Internship proposal
Locally-injective mapping

Dmitry Sokolov et Étienne Corman
dmitry.sokolov@univ-lorraine.fr, etienne.corman@cnrs.fr

Nancy, LORIA, Pixel team
`https://pixel.inria.fr`

1 Context

Mapping a 3D surface to 2D space (or a solid domain to 3D space) is one of the most important problems in geometry processing, because it is much easier for many applications to work in the map than to directly manipulate the object itself. Historically, maps were introduced to represent the surface of the Earth. Mathematically speaking, to compute a map of the Earth, we need to cut a sphere to obtain topological disc(s), and then deform it to make a flattening (Figure 1–top). At the end, the most fundamental property we have to ensure is the invertibility of the map, and the importance of this property cannot be overstated. The very idea of computing a map is being able to go back and forth between the object itself and its image without any ambiguity.

The most versatile way to represent geometric objects inside a computer is to discretize them. An object can be approximated by a mesh, i.e. a number of primitives such as polygons in 2D and polyhedra in 3D. Generally, for computing mesh deformations, using elasticity analogy was found to be very fruitful.

The idea is to say that a mesh represents an elastic material, whose stored energy of deformation can be measured as $\int_\Omega f(J)$, where Ω is the input domain, J is the Jacobian matrix of elastic deformation, and f is a measure of distortion. To choose f is, in fact, to choose the elastic material. Then, obviously, we want to minimize the stored energy of deformation.

Figure 1–bottom shows a discrete map of the Earth made with office supplies (rubber bands and push pins). First of all, we approximate a sphere by a polygonal surface (here by a dodecahedron), then we cut it into two topological discs. Finally, we represent each edge by an elastic band; we pin the boundary of the surface to flatten, and the stable position of the mesh inside provides a discrete map.

In fact, this procedure corresponds exactly to a method proposed by Tutte in 1963 in his famous paper "How to draw a graph" [Tut63]. It corresponds to the choice of material $f := \frac{1}{2}\|J\|_F$. For several decades, Tutte embedding remained the only way with theoretical guarantees of success: if we pin the boundary vertices to lie on a convex polygon, we are guaranteed to obtain an invertible map (free of folds). Unfortunately, there are severe limitations for Tutte embeddings. First of all, if the boundary is constrained to a non-convex polygon, folds may be present (Jacobian determinant of the map is not everywhere positive). Figure 2–left provides

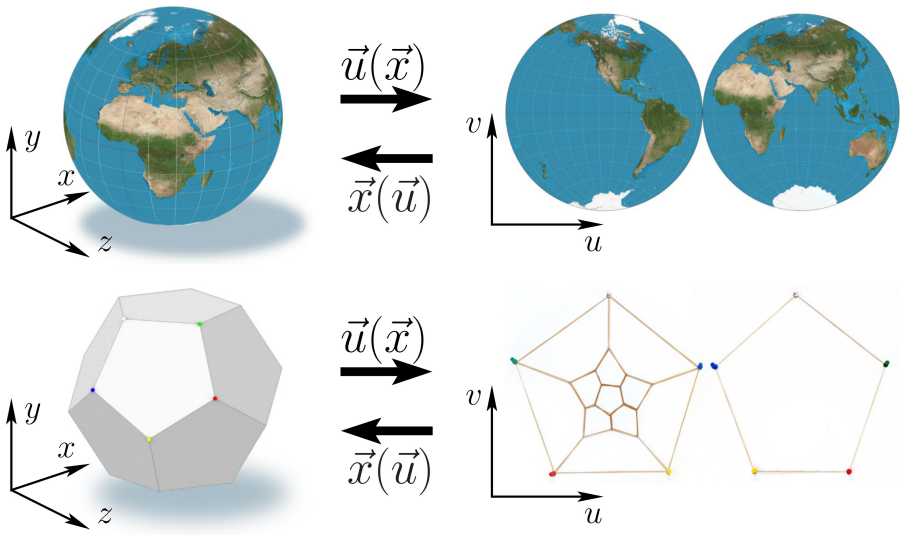

Figure 1: Top row: a map of the Earth by Nicolosi globular projection. Bottom row: a discrete map of the Earth via Tutte embedding made with office supplies (rubber bands and push pins).

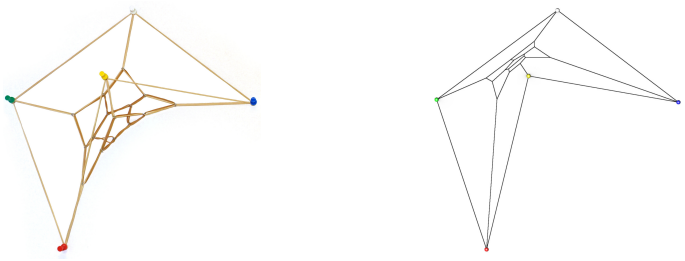

Figure 2: Tutte embedding may produce foldovers if the boundary is constrained to a non-convex polygon (left image), whereas our untangling procedure is guaranteed to produce valid results (right).

an illustration. This map contains foldovers, and thus is not invertible, there are points in the map with two distinct pre-images. In addition to that, Tutte embedding does not apply to free boundary deformations nor to 3D deformations, it only works for planar meshes with boundary constrained to a convex polygon.

Recently we proposed [GKK+21] the first mapping method free of the limitations of Tutte embeddings, but still with theoretical guarantees of success. Our method supports both constrained as well as free boundary mapping, and is also suitable to solid domains. The idea behind is very simple: first deform the mesh subject to necessary constraints (planarity, position of vertices etc.), ignoring eventual folds in the map. In this way we obtain a *tangled* mesh, a mesh with inverted elements. We have already seen an example of a tangled mesh in left image of Figure 2. Then, we *untangle* the mesh by solving a series of numerical optimization problems, penalizing the folds more and more until no inverted elements are present in the domain. Figure 2–right shows the resulting untangled mesh.

In other words, we progressively stiffen our material in certain places so it resists to folding. We need to solve a minimization problem $\min \int_\Omega f$, a number of times, and to do so, we use a very generic solver like L-BFGS [ZBN97], a quasi-Newton optimization algorithm for solving large nonlinear optimization problems. It employs function value and gradient information to search for the local optimum.

2 To do

While using black-box solver works pretty well in practice, it can be (it is) quite slow. We can do better by injecting into the solver some information about the nature of the problem we are solving. Recently, Rabinovich et al. [RPPSH17] proposed such a technique: they optimize for the fold-preventing energies indirectly by minimizing a simpler proxy energy and compensating for the difference with a reweighting scheme. For this internship we want to test whether such kind of methods can be used to speed up our computations.

3 Expected skills

The main quality expected is a desire to learn and to work as part of a team. On the other hand, it will be necessary to be sufficiently at ease with mathematics and computer science. The code should be fairly simple, since it essentially involves constructing the matrices and vectors corresponding to the discretization of the problem. The choice of programming language is up to you, but by default we would suggest C++ or Matlab.

References

[GKK+21] Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais, Nicolas Ray, and Dmitry Sokolov. Foldover-free maps in 50 lines of code. *ACM Transactions on Graphics*, 40(4):Article No.102, pp 1–16, July 2021. `https://arxiv.org/abs/2102.03069`.

[RPPSH17] Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. Scalable locally injective mappings. *ACM Trans. Graph.*, 36(2), apr 2017.

[Tut63] W. T. Tutte. How to Draw a Graph. *Proceedings of the London Mathematical Society*, s3-13(1):743–767, 01 1963.

[ZBN97] C. Zhu, R. H. Byrd, and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.