Final Report Group 06

An implementation of the *As-Rigid-As-Possible* surface modelling scheme

As our final project of the *3D Scanning & Motion Capture* lecture we implement the As-Rigid-As-Possible (ARAP) algorithm for mesh deformations in a 3-dimensional environment. Depending on a user's set of control points and predefined constraints, it is possible to render a visually appealing mesh deformation. The mesh itself can be freely rigged in real-time by enabling and disabling vertices as control points and then dragging them to a desired location. The ARAP modelling algorithm is an iterative scheme which enables a simulation to produce intuitively realistic deformations. The problem is solved using linear constraints.

1 Introduction

Shape deformation of scanned or sculpted 3D objects is used in many visualization tasks and plays an important role in modern computer visualization. However, for a great majority of the deformation tasks from visualization of specific areas of the input model to animation creation, modelers often times want to preserve the general shape of their meshes. When performing a smooth large-scale deformation on an object, like moving the arm of a human, this is intuitively accomplished by preserving the local structure of the object. This means that in this case, the arm is only rotated and translated, but not scaled or sheared. Thus it follows that when local shapes of the object are changed as rigidly as possible, surface details of the mesh tend to be preserved.
The ARAP algorithm implements a shape deformation framework based on the above observation to maximize local rigidity. Starting from polygonal meshes, the algorithm tries to find an ideal deformation by keeping the surface in each local vertex fan as rigid as possible. Local rigidity is measured by aligning each deformed cell with its original shape using e. g. the Procrustes algorithm [6] and thus calculating the local deviation from rigidity when applying a rotation matrix per vertex fan. Consequently, all calculated local cell deviations are summed up in an energy function. When picking and moving a cell of the mesh, an optimization of the energy function can then be used to enforce the movement of all other cells in the mesh to be similar to the deformed part, while maximizing local rigidity.

2 Related Work

Our implementation is based on the paper of Liu et al. [4] and Alexa et al. [1] which introduce the ARAP scheme.
For notations, the structure of a given triangle mesh S and its deformed form S' is

specified by n vertices, vertex positions $p_i \in R^3$ and $p'_i \in R^3$ respectively, as well as m triangles. $\nu(i)$ defines the set of vertices connected to a vertex i via edges in the mesh. The edges of vertex i are defined as $e_{ij} = p_i - p_j,\ j \in \nu(i)$.

2.1 Rigidity Measures

Liu et al. observe that a rotation matrix R_i exists for every rigid transformation between two mesh cells $C \to C'$:

$$p'_i - p'_j = R_i(p_i - p_j),\ \forall j \in \nu(i) \tag{1}$$

However, a complete local rigid transformation of any surface deformation scenario is impossible to achieve, since it would thus follow that the surface is globally rigid. In favor of maximizing local rigidity and preserving local cell shape between the deformations, Liu et al. suggest the optimization of an energy term that fits the above equation in order to solve for the best approximation of the local rotation R_i.

$$E(C_i, C'_i) = \sum_{j \in \nu(i)} w_{ij} \left\| e'_{ij} - R_i e_{ij} \right\|^2 \tag{2}$$

where w_{ij} are edge weights applied to make the function as mesh-independent as possible and the edge notation $e_{ij} = p_i - p_j$ is used for simplicity. The proper definition of the weights is discussed in section 4.
In order to measure and optimize the rigidity of the deformation of the whole mesh and to solve for optimal translations and rotations, the above equation of deviation from rigidity per cell can then be applied to every vertex in order to formulate a weighted global energy function:

$$E(S') = \sum_{i=1}^{n} w_i E(C_i, C'_i) = \sum_{i=1}^{n} w_i \sum_{j \in \nu(i)} w_{ij} \left\| e'_{ij} - R_i e_{ij} \right\|^2 \tag{3}$$

where w_i define specific cell-weights.

2.2 Procrustes Algorithm

The procrustes algorithm is a well-known method of shape comparison that aims at aligning two objects, given known correspondences. In this paper the algorithm is applied to the above defined problem to solve for as-rigid-as-possible deformations of an input mesh. In order to minimize the global energy term defined in equation 3, one has to calculate optimal rotation matrices R_i that map and align the original undeformed vertex fans to their new deformed state. One can observe that each term in equation 3 incorporates only the local transformations R_i, thus a solution that aims at optimizing rigidity for local cell transformations without regard for the transformations of other cells is also a solution for the global problem.
Given a problem to find the best rotations R_i to align the deformed edges of a vertex

fan e'_{ij} with their original position e_{ij}, the procrustes algorithm can be employed to first precompute the covariance matrix S_i as

$$S_i = \sum_{j \in \nu(i)} e_{ij} e'_{ij} \tag{4}$$

and then using the singular value decomposition of $S_i = U_i \Sigma_i V_i'$ to compute the optimal rotation R_i [6]:

$$R_i = V_i U_i^T \tag{5}$$

Since the global energy equation 3 incorporates weights, the local minimization also needs to consider these:

$$S_i = \sum_{j \in \nu(i)} w_{ij} e_{ij} e'_{ij} \tag{6}$$

3 Implementation

The implementation of the ARAP algorithm itself consists mainly of three parts: A weight-based initialization, a local solving step, as well as a global step, which produces the transformed vertex positions for the entire mesh. This scheme reflects a kind of *flip-flop-pattern* by first solving for local then for global deformations per iteration, as can be seen in Figure 1. The application is based on OpenGL utilizing functionalities of the *OpenMesh* library [8] for mesh rendering purposes, *Eigen* [3] for linear algebraic computations including the usage of a linear solver for the local and global solving step, and *ImGui* [2] as graphical user interface library. Contrary to our previous expectations, a non-linear solver was not required. Additionally, Matcaps were used to increase user experience when interacting with the mesh. The assets were taken from an online matcap library [5].

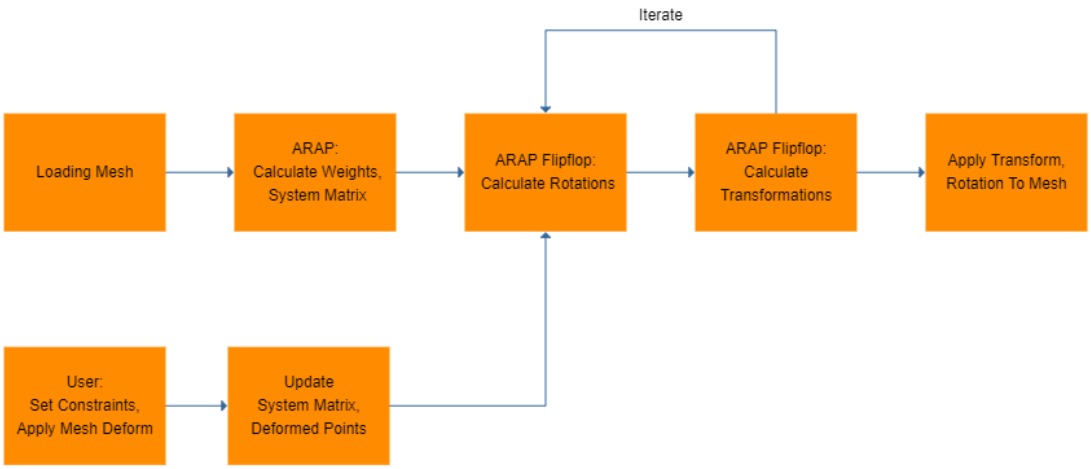

Figure 1: Overview of the ARAP algorithm.

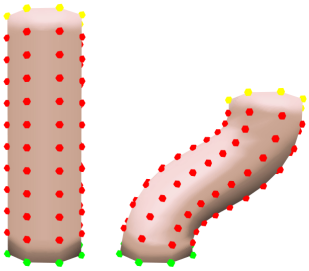

Figure 2: ARAP performed on a cylinder.

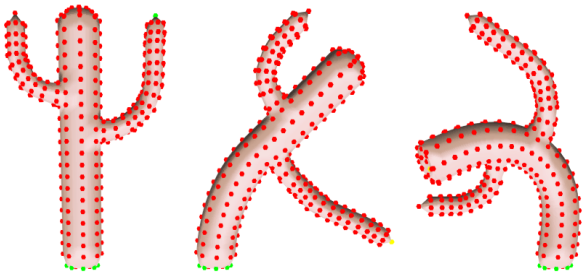

Figure 3: ARAP performed on a cactus. Undeformed; deformed with one control point on the right arm; deformed with one control point on the top.

3.1 Initialization of Weights and System Matrix

One of the most difficult aspects was the search for a properly suited weight function. The initial approach revolved around using the cotangent for computing weights for a given vertex v_i, assigning a weight $w_{i,j}$ to each outgoing edge. The vertex v_i and its neighbors v_j are called a vertex-fan and at the initialization phase, weights are computed based on the initial mesh structure for each vertex-fan. Assuming, a mesh contains the vertices v_1, v_2 and v_a, which form a triangle $\Delta v_1 v_2 v_a$, and v_b, which forms another triangle with v_1 and v_2, $\Delta v_1 v_2 v_b$, then the weights for edge $v1v2$ are computed using the angles α and β at v_a and v_b formed by the vectors $v_a\vec{v}_1$, $v_a\vec{v}_2$ and $v_b\vec{v}_1$, $v_b\vec{v}_2$ respectively. This works perfectly for practically quad-based meshes, that are often found computer graphics as it effectively sets the weight of the diagonal edge of each quad to zero. However, we found that this weight function can be unstable in meshes with obtuse angles, as it yields negative weights. We decided to clamp these negative weights to zero.

$$w_{1,2} = \max\left(0,\ \frac{1}{2}(\cot\alpha + \cot\beta)\right) \tag{7}$$

4 Results

The implementation performs as expected: meshes with a vertex count in the order of 1000 can be manipulated fluently, and the resulting meshes look as anticipated. See figures 2 and 3.

What came as a bit of a surprise to us, is that the algorithm takes a high number of iterations to converge. The ARAP algorithm is ill-suited to propagate rotations across long parts of a mesh, whereas translations are propagated instantaneously, due to the global solving step. This is also noted in the paper [7], where the importance of the *initial guess* is stressed.

5 Conclusion

The implemented ARAP algorithm [7] provides a live robust and efficient shape deformation framework with smooth and rigid-like transformations that preserve the local shapes of the source mesh.
The ARAP algorithm can be further enhanced by combining the basic algorithm with additional physical effects of mesh deformations, for example the complementary dynamics approach by Zhang et al [9]. Complementary dynamics is a novel approach to enrich plastic rigid transformations with elastodynamic secondary effects. In the case off ARAP the implicit momentum, following user interaction, can be interpreted as motion in order to add further displacements to the mesh by simply adding more constraints to the global solving step of the deformed mesh vertex positions.

References

[1] M. Alexa, D. Cohen-Or, and D. Levin. "As-rigid-as-possible shape interpolation." In: *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* (2000), pp. 157–164. DOI: 10.1145/344779.344859.

[2] Omar Cornut. *Dear ImGui*. 2021. URL: https://github.com/ocornut/imgui (visited on 2021-01-27).

[3] Benoît Jacob and Gaël Guennebaud. *Eigen*. 2021. URL: https://eigen.tuxfamily.org/ (visited on 2021-01-27).

[4] Ya Shu Liu, Han Bing Yan, and Ralph R. Martin. "As-rigid-as-possible surface morphing." In: *Journal of Computer Science and Technology* 26.3 (2011), pp. 548–557. ISSN: 10009000. DOI: 10.1007/s11390-011-1154-3.

[5] Alex Rodin. *Matcaps*. 2020. URL: https://github.com/nidorx/matcaps (visited on 2021-01-31).

[6] F James Rohlf. "Rotational fit (Procrustes) methods." In: *Proceedings of the Michigan morphometrics workshop*. Vol. 2. University of Michigan Museum of Zoology Ann Arbor. 1990, pp. 227–236.

[7] Olga Sorkine and Marc Alexa. "As-rigid-as-possible surface modeling." In: *Symposium on Geometry processing*. Vol. 4. 2007, pp. 109–116.

[8] RWTH-Aachen University. *OpenMesh*. 2021. URL: https://www.graphics.rwth-aachen.de/software/openmesh/ (visited on 2021-01-27).

[9] Jiayi Eris Zhang et al. "Complementary dynamics." In: *arXiv* 39.6 (2020). ISSN: 23318422.