# Meshless Deformations Based on Shape Matching

Georg Streich, Alain Schöbi

December 21, 2021

## 1 Introduction

Interested in having a deeper look into different ways of simulating deformations, we found the [MHTG05] paper that approached the problem in a new and exciting way. This approach is not based on physics but rather on shape matching. The algorithm tries to find the best fitting shape in the deformed space and adds forces to the actual shape towards this best fitting shape. This method is not only rather simple to implement but is also unconditionally stable.

## 2 Shape matching

The goal configuration of the vertices should be a configuration where the object has no deformation energy stored. This is always the case when the original object is either translated, rotated or both. Hence, the goal configuration will consist of a translation and rotation of the original undeformed object. We will later add more freedom to the shape matching (c.f. section 5).

## 3 Minimization problem

Given two sets of vertices, $\mathbf{X}_i$ the vertices in the reference space (undeformed) and $\mathbf{x}_i$ the vertices in the world space (deformed), we want to find the best rotation matrix $R$ and the best two translation vectors $\mathbf{C}$ and $\mathbf{c}$ such that we minimize the error between the goal configuration of the vertices $\mathbf{g}_i$ and the actual position of the vertices $\mathbf{x}_i$ weighted by the mass of the vertices $m_i$.

$$\min_{R,C,c} \sum_i m_i ||\mathbf{g_i} - \mathbf{x_i}||^2 \tag{1}$$

The goal position of the vertices are given by a rotation and translation of the the undeformed vertices:

$$\mathbf{g_i} = R(\mathbf{X_i} - \mathbf{C}) + \mathbf{c} \tag{2}$$

$$\min_{R,C,c} \sum_i m_i ||R(\mathbf{X_i} - \mathbf{C}) + \mathbf{c} - \mathbf{x_i}||^2 \tag{3}$$

Thanks to a physical reasoning, it turns out that the best translation vector $\mathbf{C}$ is the center of mass of the undeformed vertices and that $\mathbf{c}$ is the center of mass of the deformed vertices.

$$C = \frac{\sum_i m_i \mathbf{X_i}}{\sum_i m_i} \qquad c = \frac{\sum_i m_i \mathbf{x_i}}{\sum_i m_i} \tag{4}$$

The minimization problem becomes:

$$\min_{R} \sum_i m_i ||R\bar{\mathbf{X}}_\mathbf{i} - \bar{\mathbf{x}}_\mathbf{i}||^2 \tag{5}$$

To find R, we first release the constraint that $R$ has to be a rotation matrix and find the best matrix $A \in R^{3x3}$ that minimizes equation 5. $A$ is given by: (c.f. A.1)

$$A = \sum_i (m_i \bar{\mathbf{x}}_\mathbf{i} \bar{\mathbf{X}}_\mathbf{i}^T) \cdot (m_i \bar{\mathbf{X}}_\mathbf{i} \bar{\mathbf{X}}_\mathbf{i}^T)^{-1} = A_{rot} \cdot A_{sym} \tag{6}$$

where $\bar{\mathbf{X}}_\mathbf{i}$ and $\bar{\mathbf{x}}_\mathbf{i}$ are the coordinates of the vertices with respect to the center of mass (of the corresponding space).

$$\bar{\mathbf{X}}_\mathbf{i} = \mathbf{X_i} - \mathbf{C} \qquad \bar{\mathbf{x}}_\mathbf{i} = \mathbf{x_i} - \mathbf{c} \tag{7}$$

The second part of equation 6, $A_{sym}$ is a symmetric matrix and contains therefore no rotation part. The only rotational part is contained in $A_{rot}$ that can be extracted using a polar decomposition:

$$A_{rot} = R \cdot S \qquad \Rightarrow \qquad R = A_{rot} \cdot S^{-1} \tag{8}$$

where $S \in R^{3x3}$ is a symmetric matrix and $R$ is finally the rotation matrix that minimizes equation 5.

# 4 Integrator

The velocity update is done with an explicit Euler scheme, whereas the position update is done with an implicit Euler scheme. This whole update is called symplectic Euler.

$$\dot{\mathbf{x}}_\mathbf{i}^{\,t+1} \leftarrow \dot{\mathbf{x}}_\mathbf{i}^{\,t} + \frac{1}{\Delta t} f_{def,i} + \frac{1}{m_i} \cdot \Delta t \cdot f_{ext,i} \tag{9}$$

$$\mathbf{x}_\mathbf{i}^{\,t+1} \leftarrow \mathbf{x}_\mathbf{i}^{\,t} + \Delta t \cdot \dot{\mathbf{x}}_\mathbf{i}^{\,t+1} \tag{10}$$

The deformation force $f_{def,i}$ generated at a specific vertex $\mathbf{x}_i$ is a force that pulls the vertex towards its goal position $\mathbf{g}_i$. This force is given by:

$$f_{def,i} = \alpha(\mathbf{g_i} - \mathbf{x_i}) \tag{11}$$

If the vertex $i$ has no initial velocity (i.e. $\dot{\mathbf{x}}_\mathbf{i}^{\,t} = 0$), the parameter $\alpha \in ]0, 1]$ gives the relative distance traveled towards the goal position for one time step as:

$$\dot{\mathbf{x}}_\mathbf{i}^{\,t+1} \leftarrow \frac{1}{\Delta t} f_{def,i} \tag{12}$$

$$\mathbf{x}_\mathbf{i}^{\,t+1} \leftarrow \mathbf{x}_\mathbf{i}^{\,t} + \Delta t \cdot \dot{\mathbf{x}}_\mathbf{i}^{\,t+1} = \mathbf{x}_\mathbf{i}^{\,t} + \alpha(\mathbf{g}_i - \mathbf{x}_\mathbf{i}^{\,t}) = \alpha \cdot \mathbf{g_i} + (1 - \alpha) \cdot \mathbf{x}_\mathbf{i}^{\,t} \tag{13}$$

If $\alpha$ is set to 1, then the simulation deals with a rigid object as the vertices always reach their respective goal position in the next time step.

**Algorithm 1** Global idea of the implementation

Set the $\mathbf{X_i}$, $\mathbf{x_i}$, $\dot{\mathbf{x}}_\mathbf{i}$, $\forall i$
$C \leftarrow \frac{\sum_i m_i \mathbf{X_i}}{\sum_i m_i}$
**for** every time step **do**
  $c \leftarrow \frac{\sum_i m_i \mathbf{x_i}}{\sum_i m_i}$
  $A_{rot} \leftarrow \sum_i (m_i \bar{\mathbf{x}}_\mathbf{i} \bar{\mathbf{X}}_\mathbf{i}^{\,T})$
  $S \leftarrow \sqrt{A_{rot}^{\,T} A_{rot}}$ ▷ Polar decomposition
  $R \leftarrow A_{rot} \cdot S^{-1}$
  **for** every vertex $i$ **do**
    $\mathbf{g_i} \leftarrow R(\mathbf{X_i} - \mathbf{C}) + \mathbf{c}$ ▷ Goal positions computation
    $f_{def,i} \leftarrow \alpha(\mathbf{g_i} - \mathbf{x}_\mathbf{i}^{\,t})$
    $\dot{\mathbf{x}}_\mathbf{i}^{\,t+1} \leftarrow \dot{\mathbf{x}}_\mathbf{i}^{\,t} + \frac{1}{\Delta t} f_{def,i} + \frac{1}{m_i} \cdot \Delta t \cdot f_{ext,i}$ ▷ Symplectic integrator
    $\mathbf{x}_\mathbf{i}^{\,t+1} \leftarrow \mathbf{x}_\mathbf{i}^{\,t} + \Delta t \cdot \dot{\mathbf{x}}_\mathbf{i}^{\,t+1}$
  **end for**
**end for**

# 5 Rotation, linear and quadratic shape matching

We first implemented this method with the goal positions of the vertices computed as explained in section 3. The results do not look very realistic when using big meshes as in a bunny for example. The issue is that when only a couple of vertices are moved from the original position, the best fitting shape will still remain very close to the undeformed original shape, and therefore only the few vertices will oscillate around the goal position which generates weird results (c.f. figure 1).

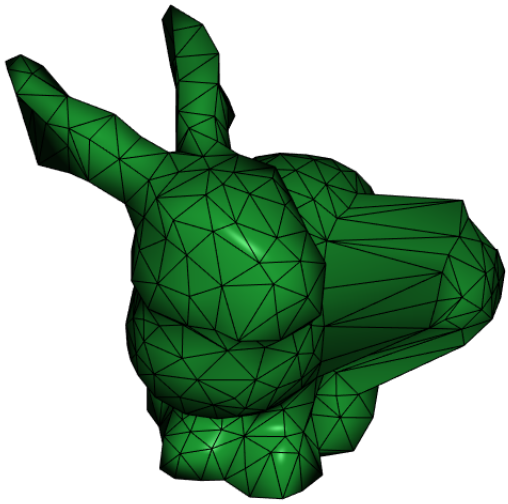

Figure 1: Simulation using only rotation and translation for the shape matching

One option to go around this issue would be to add more freedom to the shape matching with allowing not only rotations and translations but also some linear deformations (c.f. [MHTG05]). The results are already better but still don't look completely realistic (c.f. figure 2).

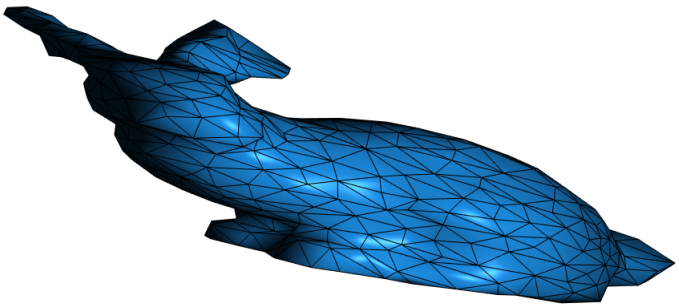

Figure 2: Simulation using linear deformations

In addition to that, we can also allow quadratic deformations which produces more appealing simulations (c.f. figure 3).

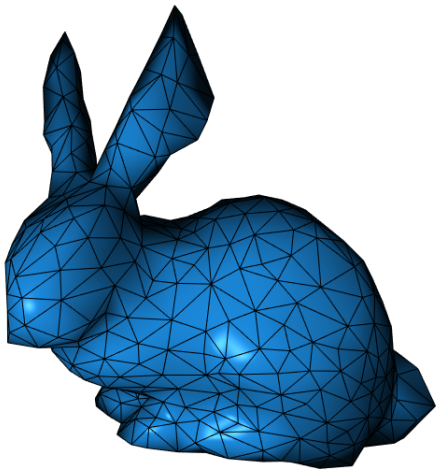

Figure 3: Simulation using quadratic deformations

The next improvement that can be done is to introduce clusters in order to split the vertices into subsets which gives more freedom to the vertices.

## 6 Clusters

To improve the appearance of our results we implement the clustering extension described in [MHTG05]. It extends the algorithm by solving the shape matching problem for multiple subsets and summing up the forces for each of them. Here we note, that for the final result it is quite important as to how the mesh is decomposed into clusters. We experimented with both a fine grained decomposition that uses individual tetrahedrons as clusters and a coarse grained clustering which we generated using a spectral graph clustering algorithm [SM00]. While the fine grained gives the most visually appealing result it loses some of the performance advantages that this method has over other methods that simulate deformable objects. The coarse grained decomposition is slightly less visually appealing but scales better to a larger number of vertices.

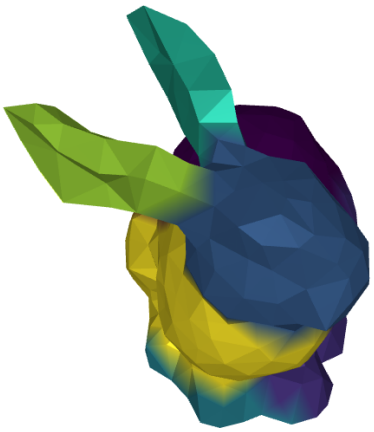

Figure 4: Mesh Decomposition Generated with Spectral Clustering

# A Appendix

## A.1 Solving for matrix A

$$\min_A \sum_i \tfrac{1}{2} m_i \left\| \bar{x}_i' - A\bar{x}_i \right\|^2$$

$$= \min_{A_{ij}} \sum_i \tfrac{1}{2} m_i \left\| \bar{x}_i' - \begin{bmatrix} \bar{X}_{ix} & 0 & 0 & \bar{X}_{iy} & 0 & 0 & \bar{X}_{iz} & 0 & 0 \\ 0 & \bar{X}_{ix} & 0 & 0 & \bar{X}_{iy} & 0 & 0 & \bar{X}_{iz} & 0 \\ 0 & 0 & \bar{X}_{ix} & 0 & 0 & \bar{X}_{iy} & 0 & 0 & \bar{X}_{iz} \end{bmatrix} \begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \\ A_{12} \\ \vdots \\ A_{33} \end{bmatrix} \right\|^2$$

(column wise flattening)

$$= \min_{\mathrm{vec}(A)} \sum_i \tfrac{1}{2} m_i \left\| \bar{x}_i' - (\bar{X}_i^T \otimes I)\,\mathrm{vec}(A) \right\|^2$$

($3\times3$ identity)

$$\frac{\partial(\dots)}{\partial \mathrm{vec}(A)}(\mathrm{vec}(A)) = 0_{1\times 9}$$

$$\sum_i \tfrac{1}{2} m_i \cdot 2 \left\| \bar{x}_i' - (\bar{X}_i^T \otimes I)\,\mathrm{vec}(A) \right\| \cdot \frac{\left(\bar{x}_i' - (\bar{X}_i^T \otimes I)\,\mathrm{vec}(A)\right)^T \cdot \left(-\bar{X}_i^T \otimes I\right)}{\left\| \bar{x}_i' - (\bar{X}_i^T \otimes I)\,\mathrm{vec}(A) \right\|}$$

$$= \sum_i m_i \left(\bar{x}_i' - \underbrace{(\bar{X}_i^T \otimes I)\,\mathrm{vec}(A)}_{A\bar{X}_i}\right)^T \left(-\bar{X}_i^T \otimes I\right)$$

$(1\times 3)\cdot(3\times 9) \overset{!}{=} (1\times 9)$

$$= -\sum_i m_i \left(\bar{x}_i' - A\bar{X}_i\right)^T \left(\bar{X}_i^T \otimes I\right) = 0_{1\times 9}$$

$(1\times 3)\quad (3\times 9)$

$()^T$

$$\sum_i m_i \left(\bar{X}_i^T \otimes I\right)^T \left(\bar{x}_i' - A\bar{X}_i\right) = 0_{9\times 1} \quad \square$$

$$(*):\ (a^T \otimes I)(I \otimes b) = \begin{bmatrix} a_x & 0 & 0 & a_y & 0 & 0 & a_z & 0 & 0 \\ 0 & a_x & 0 & 0 & a_y & 0 & 0 & a_z & 0 \\ 0 & 0 & a_x & 0 & 0 & a_y & 0 & 0 & a_z \end{bmatrix} \begin{bmatrix} b_x & 0 & 0 \\ b_y & 0 & 0 \\ b_z & 0 & 0 \\ 0 & b_x & 0 \\ 0 & b_y & 0 \\ 0 & b_z & 0 \\ 0 & 0 & b_x \\ 0 & 0 & b_y \\ 0 & 0 & b_z \end{bmatrix}$$

$$= \begin{bmatrix} a_x b_x & a_y b_x & a_z b_x \\ a_x b_y & a_y b_y & a_z b_y \\ a_x b_z & a_y b_z & a_z b_z \end{bmatrix} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} = b a^T \quad \square$$

$$\sum_i m_i \begin{bmatrix} \bar{X}_{ix} & 0 & 0 \\ 0 & \bar{X}_{ix} & 0 \\ 0 & 0 & \bar{X}_{ix} \\ \bar{X}_{iy} & 0 & 0 \\ 0 & \bar{X}_{iy} & 0 \\ 0 & 0 & \bar{X}_{iy} \\ \bar{X}_{iz} & 0 & 0 \\ 0 & \bar{X}_{iz} & 0 \\ 0 & 0 & \bar{X}_{iz} \end{bmatrix} \begin{bmatrix} | \\ (\bar{x}_i' - A\bar{X}_i) \\ | \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\sum_i m_i \begin{bmatrix} \bar{X}_{ix} & 0 & 0 & \bar{X}_{iy} & 0 & 0 & \bar{X}_{iz} & 0 & 0 \\ 0 & \bar{X}_{ix} & 0 & 0 & \bar{X}_{iy} & 0 & 0 & \bar{X}_{iz} & 0 \\ 0 & 0 & \bar{X}_{ix} & 0 & 0 & \bar{X}_{iy} & 0 & 0 & \bar{X}_{iz} \end{bmatrix}_{3\times 9} \begin{bmatrix} | & | & | \\ (\bar{x}_i' - A\bar{X}_i) & 0 & 0 \\ | & | & | \\ 0 & (\bar{x}_i' - A\bar{X}_i) & 0 \\ | & | & | \\ 0 & 0 & (\bar{x}_i' - A\bar{X}_i) \\ | & | & | \end{bmatrix}_{9\times 3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{3\times 3}$$

$$\sum_i m_i \left(\bar{X}_i^T \otimes I\right)\left(I \otimes \left(\bar{x}_i' - A\bar{X}_i\right)\right) = 0_{3\times 3}$$

$$\sum_i m_i \left(\bar{X}_i^T \otimes I\right)\left(I \otimes \bar{x}_i'\right) = \sum_i m_i \left(\bar{X}_i^T \otimes I\right)\left(I \otimes \left(A\bar{X}_i\right)\right)$$

(*)

$$\sum_i m_i\, \bar{x}_i' \bar{X}_i^T = \sum_i m_i (A\bar{X}_i) \bar{X}_i^T = A \sum_i m_i\, \bar{X}_i \bar{X}_i^T$$

$$\boxed{A = \sum_i m_i\, \bar{x}_i' \bar{X}_i^T \left(\sum_i m_i\, \bar{X}_i \bar{X}_i^T\right)^{-1}}$$

# References


[MHTG05] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM transactions on graphics (TOG)*, 24(3):471–478, 2005.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.