3D Geometry Processing
Exercise 6

Fall 2013

Hand in: 5.12.2013, 16:00

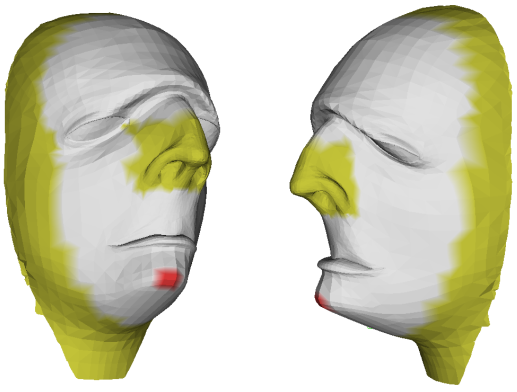

Figure 1: Two views on a deformation of the head.obj mesh. Two deformations were applied: first the nose was pushed slightly into the face and translated vertically, then the chin was slightly pulled to the front.

As Rigid as Possible Surface Modeling

In this assignment, your task is to implement an interactive modeling system based on the "as rigid as possible surface modeling" algorithm (henceforth RAPS), discussed in the lecture [1]. The general setting is the following: a user loads a mesh and picks two regions; the first is constrained to stay fixed, the second can be rotated and translated interactively by the user. Your system should then reconstruct the mesh in the background, such that it fulfills the two constraints while being deformed in a plausible way.

In the base code you will find a crude GUI which supports the selection of regions, as well as their translation and rotation. Feel free to extend the GUI and to enhance the user experience.

[1] See also the original paper `http://igl.ethz.ch/projects/ARAP/arap_web.pdf`

The Algorithm in a Nutshell

As discussed in the lecture, the RAPS algorithm tries to minimize the following energy, while fulfilling additional user specified constraints:

$$E(S') = \sum_i \sum_{j \in N_i} w_{ij} ||(p'_i - p'_j) - R_i(p_i - p_j)||^2. \tag{1}$$

Here p are the positions of the original undeformed mesh S, p' are the a priori unknown positions of the deformed mesh S', R_i are the unknown rotations which describe the rigid transformations of one-neighborhoods of the original mesh to the one-neighborhoods of the deformed mesh, and w_{ij} are some weights, which turn out to be nothing else than the cotangent weights. This energy expresses, that the sought deformed mesh S' should locally be approximated as well as possible by rotations, thereby preserving local details.

To find both the optimal rotations and the deformed positions an iterative approach is taken:

- All rotations are set to the identity
- Iterate:
  - Find the optimal positions, given the current per vertex-neighborhood rotations.
  - Find the optimal rotations, given the current deformed vertex positions.

These substeps are described in detail in the following exercises.

Installing the Base Code

As this is the first time this exercises are held, the base code for this assignment is not completely consistent with the basecode before (of course this is also a good demonstration, that code should always be distributed via a git repository, and never! via zip files...). The changes in already distributed files are minor; if you do not want to just copy everything into your project and potentially break your old code, have a look at the readme.pdf in the .zip package, where the file changes are documented in detail. After installing the basecode, running Assignment6_interactive.java should open a GUI that allows the picking, dragging and rotation of subsets of the displayed mesh.

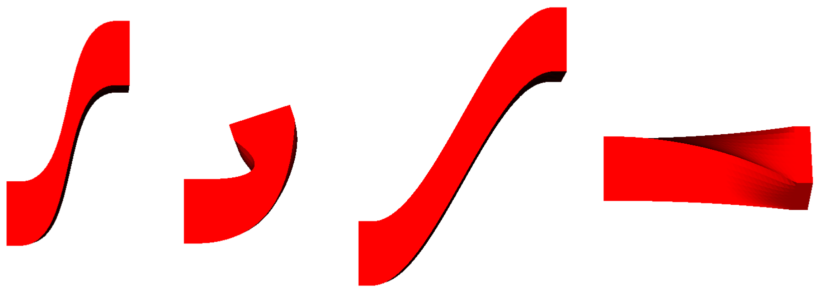

Figure 2: Result of running the demos from the class Assignment6_examples. The equation 2 is solved only once for optimal positions, while all the rotations are set to the identity.

1 Optimal Positions

As discussed in the lecture, minimizing the described energy from Equation 1 for fixed rotations amounts to solving:

$$0 = \frac{\partial}{\partial p'_i} E = \sum_{j \in N_i} 4w_{ij}((p'_i - p'_j) - 0.5f(R_i + R_j)(p_i - p_j)$$

which is equivalent to

$$\sum_{j \in N_i} w_{ij}(p'_i - p'_j) = \sum_{j \in N_i} 0.5 w_{ij}(R_i + R_j)(p_i - p_j).$$

When using our old acquaintances the cotangent edge weights $w_{ij} = cotan(\alpha) + cotan(\beta)$, the left-hand side simply is the *unnormalized* cotangent Laplacian (not devided by any areas) of the original mesh applied to the deformed positions. The positions p' have to satisfy

$$L_S^{cotan\ unnorml.} p' = b \quad \text{where} \quad b = \sum_{j \in N_i} 0.5 w_{ij}(R_i + R_j)(p_i - p_j).$$

To incorporate additional positional user constraints $p' = p_{constr}$ we resort to the normal equations to satisfy all constraints in the least squares sense, while giving the user constraints some weight w.

$$\left(L^T \quad w \cdot I|_{constr}\right) \cdot \begin{pmatrix} L \\ w \cdot I|_{constr} \end{pmatrix} p' = (L^T L + w^2 I)p' = L^T b + w^2 I|_{constr} p_{constr} \qquad (2)$$

1. Implement the step to find optimal positions for given rotations by solving Equation 2. The class `RAPS_modeling.java` provides a skeleton for the RAPS algorithm and in the class `Assignment6_examples.java` you find a set of example deformations. Test your current deformation algorithm on these examples, and compare your results to the results depicted in Figure 2. *Details*

   - Choose a large weight w for the user constraints of at least 100.

- As always, boundaries have to be treated separately. For simplicity, we assume that the boundary is always constrained explicitly. Set the laplacian to zero on lines corresponding to boundary vertices, as we already did in earlier assignments. The Laplacian looses its symmetry, so be sure to compute $L^T L$ in Equation 2.
- To solve the linear system, you can either use the iterative JMTSolver() with the current deformed positions as an initial guess, or the provided Cholesky solver, which explicitly inverts the provided matrix using a Cholesky decomposition. The Cholesky decomposition has to be recomputed every time the set of handle vertices change. The Cholesky solver is slightly faster during a deformation session, but needs time to decompose the matrices. If the Cholesky decomposition fails, your matrix is slightly erroneous. Either it is not symmetric, or it is not positive definite.
- For another speed gain, avoid recomputing the cotangent weights every time you compute right hand sides; compute them only once and recycle them.

2. The class `Assignment6_interactive.java` demonstrates the usage of the picking GUI. The class DeformationPickingProcessor.java provides the callback functions for the Picking display. Adapt this class such that your current deformation algorithm can be used interactively.

5 Points

2 Optimal Rotations

The second step is to find the optimal rotations for fixed deformation positions p'. The error energy $E(S') = \sum_j w_{ij}||(p'_i - p'_j) - R_i(p_i - p_j)||^2$ can be minimized separately for each vertex v_i, by finding an optimal rotation R_i. Writing the squared norm as a dot product and dropping all terms not depending on R_i yields that the minimizing rotation maximizes the following expression, denoting the differences $p_i - p_j$ and $p'_i - p'_j$ by e_{ij} and e'_{ij} respectively:

$$R_i = \arg\max_{R_i} \sum_j w_{ij} e'^T_{ij} R_i e_{ij}.$$

As it turns out, this maximizing rotation R_i can be solved for using a singular value decomposition [2]. If for the matrix $S_i = \sum_{j\in N} w_{ij} e_{ij} e'^T_{ij}$ the svd is given by $S_i = U_i \Sigma_i V i^T$, the rotation R_i is described by

$$R_i = V_i \tilde{U}_i^T,$$

where $\tilde{U}$ is U, but with the last column (corresponding to the smallest singular value) scaled by -1 or 1 such that the determinant $\det(\tilde{U})$ is positive.

You can find an implementation of the SVD decomposition of 3x3 matrices in the basecode, in the class `linalg3x3.java`.

1. For every vertex, compute the optimal rotations R_i as described. Use absolutes of contangent weights $w_{ij} = |cotan(\alpha) + cotan(\beta)|$; negative weights tend to produce artifacts. Test your method:

[2]`http://igl.ethz.ch/projects/ARAP/svd_rot.pdf`

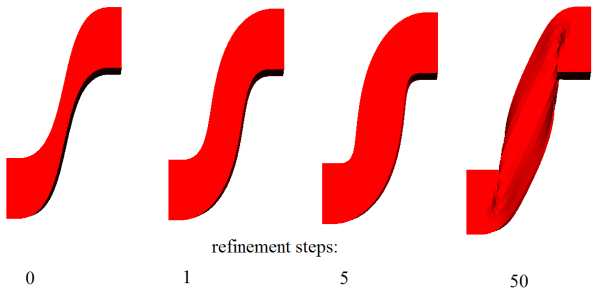

Figure 3: After setting the rotations to the identity, the optimal positions are computed. Then a number of rotation/position refinement iterations are made. Note how the process converges to a result that would be plausible for a fully rigid surface (think of a non stretchable cloth for instance) but which is visually not very pleasing. So it might be worth to stop iterating after only a few iterations.

   (a) Compute the optimal rotations for a mesh that has not been deformed; the identity matrix should be found for every vertex.

   (b) Rotate and translate the mesh (without any additional deformations) and compute the optimal rotations. The rotation you applied should be found for every vertex.

2. Finish implementing the RAPS deformation algorithm. Experiment with the example deformations provided in Assignment6_examples.java and prepare demos or snapshots to illustrate your results.

   (a) What is the impact of the number of refinement iterations on the example deformations provided in Assignment6_examples.java.

   (b) In the continuous deformation demonstrations, what is the impact of not resetting the rotations to the identity at the beginning of each iterative deformation optimization? For which deformations do you obtain better results?

3. Adapt the classes Assignment6_interactive.java and DeformationPickingProcessor.java such that your implementation of the RAPS algorithm can be used interactively. Save snapshots of some of your modeling sessions, or store the resulting meshes using the provided OBJWriter.java. Note that your meshes should not be too large, with at most a few thousand vertices.

4. **Bonus points:** Improve the GUI and the user experience. The bonus point(s) are distributed based on how good you sell your GUI enhancement in the demo session (: . Example ideas:

   - Make the rotation input more intuitive by centering a trackball at the center of the region of interest and interpreting rotations as rotations around the center of the region of interest.
   - Plot a handle, similar as the handles shown in the lecture slides.

- Make selection more intuitive. For example allow the selection of only visible vertices and adapt or reimplement the selection method to work correctly also when objects are close to the camera.
- Any improvement or combination with further tools you can think of.

(5 Points + bonus point(s))

3 Hand-In

Don't forget to:

1. Commit your code via the Ilias exercise page before the deadline.
2. Prepare your code demonstration and reserve a time slot for your demo via the Google Doc shared in the forum.