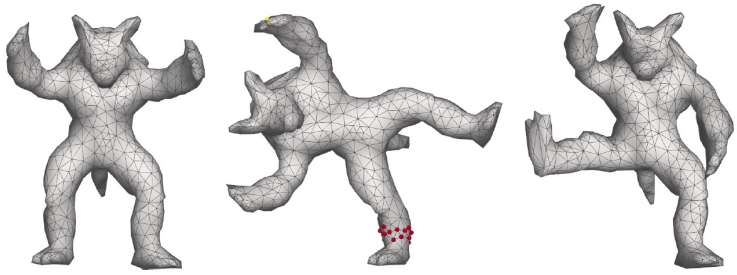
IFT 6113 - Assignment 3, 100 pts

October 30, 2024



In this homework you will understand more about vector fields and implement two deformation tools: biharmonic deformation and As-Rigid-As-Possible (ARAP).

Submission

Deadline: Monday, November 20th, 23:59

Submit you work via private post in Piazza with zip archive with your code and name (i.e. `hw3_python_john.zip`). It should contain: (1) your code; (2) derivations for the theory part and the practice problem 1.1; (3) screenshots of the flowlines in the theory part and the deformed surfaces; (4) instructions on how to use your code.

Make sure you understand every line of the code you write.

Requirements

You are allowed use built-in linear algebra solvers and decompositions, sparse matrix operations, cotangent laplacian, and mass matrix.

- **C++**: use `libigl` and `Eigen`
- **Python**: use `scipy.linalg` and `scipy.sparse.linalg` link

Code templates

Course github: github.com/ivanpuhachov/ift6113_2021

Theory Problem 1 (14 pts)

For a vector field on a plane $\mathbb{R}^2$, $u(x, y) = (y + 1, x - 1)$:

1. Try to integrate this vector field, i.e., try to find a function $f(x, y)$, such that $\nabla f = u$. Does such function exist? What are the conditions for its existence? Are those conditions satisfied for u? If f exists, is it unique?

2. Find this vector field's flow, i.e. such a mapping, for a given time t: $\psi_t : \mathbb{R}^2 \to \mathbb{R}^2$:

$$\frac{d\psi}{dt} = u(\psi_t).$$

   Show your work. Plot flowlines.

   Hint: This link (and the next page there) might be helpful:

   https://tutorial.math.lamar.edu/Classes/DE/SystemsDE.aspx

Practice Problem 0 (1 pt)

Implement a way to specify user anchor constraints, i.e. user would somehow select a subset of vertices and specify their target coordinates. Some vertices may remain on the same position (fixed points), some other are moved to new location. This can be done as an input text files or command line interface. Make it easy to try deformations on several models.

For example, you may store constraints for `dino.off` [1] in two files: `indices-dino.off.txt` has indices of points from mesh, `positions-dino.off.txt` has positions for specified vertices.

Provide instructions on how to use your code.

Practice Problem 1 (35 pts)

Your task is to derive a solution to optimization problem (on paper), implement it and show the results.

[1] `.off` files can be parsed with the same `libigl` functions you use for `.obj`

Recapping what we studied in the lectures, we formulate our deformation as an optimization for a displacement vector field $d : M \to \mathbb{R}^3$, discretized on mesh vertices. **Biharmonic deformation** algorithm looks for the vector field minimizing the following quadratic energy with the linear user constraints:

$$\min_d \int_\Omega \|\Delta d\|^2 d\Omega$$

$$\texttt{s.t.} \quad d_{user} = \hat{d}_{user}$$

where $d_{user} = \hat{d}_{user}$ stands for fixing the displacement of certain vertices.

After discretization, this becomes the following quadratic optimization, where L is the familiar cotangent Laplacian matrix, M is the mass matrix, and you can think of $K = L^\top M^{-1} L$ as the discretization of the squared Laplacian (bi-Laplacian):

$$\min_d \int_\Omega \|\Delta d\|^2 d\Omega \approx \min_d d^\top K d$$

$$\texttt{s.t.} \quad d_{user} = \hat{d}_{user}$$

1. How to solve this optimization problem? What algorithm should you use? Show your work.

2. Implement this solution and show your results on `armadillo_1k.off`. Add screenshots to your submission, save the displacement settings to reproduce quickly.

Practice Problem 2 (50 pts)

Your task is to implement **ARAP** deformation and show how it works (including both final result and intermediate iterations). Save the displacement settings to reproduce quickly.

Refer to the course slides and the original paper: Olga Sorkine and Marc Alexa, "As-rigid-as-possible surface modeling", 2007. igl.ethz.ch/projects/ARAP/

In short, to minimize ARAP energy by local-global algorithm, we alternate between two steps:

- (local step) finding the optimal rotations R_i assuming the vertex positions p are fixed
- (global step) finding the optimal vertex positions p assuming all rotations R_i are fixed

Run several iterations (until you are satisfied with deformation result). Show the mesh after each global iteration.

Hint: Initialize rotation matrices to identity matrices and implement the global solve. Run it once, without the local step. Debug your code on simple meshes like `bar-2.off`

Debug hint: try setting only 1 user constraint: move 1 point to a new location. What is the expected behavior? Does your algorithm behave as expected?

Debug hint: try three point constraints specifying a single perfect rotation.

Bonus: Since for each iteration you are solving a linear system with the same matrix, you have a chance to improve the efficiency. Find a way to pre-factorize (precompute) the matrix before iterations and show how it improves computation time.

Reference images

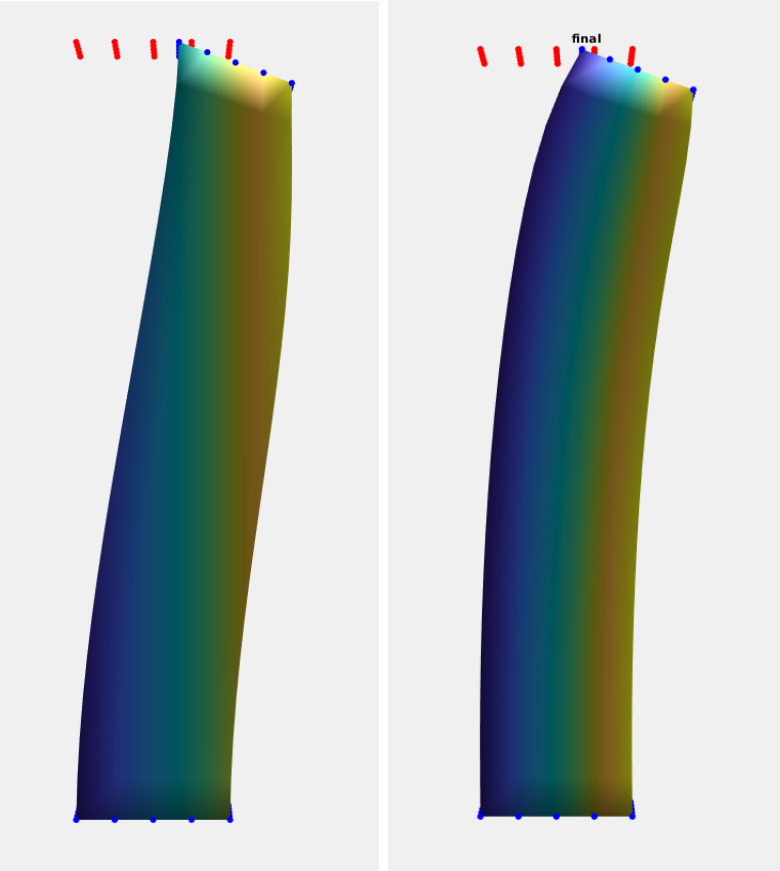


Figure 1: Biharmonic deformations (left) and ARAP (right) for `bar2.off`