Universal Meshes:
Enabling High-Order Simulation of Problems with Moving Domains

by
Adrián J. Lew, Ramsharan Rangarajan, Michael J. Hunsweck, Evan S. Gawlik, Hardik Kabaria,
Stanford University
and Yongxing Shen
Universitat Politecnica de Catalunya

What do crack propagation problems, fluid-structure interaction problems, phase-boundary evolution problems, and shape optimization problems have in common? By the time you finished reading the question the answer would have likely popped in your mind: they are types of problems motivated by important engineering applications in which finding the domain of the problem is part of the solution. These problems are more generally known as free or moving boundary problems. If these do not feel challenging enough, then try considering problems in which the boundary of the domain itself has some interesting dynamics. For example, the in-plane turbulent motion of a soap film induced by the surrounding air, the swimming of small microorganisms in which the thin vesicles that form their bodies are currently modeled as two-dimensional fluids with through-the-plane bending stiffness [1], and the propagation of a hydraulic fracture, in which the dynamics of the fracturing fluid on the crack surface is often modeled with Reynolds' lubrication equations [2].

It would not be an understatement to say that this class of problems has fascinated the computational mechanics community for decades now. The fundamental issue that needs to be addressed is, among several others, how to approximate both the domain and the solution as the domain evolves. It would be a mirage to pretend that within this article we could comprehensively review the universe of proposed methods for this class of problems. It is useful, however, to think about them in terms of the order of approximation of the domain; after all, the domain is part of the solution, so it should be approximated with the same order as the

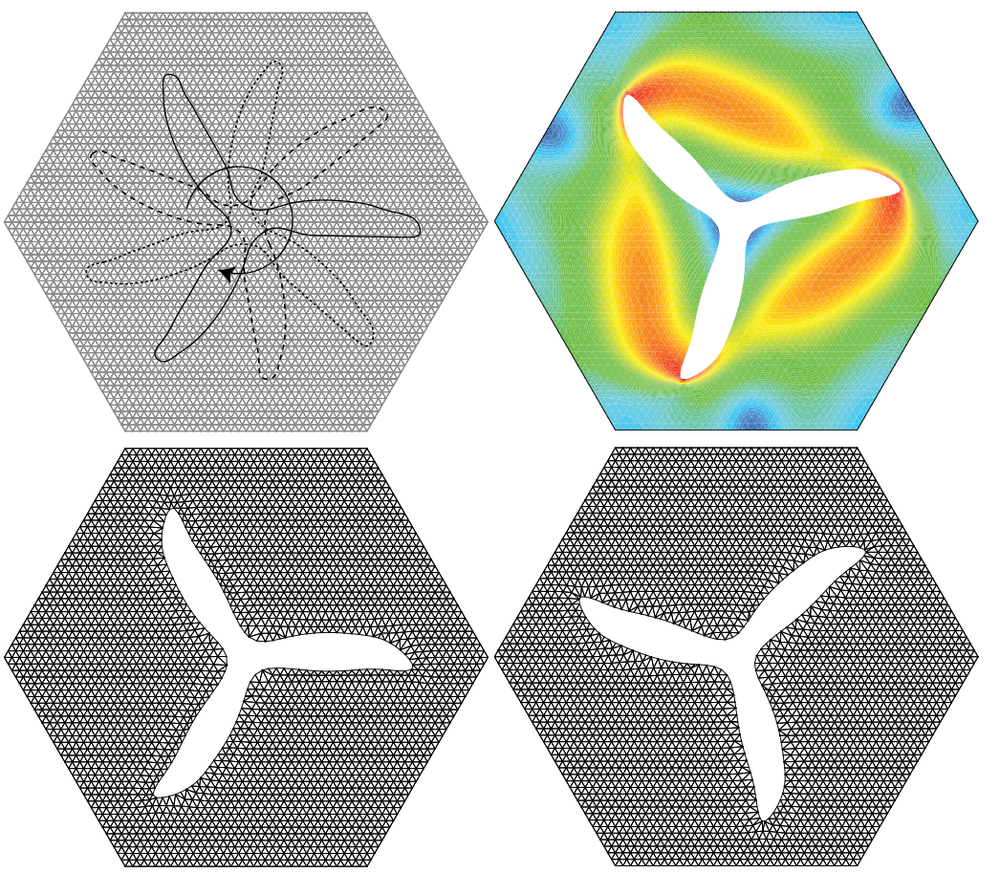


Figure 1: *Rigid propeller rotating at a constant angular velocity in a Newtonian fluid in two-dimensions. The Universal Mesh made of equilateral triangles (top-left) is deformed to exactly mesh the domain of the fluid for any angle of the propeller (bottom). Top-right: Snapshot of the speed contours.*

solution sought. For example, the so-called level set method (which truly stands for a variety of algorithms) is often applied on structured meshes and the domain is represented implicitly with a level set function. In most methods this function is piecewise affine, and this constrains the approximation of the domain to be at most second-order with the mesh size (for example, in the Hausdorff distance). Level set functions based on piecewise higher-order polynomials or rational functions are possible, but other complexities often appear when constructing approximation schemes of the same order for the solution. A similar remark applies to most embedded or immersed boundary methods, which either cut elements, as in extended finite element methods, or represent the boundary of the domain as a collection of connected segments [3,4]. As a second example we mention fluid-structure interaction methods based on an Arbitrary Lagrangian-Eulerian (ALE) description for the kinematics of the domain. In this context, approximations of the evolving domain of any order are possible. These methods are based on deforming a mesh that is fixed to a reference domain. Such deformation is described by a deformation mapping (or diffeormorphism), and its image is the deformed domain at each time instant. The limitation of these methods is found for some large deformations of the reference domain, such as in the propeller example of *Figure 1*. In these situations the deformed mesh can be highly sheared or even entangled, similar to what is often encountered when simulating solids under very large shear deformations.

Universal Meshes

This scenario prompted us to rethink the way to construct approximations to problems with moving domains. In particular, in a situation like the one in *Figure 1*, in which a mesh needs to be constructed for any possible position of the propeller, we made the observation that while the set of all elements intersecting the propeller at any single position (*Figure 1, top-left*) does not exactly mesh the domain, it is actually very close to doing so. Hence we wondered: would it be possible to simply deform such a set of elements so as to exactly match the domain? If we could do that for any position of the propeller, then the mesh on the background would be a *Universal Mesh* for all the domains needed in the problem.

It turns out that this idea is possible, so far in two-dimensions and with meshes made of triangles. The algorithm we proposed to do so is illustrated in *Figure 2* [5,6].

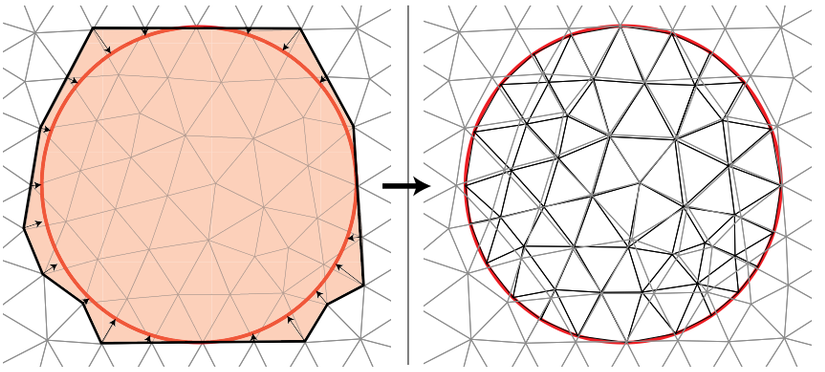


Figure 2:
Sketch of the algorithm applied to a circular domain (in red)
Left: *Elements with one node inside the domain are selected (in light red). Then their boundary (in black) is projected onto the circle with the closest point projection, and nodes in the interior of the circle are relaxed away from the boundary*
Right: *A theorem guarantees that the resulting mesh for the circle (in black) will have good quality elements. For comparison, the original Universal Mesh is also shown in the background (in gray)*

It consists of three steps:
(a) Loop over the elements in the mesh and select all elements that have at least one vertex inside the domain,
(b) project the boundary of the region formed by the (closure) of all these elements onto the boundary of the domain through the closest point projection, and
(c) relax some of the vertices near the boundary away from it, for example but not necessarily, along the direction normal to the boundary.
A video explanation of how this algorithm works can be found in [7], in which we showcase the robustness and speed of the algorithm through an interactive implementation in a tablet device. Given that the algorithm is pretty simple, it is fair to ask why it has not been proposed earlier (a related idea can be found in [8])? Well, a naïve application of this idea on arbitrary meshes, even Delaunay meshes, quickly reveals that elements with bad aspect ratios or inverted elements could easily appear [5]. We analyzed under what conditions on the Universal Mesh and the domain it is possible to use the algorithm above and obtain a mesh with a guaranteed lower bound on the element quality. The result is expressed as a theorem [6], and it essentially states that if (1) the domain is smooth (C^2), (2) the size of each element intersected by the boundary is

small enough (with upper bounds that depend on the local curvature or local feature size of the domain), and (3) some angles of the element, or better perhaps all angles in the mesh, are acute, then the algorithm renders a good-quality mesh, with the boundary of the mesh coinciding exactly with the boundary of the domain.

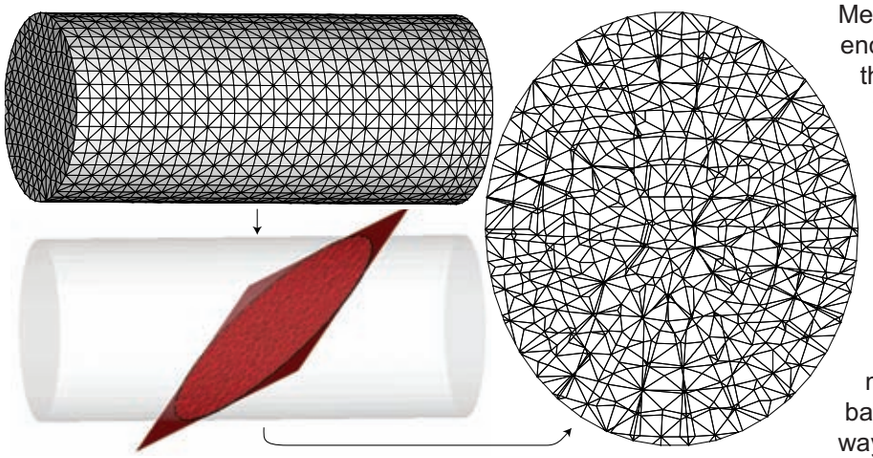


Figure 3:
Why is simulating hydraulic fractures a difficult problem? One reason is that a nice mesh is needed on the crack surfaces to solve for the motion of the fluid, and simply cutting elements does not render one

Furthermore, the upper bounds for the element mesh size near each point of the domain boundary can be explicitly computed, and used to robustly and automatically estimate whether a given Universal Mesh is able to mesh the domain or needs to be replaced. Another appealing feature of the algorithm is that it defines a one-to-one map between the domain and the closure of all the elements selected in step (a), and that by interpolating this map we precisely recover the isoparametric map. Consequently, when such interpolation is performed, the algorithm above can be simply regarded as a robust meshing preprocessor, and any elements of the user's liking can be adopted.

So, what is distinctive about a Universal Mesh then? A key advantage and difference over standard meshing algorithms is that for a problem with a moving domain the connectivity of the Universal Mesh is retained as the domain changes (as long as no new smaller features appear), and hence the data structures in the problem do not have to be rebuilt from scratch (or just minimally altered). Things like iterating over the geometry, often necessary when the domain itself is part of the solution, are now simple and possible. Algorithms based on a Universal Mesh have in some way the best of both worlds: the advantages of immersed boundary methods in that the geometry can undergo large changes or motions with minimal changes to the grid, and the accuracy of ALE methods that deform the mesh to approximate the domain.

This is the basic idea. We show some application examples next.

Applications

When we showcase the use of Universal Meshes we like to begin with the example of the propeller in *Figure 1*, since it precisely embodies the balance between ALE and immersed boundary methods that a Universal Mesh provides. To construct the spatial discretization for this problem

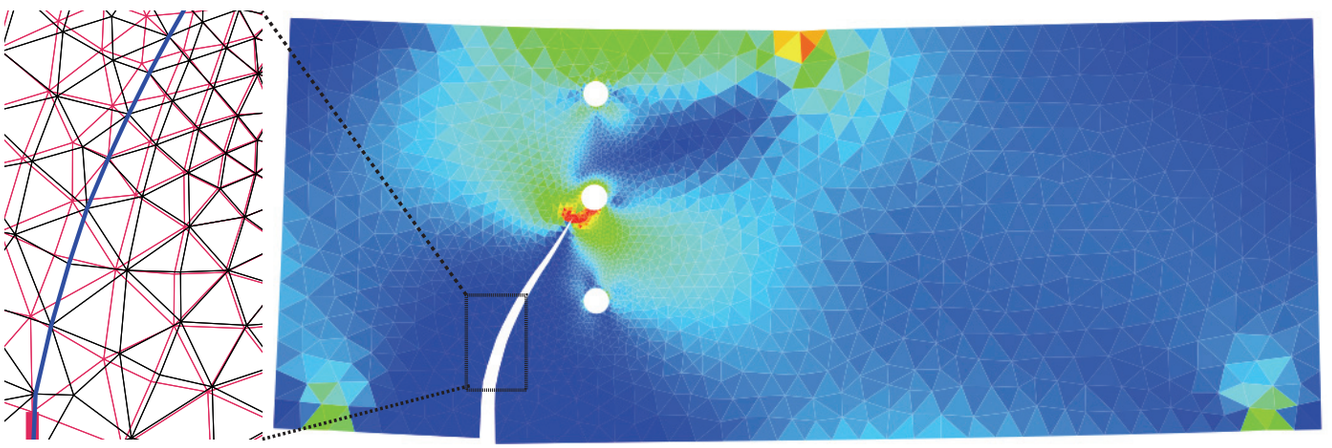


Figure 4:
A Universal Mesh used to simulate a traditional example of brittle crack propagation. The inset shows the Universal Mesh in red and part of the initial crack in a thick pink segment. The deformed mesh (in black) exactly meshes the crack path (in blue), so no elements were cut

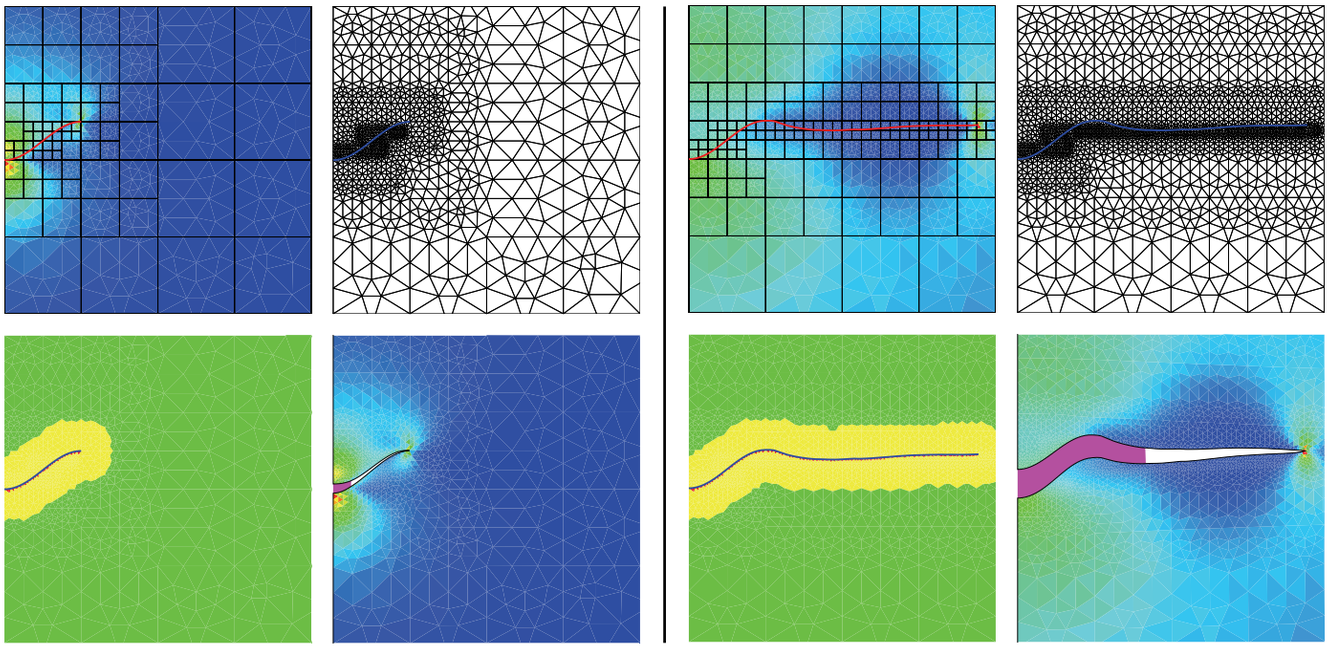


Figure 5:
Simulation of an initially curved hydraulic fracture in plane strain, with zero far-field stresses and symmetry conditions on the left. Each of the two snapshots shows the Universal Mesh (top-right), the quadtree used to refine it as the fracture evolves (top-left), the elements perturbed to accommodate the crack in the reference configuration (in yellow, bottom-left), and the von Mises stress in the deformed configuration of the rock (enlarged), with the fluid inside the crack in pink (bottom-right). At each loading step the Universal Mesh is deformed to exactly mesh the crack surface

we did not have to build any special elements, rather, we could choose among any of the stable combinations for incompressible flow: in this case a P2/P1 Taylor-Hood element. It would be nice to have the same flexibility for time integration, wouldn't it? It so happens that it is also possible to choose the preferred time integrator; we call it a plug-and-play approach, and for problems with smooth enough solutions, it enables the construction of methods of any order for problems with moving domains [9]. For the propeller we adopted a standard second-order Runge-Kutta method.

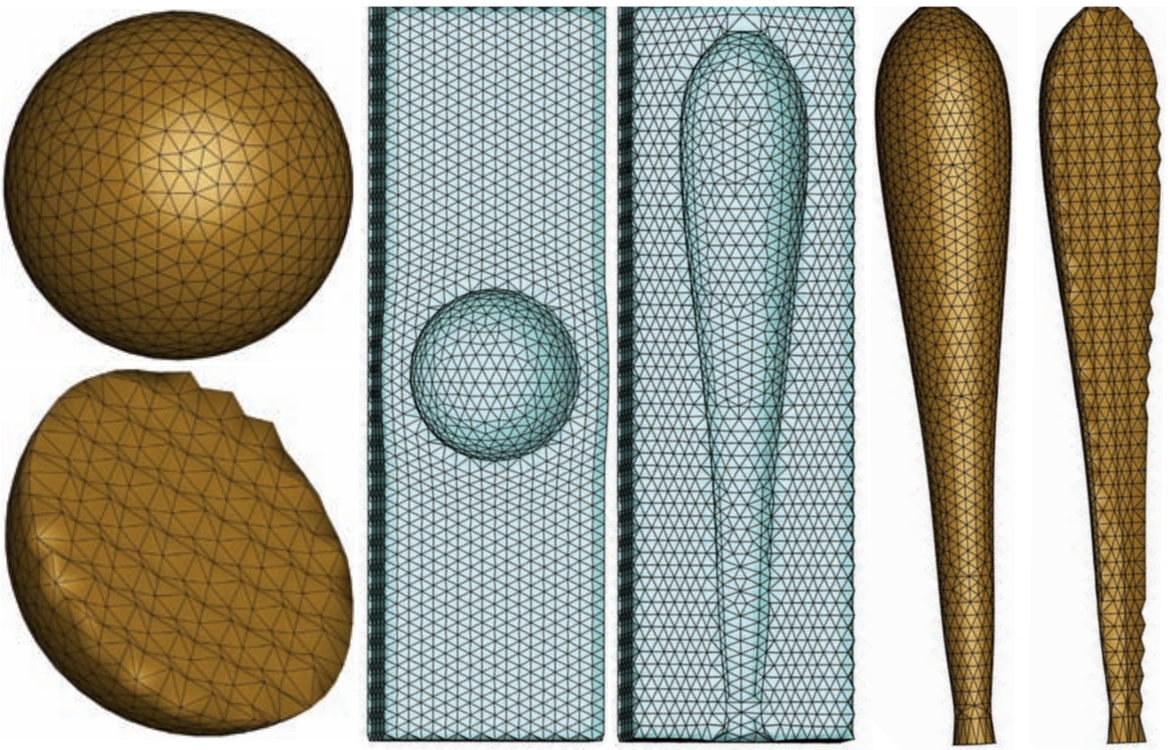
A different type of fluid-structure interaction problem was what motivated us a few years ago to create Universal Meshes. You may have heard about it lately, since it holds the promise to essentially change the global energetic landscape [10]. We are talking about the simulation of hydraulic fractures. Extensive reserves of natural gas are trapped in rocks with low permeability, and the main way to enhance gas flow is to massively fracture the rock by injecting fluid at high-pressure. The reason this problem is difficult to simulate is that not only do we need to allow arbitrary crack propagation, but as the crack surfaces evolve, we also need a reasonable mesh on them to solve the partial differential equations that describe the fluid motion through the crack. Simply cutting elements does not lead to a good mesh on the crack surfaces (*Figure 3*). In contrast, by deforming a Universal Mesh we are guaranteed to have a nice surface mesh on the cracks (*Figure 4*). Of course, for accuracy reasons the Universal Mesh often needs to be periodically changed, and this is what we show in *Figure 5* for an initially curved hydraulic fracture in plane strain (for straight fractures, see [2]). A peculiar aspect of these simulations is that at each time-step we need to iterate over possible cracks to find one that satisfies Griffith's criterion, and the Universal Mesh makes this a rather simple chore. On a more general note, the computation of the fluid motion on the crack surfaces is one example of how to take advantage of a Universal Mesh to solve partial differential equations on a manifold embedded in the mesh (not necessarily meshed).

" The beauty of a Universal Mesh is that we now know how to robustly deform it to exactly mesh a domain, provided the conditions laid out by the theorem are satisfied. "

Outlook

The beauty of a Universal Mesh is that we now know how to robustly deform it to exactly mesh a domain, provided the conditions laid out by the theorem are satisfied. So far we have a theorem in two-dimensions and for geometries without corners, so the story is just starting and a lot remains to be done. We have compelling reasons to believe that similar ideas will work in three-dimensions, and *Figure 6* shows a preliminary example. For those who may want to test them, we will soon have an open source code available at our website: lavxm.stanford.edu. ●

Figure 6:
Preliminary examples in three-dimensions. The same Universal Mesh of tetrahedra was deformed to exactly match the surfaces of a sphere and a baseball bat. The surface meshes shown in both objects are of good quality, and so are the elements in the interior, exposed here by removing all elements with some vertices above the cut plane. In the process, the exterior of each object has been meshed as well, and those meshes are displayed in the two central pictures



References

[1] *Marino Arroyo, Luca Heltai, and Antonio DeSimone*. **Reverse engineering the euglenoid movement.** Bulletin of the American Physical Society 57 (2012).

[2] *Michael J. Hunsweck, Yongxing Shen, and Adrián J. Lew.* **A finite element approach to the simulation of hydraulic fractures with lag**. International Journal for Numerical and Analytical Methods in Geomechanics (2012).

[3] *Adrián J. Lew and Gustavo C. Buscaglia*. **A discontinuous-Galerkin-based immersed boundary method**. International Journal for Numerical Methods in Engineering 76, no. 4 (2008): 427-454.

[4] *Adrián J. Lew and Matteo Negri*. **Optimal convergence of a discontinuous-Galerkin-based immersed boundary method**. ESAIM: Mathematical Modelling and Numerical Analysis 45, no. 4 (2011): 651.

[5] *Rangarajan Ramsharan and Adrián J. Lew*. **Parameterization of planar curves immersed in triangulations with application to finite elements**. International Journal for Numerical Methods in Engineering 88, no. 6 (2011): 556-585.

[6] *Rangarajan Ramsharan and Adrián J. Lew*. **Analysis of a method to parameterize planar curves immersed in triangulations**. arXiv preprint arXiv:1109.5890 (2011).

[7] **http://www.stanford.edu/group/lavxm/Ram.m4v**

[8] *M. Gonzalez and M. Goldschmit*. I**nverse geometry heat transfer problem based on a radial basis functions geometry representation**. International Journal for Numerical Methods in Engineering 65, no. 8 (2005): 1243–1268.

[9] *Evan S. Gawlik, Rangarajan Ramsharan and Adrián J. Lew*. **High-Order Finite Element Methods for Moving Boundary Problems with Prescribed Boundary Evolution**. In preparation.

[10] **TIME Magazine**, April 11, 2011.