Homework 2: SVD and Least-Squares

Due October 12, 2023

In this homework, we will derive an implement the algorithm described in "As-Rigid-As-Possible Surface Modeling" (Sorkine and Alexa, Symposium on Geometry Processing 2007), known in graphics as ARAP. You are encouraged to read the ARAP paper and to use it to help solve problems in this homework, although you must give complete derivations rather than just referencing the equations from the paper; note the model in this homework is a slightly simplified version of theirs.

ARAP deforms 3D triangle meshes according to motions of a few handle points, as in the following image:

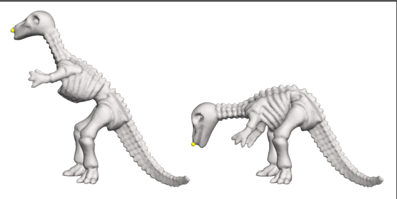

Figure 1: *Large deformation obtained by translating a single vertex constraint (in yellow) using our as-rigid-as-possible technique.*

We will build the ARAP model from the ground up and then implement it numerically.

Problem 1 (15 points)**.** First, a few warm-up problems to review materials from lecture:

(a) Suppose $A \in \mathbb{R}^{m\times n}$ is factored $A = QR$, with $Q \in \mathbb{R}^{m\times n}$ and $R \in \mathbb{R}^{n\times n}$. Show that $P_0 = I_{m\times m} - QQ^\top$ is the projection matrix onto the null space of $A^\top$.

(b) Suppose $A \in \mathbb{R}^{m\times n}$ with linearly independent columns. Show how to obtain a QR factorization of A from the Cholesky factorization of $A^\top A$ and vice versa.
Hint: If $A = LL^\top$, show that the columns of $A(L^\top)^{-1}$ are orthogonal.

(c) Suppose $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ with $\mathbf{u}^\top \mathbf{v} = 1$. How many iterations does power iteration take to converge to the dominant eigenvalue of $A = \mathbf{u}\mathbf{v}^\top$?

Problem 2 (10 points)**.** Suppose $f : \mathbb{R}^3 \to \mathbb{R}^3$ is a mapping of 3D space that preserves distances between points; we will call such a mapping *rigid*. Argue there exists a matrix $R \in \mathbb{R}^{3\times 3}$ satisfying $R^\top R = I_{3\times 3}$ and a vector $\mathbf{b} \in \mathbb{R}^3$ so that $f(\mathbf{x}) = R\mathbf{x} + \mathbf{b}$ for all $\mathbf{x} \in \mathbb{R}^3$.

Problem 3 (30 points)**.** ARAP is built on a model where every vertex $\mathbf{p}$ of the mesh and its neighbors $\{\mathbf{q}_i\}_{i=1}^{n_\mathbf{p}}$ move *nearly* rigidly.

(a) Suppose $f : \mathbb{R}^3 \to \mathbb{R}^3$ is rigid, as in Problem 2. Argue that $f(\mathbf{q}_i) - f(\mathbf{p}) = R(\mathbf{q}_i - \mathbf{p})$.

(b) Now, let's reverse the situation. Suppose we are given vectors $\mathbf{p}, \mathbf{p}' \in \mathbb{R}^3$, matrices $Q, Q' \in \mathbb{R}^{3\times n_\mathbf{p}}$ (with columns $\mathbf{q}_j, \mathbf{q}'_j \in \mathbb{R}^3$, resp.), and nonnegative weights $\{w_j\}_{j=1}^{n_\mathbf{p}}$. Provide an algorithm for finding the matrix $R \in \mathbb{R}^{3\times 3}$ that solves the following optimization problem:

$$\min_{R^\top R = I_{3\times 3}} \sum_{j=1}^{n_\mathbf{p}} w_j \| R(\mathbf{q}_j - \mathbf{p}) - (\mathbf{q}'_j - \mathbf{p}') \|_2^2.$$

This problem finds the rigid R from part (a) that best approximates the map from unprimed to primed variables.

(c) In the attached code, implement `findARAPRotations`, which takes a sparse matrix of weights w_{ij} as well as sets of points $\{\mathbf{p}_i, \mathbf{p}'_i\}$ and finds a set of matrices R_i such that each R_i solves

$$\min_{R_i^\top R_i = I_{3\times 3}} \sum_j w_{ij} \|R_i(\mathbf{q}_j - \mathbf{p}_i) - (\mathbf{q}'_j - \mathbf{p}'_i)\|_2^2.$$

Note: In Homework 2, aim for code correctness rather than efficiency; your final ARAP implementation may take up to 1 minute to compute a deformation on a standard laptop.

Problem 4 (30 points)**.** Suppose we are given a triangle mesh with n vertices with positions $\{\mathbf{p}_i\}_{i=1}^n \subset \mathbb{R}^3$; for a subset of vertices $S \subseteq \{1, \ldots, n\}$ we are given displaced positions $\overline{\mathbf{p}}'_i$. ARAP seeks deformed vertex positions $\{\mathbf{p}'_i\}_{i=1}^n$ and matrices $R_i \in \mathbb{R}^{3\times 3}$ solving the problem:

$$\begin{array}{rl} \min_{\{\mathbf{p}'_i\},\{R_i\}} & \sum_{i=1}^n \sum_j w_{ij} \|R_i(\mathbf{p}_j - \mathbf{p}_i) - (\mathbf{p}'_j - \mathbf{p}'_i)\|_2^2. \\ \text{subject to} & \mathbf{p}'_i = \overline{\mathbf{p}}'_i \ \forall i \in S \\ & R_i^\top R_i = I_{3\times 3} \ \forall i \in \{1, \ldots, n\} \end{array}$$

(a) In words, explain the variables and objective of this optimization problem. How would you use solvers for this problem in 3D modeling software?

(b) Assuming $w_{ij} = w_{ji}$ for all i, j, take the gradient of the objective function with respect to $\mathbf{p}'_i$.

(c) By setting your answer to the previous part to $\mathbf{0}$ and moving constants with respect to $\{\mathbf{p}'_i\}_{i=1}^n$ to the right-hand side, derive a linear condition satisfied by the every $\mathbf{p}'_i$ with $i \notin S$.
Hint: The left-hand side of your expressions should depend on the w_{ij}'s and $\mathbf{p}'_i$'s, while the right-hand side should depend on the R_i's and the $\mathbf{p}_i$'s.

(d) Define an $n \times n$ matrix A and an $n \times 3$ matrix B so that $A^{-1}B$ gives an $n \times 3$ matrix whose rows are the optimal $\mathbf{p}'_i$'s.

(e) In the attached code, implement `ARAPMatrix` and `ARAPRHS`, which compute A, B.

Problem 5 (15 points)**.** The ARAP algorithm alternates between optimizing for the R's with the $\mathbf{p}'$'s fixed ("local step") and optimizing for the $\mathbf{p}'$'s with the R's fixed ("global step"). Implement ARAP in the attached code. For full credit, you should only compute the matrix A from the previous problem once, and you should use pre-factorization to accelerate the linear solves.

Problem 6 (Extra credit, 20 points maximum)**.** Here are some possible extensions to ARAP:

(a) Our barebones implementation optimizes for a fixed number of loop iterations. Propose and implement a criterion for measuring (near-)convergence of the algorithm.

(b) Optimize your implementation of ARAP so that each iteration takes ≤ 0.1 seconds.

(c) The matrix A in our implementation is asymmetric. Reformulate ARAP to use a symmetric, positive definite matrix in the global step. Is your final implementation more efficient?

(d) Suppose we add a constraint $\det R_i = 1$ to avoid flipping the surface inside-out. How does this change the local step? Derive the change mathematically.