of an element (although this may not be a scale-invariant solution). In general, iterating Snap Refine produces very bad results, with a lot of very thin elements. A good strategy is often to use the regular Refine routine first, and to apply Snap Refine just once at the end to snap nodes and edges to the interface.

Currently, Snap Refine finds at most two transition points along an edge, by scanning the edge starting from each endpoint. The subdivision of an element may also introduce new nodes with the element. These nodes may also be located at the interface of pixel categories.

Rationalize



(A) (B)

FIG. 7: Before (A) and after (B) application of the Rationalize routine on the highlighted elements. In (A), the elements that are fixed by the Rationalize routine include (clockwise from left) a flat triangle, two quadrilaterals that share a short side, a wide quadrilateral, and two sharp or thin triangles.

Rationalize fixes badly-shaped elements in a mesh by either removing them or modifying

them and their immediate neighbors. Rationalize has three components (called Rationalizers):

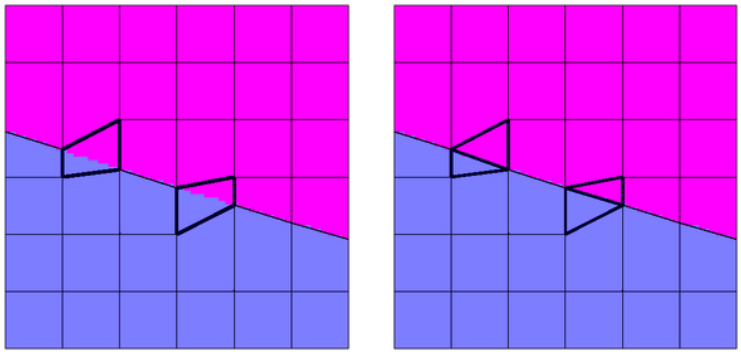
1. Remove short sides - Eliminate the shortest side of a quadrilateral.

2. Split wide quads - Split quadrilaterals with large interior angles.

3. Remove bad triangles - Remove triangles with extreme interior angles.

An example is shown in Figure 7.

Split Quads



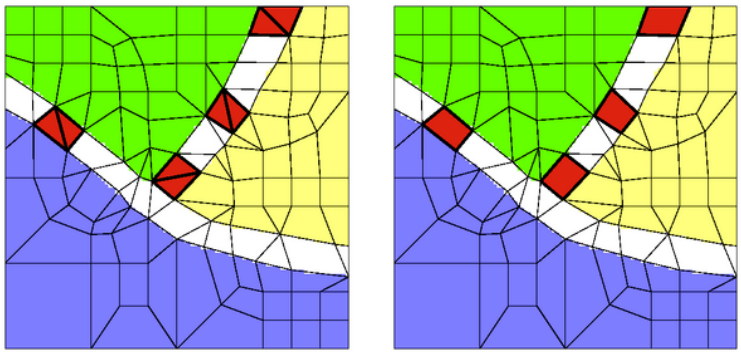
(A) (B)

FIG. 8: Before (A) and after (B) application of the Split Quads routine.

[Put Split Quads, Swap Edges, Merge Triangles into one subsubsection?] Split Quads improves the homogeneity energy of a quadrilateral by splitting it along a diagonal into two triangles. Applying a Snapper such as the Snap Nodes routine tends to leave some quadrilaterals tailor-made for splitting along a diagonal. An example is shown in Figure 8.

Merge Triangles

Merge Triangles merges two triangular elements into one quadrilateral element. (see Figure 9) Given the nature of the process (merging elements), it is easy to see that it focuses mostly on reducing elements' shape energy rather than increasing elements' homogeneity. It also reduces the overall degrees of freedom in the resulting mesh, thereby decreasing the

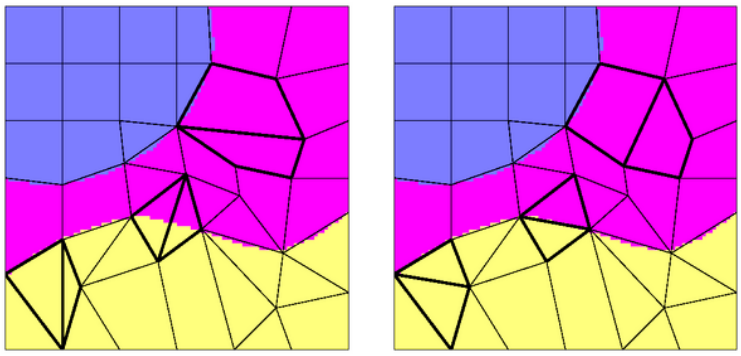


(A) (B)

FIG. 9: Before (A) and after (B) application of the Merge Triangles routine.

solution time (and accuracy), if the mesh is going to be built from high-order elements. For meshes made of linear elements (4-noded quads or 3-noded triangles) merging triangles does not reduce the number of degrees of freedom.

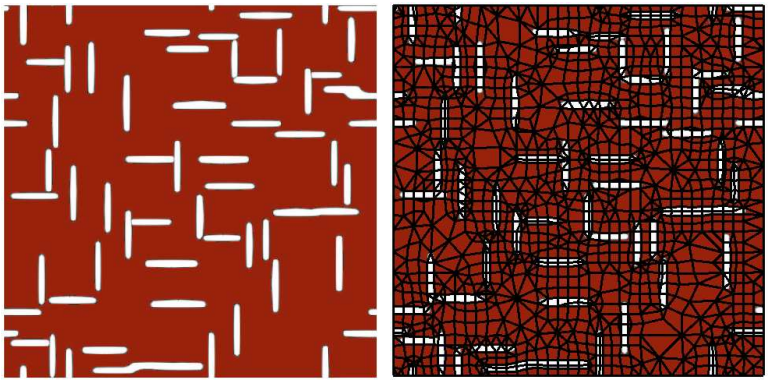
Swap Edges



(A) (B)

FIG. 10: Before (A) and after (B) application of the Swap Edges routine.

Swap Edges rearranges the interior segments of neighboring elements to improve their shape energy or homogeneity. An example is given in Figure 10.



(A) (B)

FIG. 11: (A) Microstructure created from a 484 pixel × 484 pixel image. (B) Mesh with 1851 elements created with OOF's automatic mesh generation feature. The parameters used were $maxscale = 60$ pixels, $minscale = 20$ pixels and $threshold = 0.9$.

Automatic Mesh Generation

OOF has a close-to-automatic mesh generation feature built from a sequence of the mesh adaptation routines described in the previous sections. It was originally conceived with the goal of using OOF as part of the automated MatCASE [15] framework for computational materials design.

OOF's automatic mesh generation feature requires three parameters: (1) A rough size of the elements in the initial mesh (*maxscale*). This should correspond to the linear size of the largest features in the microstructure. (2) A rough size for the smallest elements in the desired mesh (*minscale*). This is the resolution to which the elements have to be refined by the mesh adaptation routines, and should correspond to the linear size of the smallest features in the microstructure. (3) A homogeneity *threshold* that determines which elements will be refined. All elements whose homogeneity is smaller than the threshold will be subdivided until the element size falls below *minscale*. These parameters may be estimated for a class of microstructures based on the intuition of a materials engineer or from the nature of the physical problem. In some cases, the parameters may be obvious. For example, *minscale* may be set to some average pore size in a microstructure.

As an example, we used OOF to generate a mesh for a microstructure consisting of the 484 pixel × 484 pixel image given in Figure 11-A. The light and dark regions define different phases or materials. For this microstructure, the parameters used were $maxscale = 60$

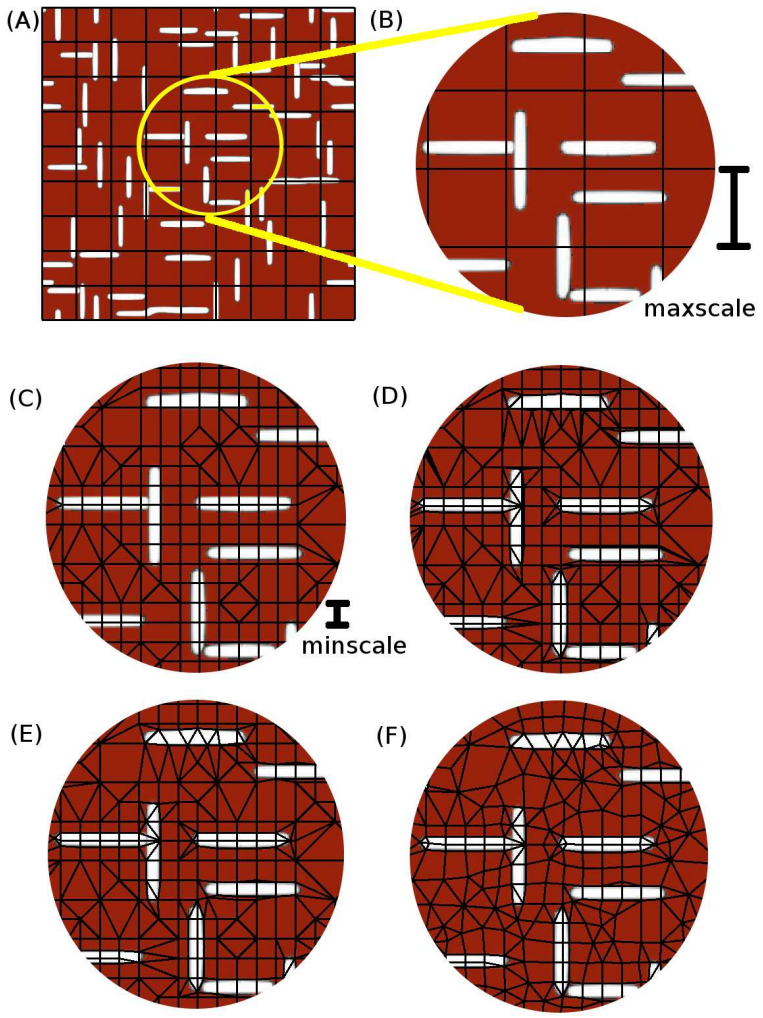


FIG. 12: (A) Microstructure with initial mesh. (B) Close-up of a portion of the microstructure and mesh. *maxscale* is the rough size of each element. The area-weighted average homogeneity of the elements is 0.846. (C) The mesh after a few applications of the Refine routine. Elements with homogeneity less than *threshold* were refined until the resulting elements reached a resolution given approximately by *minscale*. At this stage the homogeneity is 0.890. (D) The mesh after one application of the Snap Refine routine. Note that nodes and edges have been placed at pixel boundaries. The homogeneity is 0.980. (E) The mesh after two applications of the Rationalize routine that fixes badly-shaped elements. The homogeneity is 0.981. (F) The mesh after 5 iterations of the Smooth routine.

pixels, *minscale* = 20 pixels and *threshold* = 0.9. The resulting mesh overlaid on the microstructure is given in Figure 11-B.

The main sequence in the automatic mesh generation is depicted in Figure 12. OOF first creates a mesh with quadrilateral elements. The size of the elements is set so that the initial mesh resolves features of size *maxscale* in the microstructure. The mesh is then refined iteratively with the Refine routine (edges are bisected and $\alpha = 0.8$), until the linear dimensions of the smallest elements are smaller than *minscale*. To get rid of rough corners and to snap nodes and edges onto boundaries, Snap Refine is applied once. All refinement operations are applied only to elements whose homogeneity is less than that specified by *threshold*. After refining, OOF cleans up the mesh by applying the Rationalize routine twice. It uses specific Rationalizers to remove short sides of quadrilaterals (those with ratio of the longest side to the shortest side larger than 5.0), split wide quadrilaterals (those with angle greater than 150 degrees), and remove bad triangles (those with acute angle less than 15 degrees and obtuse angle greater than 150 degrees). Finally, OOF applies the Smooth routine after pinning the internal boundary nodes, which OOF unpins when it is done. Smoothing is done for 5 iterations (with $\alpha = 0.3$).

COMPARISON TO OTHER METHODS

will need to fill in with data.

3D?

- this approach was designed with extending to 3d in mind
- currently working on 3d
- challenges of adapting to 3d

[Mention microstructure representation in 3D. Simplest is 3D array of pixels or voxels.] The concept of homogeneity is readily extended to 3D. For the shape quality of an element, one can define a quality measure analogous to the expression in Equation (1).

$$q = CV^2/S^3 \tag{8}$$

where V is the volume of the element, S is the surface area of the element, and C is a constant that normalizes q ($C = 216\sqrt{3}$ for a tetrahedron. Got this from the internet).

[It seems defining and evaluating a good q for elements in 2D and 3D is an active field of research. I like the idea of using the ratio of the radius of the inscribing sphere (or circle) and the radius of the circumscribing sphere (or circle). It might be computationally more demanding.]

Many of the 2D mesh adaptation routines have a straightforward extension to 3D. For some of the Movers (or node moving routines), the extension to 3D is no different from doing monte carlo or molecular dynamics simulation for particles living in 3D instead of 2D. [If the paper is going to be submitted to a Computer Science journal, I wonder if the statmech jargon should be reduced.] As for element refinement in 3D, we clearly have to work with many more refinement possibilities, even if we only consider a single type of element (e.g. tetrahedrons). We have worked out a set of rules for refining a tetrahedron into smaller tetrahedrons given a combination of its edges that have to be bisected [16]. 3D meshing is bound to be more computationally demanding. Whether new mesh adaptation routines for 3D that does not have a 2D counterpart have to be developed, awaits the development and testing of OOF 3D.

CONCLUSION

Fill in later.

---

* Electronic address: stephen.langer@nist.gov

[1] Langer, S. A., Fuller, Jr, E. R., and Carter, W. C. *Computers in Science and Engineering* **3**(3), 15–23 (2001).

[2] Langer, S. A., Reid, A. C. E., García, R. E., Haan, S.-I., Lua, R. C., Carter, W. C., Fuller, Jr, E. R., and Roosen, A. Webpage. http://www.ctcms.nist.gov/oof/index.html.

[3] Langer, S. A., Reid, A., Haan, S.-I., and García, R. E. website. http://www.ctcms.nist.gov/ langer/oof2man/index.html.

[4] Paul, F. B., Sun, E. Y., Plucknett, K. P., Alexander, K. B., Hsueh, C.-H., Lin, H.-T., Waters, S. B., Westmoreland, C. G., Kang, E.-S., Hirao, K., and Brito, M. E. *Journal of the American Ceramic Society* **81**(11), 2821–2830 (1998).

[5] Wang, Z., Kulkarni, A., Deshpande, S., Nakamura, T., and Herman, H. *Acta Materiala* **51**, 5319–5334 (2003).

[6] Cannillo, V., Manfredini, T., Montorosi, M., and Boccaccini, A. *Journal of Non-Crystalline Solids* **344**, 88–93 (2004).

[7] García, R. E., Carter, W. C., and Langer, S. A. *Journal of the American Ceramic Society* **88**(3), 750–757 (2005).

[8] García, R. E., Chiang, Y.-M., Carter, W. C., Limthongkul, P., and Bishop, C. M. *Journal of the Electrochemical Society* **152**(1), A255–A263 (2005).

[9] Garboczi, E., Bentz, D., and Martys, N. In *Methods in the Physics of Porous Media,* zen Wong, P., editor, 1–41. Academic Press, San Diego, CA, (1999).

[10] Roberts, A. and Garboczi, E. *J. Amer. Ceram. Soc.* **83**, 3041–3048 (2000).

[11] Shewchuck, J. (2002). preprint.

[12] Shewchuck, J. (2002). Sandia National Laboratories.

[13] GiD manual. http://gid.cimne.upc.es/.

[14] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. *Journal of Chemical Physics* **21**, 1087–1092 (1953).

[15] Webpage. http://matcase.psu.edu/index.html.

[16] Java applet. http://www.ctcms.nist.gov/ rlua/FE/refinery/.