EUROGRAPHICS Papers (2016), pp. 1–5
B.S.Santos and J.Dischler (Editors)

Cloth Simulation

inspired by Shape Matching

Authors Name

Abstract

In this paper, we exploit a popular technique known as shape matching to create a real-time cloth simulation. We explain the concept and the mathematical details from which we make alterations to maintain a real-time frame-rate. The algorithm works by taking measurements between the current and ideal cloth points. We calculate the transforms necessary to align both shapes. In order to calculate the transforms, we must make assumptions about the cloth structure in relation to each point (e.g., point distribution in relation to the mesh system). We estimate the transforms and iteratively apply corrective forces to ensure the cloth maintains its characteristics while moving in a realistic manner. We target a visually plausible cloth solution for interactive environments by reducing the mathematical complexity of the shape matching problem. Finally, we demonstrate our solution using a simple cloth system with contacts and external interaction.

Keywords: real-time, cloth, physics, animation, simulation, shape matching, stability

1 Introduction

Motivation The booming performance of modern computers allows to run what used to be expensive simulations in real-time. Since the player or the spectator can't feel invisible forces like the wind, or the movement of a character, visual help like clothes is important. Thus the use of real-time cloth simulation in the game industry is increasing - see Assassin's creed Unity, Batman Arkaham Knight, ... -. To create a feeling of immersion, such a simulation must be as physically-plausible as possible. Humans have ease to detect physics incoherence. Therefore, an implausible behavior would create the opposite of the effect we look for: it would break the immersion of the player in the scene. In the same time, real-time applications such as games needs good performances. In this paper, we efficiently implement a model of deformable objects based on shape matching and multi-resolution to simulate cloth behavior in real-time.

Challenging & Limitations Clothes constitute a specific kind of deformable objects: cloth simulation consist in limiting deformations of a bendable solid. In addition to this intrinsic complexity, deformable solids is a continuous problem and thus needs a high level of detail to be realistic which is costly. For all those reasons, cloth simulation is a complex problem.

Contribution In this paper, we exploit the shape matching technique and apply it to simulating a robust real-time cloth effect. We combine a number of techniques (e.g., impulses and penalty forces) to overcome numerical issues within the simulation. We fulfill real-time efficiency expectation through simplifications of the shape matching techniques - sacrificing accuracy and plausibility while maintaining aesthetic qualities and realism. The rest of this paper is structured as follows: first we discuss the related work in cloth simulation in section 2, and then describe the model we choose to use in Section 3. Section 4 describes how we intent to evaluated our system. Section 5 gives the conclusions.

2 Related Work

Deformable solid based on shape matching In the late 80's, [TPBF87] worked on a model for deformable solid based on a shape reference. In this model, a shape reference is keep in memory to be able to compute the deformations suffered by a body. Internal forces are then applied based on those deformations. This model success to simulate deformable behavior of a solid . Later, [TW88] proposes to enhance this model by granting the reference shape the ability to have a rigid-body behavior. That way, the model is now able to simulate deformable and rigid characteristics of a solid. [TF88] extended once again this model to simulate viscoelasticity, plasticity and fracture. In 2005, [MHTG05] proposed to model clothes by applying shape matching on overlapping regions of a cloth. since then, several optimisations and new approaches have made the shape matching technique efficient and versatile.

Our Work ...

3 Overview

Limitations Our work closely follows the work of Bender et al. [BWD13] and Bridson et al. [BFA02] for describing the shape matching technique we use in this paper. The cloth model in combination with the shape matching approach was originally presented by Bender et al. [BWD13]. We combine this with the collision detection and handling technique presented by Bridson et al. [BFA02]. We present a background introduction to the technique, emphasising technical challenges and limitations that are important during implementations. We discuss and explain practical modifications that address numerical and computational bottlenecks (i.e., modifications to the theoretical mathematics and the implementation to ensure the simulation remains stable and runs in real-time). We resolve issues by integrating in work around that produce an aesthetically pleasing solution but at the sacrifice the models accuracy.

3.1 Model

Modeling the state of a deformable solid is a continuous problem but we can simplify it by approximating it as a discrete problem. Thus in a shape matching technique, the state X of a solid at the instant t is described by the positions x_i of $i \in I$ specific points of the solid at that moment (Equation 1).

$$X(t) = (x_0(t), x_1(t), ..., x_i(t)) \tag{1}$$

These specific points create a triangle mesh over the surface of the cloth

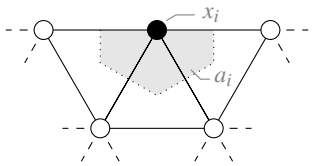

Figure 1: Geometric structure of the triangle mesh

(Figure 1). Each point represents a certain quantity of matter in the fabric. The area $ar(i)$ and mass $m(i)$ represented by a point i can be calculated using (Equation 2).

$$ar_i = \frac{1}{3}\sum_{t\in tri_i} ar_t \qquad m_i = \rho_i\, ar_i \tag{2}$$

where $tri(i)$ is the set of triangles adjacent to the point i, and $\rho(i)$ is the density of $a(i)$. Again, the problem can be simplified by using a constant density for the fabric, denoted ρ. The equation of the mass of a point becomes then:

$$m_i = \rho\, ar_i \tag{3}$$

If we know the neighbours nei_i of a point i, we can find tri_i (Equation 4). In a less formal notation, if two neighbours $\{j,k\}$ are connected then $\{i,j,k\} \in tri_i$.

$$tri_i = \{\{i,j,k\} \in I^3 \mid \{j,k\} \in nei_i^2, k \in nei_j\} \tag{4}$$

3.2 Integration

In this project, we will use the symplectic integration method for its stability. This method approximates Euler's equations and define new equations of dynamics:

$$v_{n+1} = v_n + a_n\Delta t \qquad S_{n+1} = S_n + v_{n+1}\Delta t \tag{5}$$

where v_n is the velocity, a_n the acceleration, and S_n the position of the point at the instant n. Δt is the time-step between the instant n and the instant $n+1$. We can calculate a_n using the second law of Newton (Equation 6).

$$F = m\,a \qquad \Leftrightarrow \qquad a = \frac{F}{m} \tag{6}$$

where F is the sum of all forces applied to a point, and m is its mass.

3.3 Shape Matching

In the shape matching technique, in addition to the current state of a solid X, we keep in memory its initial state $X^{(0)}$. The difference between the current shape of the solid and its initial shape is the deformation D suffered by the solid (Equation 7). To create a deformable solid that converge to its original shape, we just have to counter those deformations.

While shape matching is perfectly adapted to simulate deformable objects - like a bouncing ball for example -, a priori this model has an intrinsic weakness while used to simulate clothes. A cloth is a deformable object that is not supposed to completely go back to its original shape but should instead adapt to the shape of the object that supports it. While we want to limit the stretching and the shearing of a cloth, we want it to be able to bend as much as needed.

$$X = D \cdot X^{(0)} \tag{7}$$

Cloth simulation techniques based on shape matching use it to counter 2D structural deformations of the triangles in the mesh of the fabric, instead of countering the deformations of the cloth itself.

3.3.1 Triangle shape matching

We consider a triangle T with an initial state $T^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$ and a current state $T = (x_1, x_2, x_3)$. $N^{(0)}$ and N are respectively the normals of $T^{(0)}$ and T, and $cm^{(0)}$ and cm are their centers of mass.

Project coordinates To be sure we do not counter the bending of the cloth, when we apply shape matching to a triangle we have to compute the deformations and its countering in the 2D plane of the triangle. To do so, we project the points $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$ in the plane of $T^{(0)}$ and the points (x_1, x_2, x_3) in the plane of T. This can be done by using the projection matrix of each plane calculated with (Eqs. 8 - 9 [BWD13]) where N is the normal of the plane.

$$O_x = \frac{x_2 - x_1}{||x_2 - x_1||} \qquad O_y = \frac{N \times O_x}{||N \times O_x||} \tag{8}$$

$$P = \begin{pmatrix} O_x^T \\ O_y^T \end{pmatrix} \in \mathbb{R}^{2\times 3} \tag{9}$$

We define $\bar{x_i}$ the projection of x_i in the plane of the triangle (Eqs. 10). Because we are comparing two different states of our triangle, we have to use a frame of reference that must remain constant. We chose to express the position of a point in a triangle relative to the center of mass of the triangle.

$$\bar{x_i} = P(x_i - cm_i) \qquad \Leftrightarrow \qquad x_i = P^T\bar{x_i} + cm_i \tag{10}$$

Calculate deformations In the initial work of [BWD13], the deformations suffered by a triangle were expressed as shearing and scaling components of 2D transformation matrices. By subtracting these components, [BWD13] were able to compute a goal position for every points of a triangle. This goal position represents the position of a point relative to the center of mass if the triangle had suffered no deformations. Then, [BWD13] countered the movement of the cloth based on these goal positions. Our approach differs from this initial technique in the fact that we do not use matrix transformations to compute the goal position of a point in its triangle. Instead, we only consider the changes in the distance of each points of the triangle relative to the center of mass.

Counter deformations The goal position $\bar{g_i}$ a point i should occupy in a triangle's plane if we substract structural deformations can be calculated with (Equation 11.1). We use $\bar{g_i}$ to calculate the movement $\Delta\bar{x_i}$ we need to apply to the point i to counter deformations (Equation 11.2). We can vary this countering with a stiff coefficient α. Moreover, as point is contained by several triangles, we need to balance this movement by the number of triangles $|\Re_i|$ that contains the point i.

$$\bar{g_i} = R \cdot \bar{x}_i^{(0)} + T \qquad \Delta\bar{x_i} = \alpha\frac{1}{|\Re_i|}(\bar{g_i} - \bar{x_i}) \tag{11}$$

Since we performed the calculations in the triangle plane, the last thing we need to do is to project our results back into 3D space:

$$\Delta x_i = P^T\,\Delta\bar{x_i} \tag{12}$$

3.3.2 Edge shape matching

An edge shape matching is much simpler. Again, we define the structural deformation of an edge as the drifting of its points from its center of mass cm. Given an edge $[x_1, x_2]$ with initial coordinates $[x_1^{(0)}, x_2^{(0)}]$, we can compute g_1 and g_2 the goal positions of x_1 and x_2 (Equation13).

$$g_1 = cm - q_1^{(0)}\frac{x_2 - x_1}{||x_2 - x_1||} \qquad g_2 = cm + q_2^{(0)}\frac{x_2 - x_1}{||x_2 - x_1||} \tag{13}$$

$$q_i^{(0)} = ||x_i^{(0)} - cm^{(0)}|| \tag{14}$$

We can counter the deformation of an edge using the same principles we used for triangles. We use a stiffness coefficient (β) to control the

deformation, and since a point can by on multiple edges we balance the position change by the number of edges $|\check{\Re}_i|$ containing the point i (15).

$$\Delta x_i = \beta \frac{1}{|\check{\Re}_i|}(g_i - x_i) \tag{15}$$

3.3.3 Update our model using impulses

Finally, we can update the dynamic state of each point of our fabric. To increase the stability of our simulation, we choose to use impulses to counter the deformations.

3.4 Collision detection and handling

Due to rounding errors, direct application of mathematical formulas are likely to fail to detect collisions while performed by a CPU. To counter this, proximity tests are performed, instead of intersection checks. Thus we denote h the threshold under which we consider proximity to become collision. Most of the time, we can take advantage of this lack of precision by using a specific threshold h_t that represents the thickness of a cloth. The rest of the time we will choose h to be a very small number.

3.4.1 Data structure

In order to accelerate the collision detection algorithm, we use an axis aligned bounding box hierarchy. It is created bottom-up once, using the topology of the mesh, and updated at each update.

Hierarchy creation First we compute the alligned bounding box of every triangle, enlarged by the thickness of the cloth. These bounding boxes are the leaves of our hierarchy. We then pair-up adjacent bounding boxes to construct a higher level of our hierarchy. We repeat this process until we have a unique bounding box representing our entire cloth.

Update For each update, recalculate bounding box of each triangle, and regenerate our hierarchy.

3.4.2 Collision detection

When collision is detected with the root, we explore the tree from top to bottom to detect which triangle is currently susceptible to collide. If we end up in a leaf, we do geometry checks to verify a possible collision.

Point to triangle collision Let a triangle $T = (x_1, x_2, x_3)$. To check collision with a point p, we first use the plane equation to determinate if it is close to the plane of the triangle (Equation 16). This equation compute the minimal distance d between a point and a plane. If that distance is less than h_t, we need to check if there is an actual collision with the triangle.

$$d = \frac{(x_2 - x_1) \times (x_3 - x_1)}{||(x_2 - x_1) \times (x_3 - x_1)||} \cdot (p - x_1) \tag{16}$$

$$p' = p - d\frac{(x_2 - x_1) \times (x_3 - x_1)}{||(x_2 - x_1) \times (x_3 - x_1)||} \tag{17}$$

To check if a p point is inside a triangle, we project it onto the plane (Equation 17) and compute the barycentric coordinates $(w_1, w_2, w_3) \in \mathbb{R}^3$ of its projection using (Equation 18 - 19 [BFA02]). If $\forall i \in [1..3], w_i \in [-\delta_i, 1 + \delta_i]$ where $\delta_i = \frac{h_t}{|x_i|}$, p' is in the triangle (x_1, x_2, x_3). δ_i is the conversion of h_t into barycentric coordinate offset.

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} x_{31} \cdot x_{31} & x_{31} \cdot x_{32} \\ x_{31} \cdot x_{32} & x_{32} \cdot x_{32} \end{bmatrix}^{-1} \begin{bmatrix} x_{31} \cdot (p' - x_3) \\ x_{32} \cdot (p' - x_3) \end{bmatrix} \tag{18}$$

$$w_3 = 1 - w_1 - w_2 \qquad x_{ij} = x_j - x_i \tag{19}$$

Edge to edge collision Let two edges $S_{12} = [x_1, x_2]$ and $S_{34} = [x_3, x_4]$. To check if these edges are colliding, we calculate the closest distance between them. To do so, we first check if the segments are colinear using (Equation 20).

$$(x_{12} \times x_{34}) \cdot (x_{12} \times x_{34}) \leq h^2 \qquad \Leftrightarrow \qquad x_{12} = k x_{34} \tag{20}$$

Collinear edges If the segments are colinear, there is two cases (Figure 2). In case (a), the closest distance between S_{12} and S_{34} is the smallest distance between $|x_{13}|$, $|x_{14}|$, $|x_{23}|$, and $|x_{24}|$. In case (b), it is the distance between x_1 and its projection $x_1' = x_3 + (\frac{x_{34}}{|x_{34}|} \cdot x_{31})x_{34}$ on S_{34}.

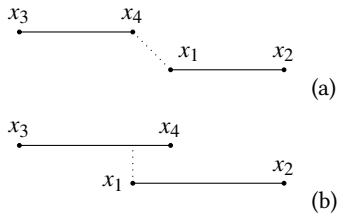

Figure 2: Closest distance between two collinear edges

To check if we are in the case (a) of (b), we simply check if at least one of the projections of x_1 and x_2 is on S_{34} or one of the prohections of x_3 and x_4 is on S_{12} using (Equation 21). If it is, we are in case (b), otherwise we are in case (a).

$$x_i' \in S_{kl} \Leftrightarrow -h \leq \frac{x_{kl}}{|x_{kl}|} \cdot \frac{x_{ki}}{|x_{ki}|} \leq 1 + h \tag{21}$$

Non collinear edges If the segments are not colinear, we have to calculate the closest distance d between the two segments. From now on, we will express a point s on a line L_{ij} using (Equation 22). The same point clamped on the segment S_{ij} will be denoted $\bar{s} = min(1, max(0, s))$.

$$L_{ij}(s) = x_i + s(x_j - x_i) \qquad where \qquad s \in \mathbb{R} \tag{22}$$

To calculate the closest distance between two noncollinear segments S_{12} and S_{34}, we first calculate the closest points $a \in L_{12}$ and $b \in L_{34}$ using (Equation 23 [BFA02]). If $(a, b) \in [-h, 1 + h]^2$, a and b are on the segments S_{12} and S_{34} thus we simply have $d = |L_{34}(b) - L_{12}(a)|$ and there is collision if $d \leq h_t$.

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x_{12} \cdot x_{12} & -x_{12} \cdot x_{34} \\ -x_{12} \cdot x_{12} & x_{34} \cdot x_{34} \end{bmatrix}^{-1} \begin{bmatrix} x_{12} \cdot x_{13} \\ -x_{34} \cdot x_{13} \end{bmatrix} \tag{23}$$

Otherwise, we clamp a and b onto S_{12} and S_{34}. This will give us one of the closest points. Indeed, the clamped point $\bar{s}$ that is the most distant from its original value s is assured to be one of the points we are looking for. To get the other one, we have to project $\bar{s}$ on the other segment using (Equation 24). We denote the projected point s'.

$$p_{ij\ to\ kl}(\bar{s}) = \frac{x_{kl}}{|x_{kl}|} \cdot \frac{S_{ij}(\bar{s}) - x_k}{|S_{ij}(\bar{s}) - x_k|} \tag{24}$$

This projection is made onto the other line, and not the other segment. Therefore s' could be out of its segment. But this would mean that the closest point to $\bar{s}$ is the extremity of the other segment closest to s', that we get when we clamp s'. Since $s' \in [-h, 1 + h] \Leftrightarrow s' = \bar{s}'$, in both case the closest point of $\bar{s}$ is $\bar{s}'$. Thus we finaly have $d = |L_{ij}(\bar{s}) - L_{kl}(\bar{s}')|$ and again there is collision if $d \leq h_t$.

3.4.3 Collision response

The collision response is made of two counter impulses applied to each object colliding. First, we prevent objects interpenetrating any more by

triangles	18	50	162	1152
edges	33	85	261	1776
max	3.00E-3 s	1.87E-3 s	1.28E-2 s	2.84E-2 s
average	5.25E-5 s	1.24E-5 s	8.32E-5 s	2.60E-4 s
min	3.26E-6 s	9.33E-6 s	7.00E-5 s	2.00E-4 s

Table 1: Performances of triangle deformation countering alone (in seconds per update)

countering their relative velocity. In a second time, we will counter the current interpenetration.

$$I_c = \frac{m \cdot v_N}{2} \tag{25}$$

The impulsion I_c used to stop future interpenetration is calculated to cancel entirely the relative movement v_N of our objects on the normal of the contact N (Equation 25). Since I_c is applied to both objects, we divide it by 2. The impulsion I_r used to gradually counter the current interpenetration (Equation 27) is based on the current overlapping o of our two objects (Equation 26.1). I_r is conceptually a simple spring impulse applied to our objects. Therefore, we can controll it using a constant factor k which can be seen a coefficient of stiffness.

$$o = max(h_t - d,\ 0) \qquad I_r^{max} = m(\frac{\delta_t oN}{\Delta t} - v_N) \tag{26}$$

$$I_r = -min(ko\Delta tN,\ I_r^{max}) \tag{27}$$

I_r can also be thought as the countering of the force pressing our objects together. Thus we could use it to calculate the frictions suffered by our objects. The essential reason why the impulse I_r would be caped (I_r^{max}) would be to counter interpenetration gradually so that the force pressing our objects together lasts for several updates. Overwise, frictions would only appear over a single update.

$$(\frac{koN\Delta t}{m} + v_N)\Delta t \leq \delta_t o \tag{28}$$

This means that our objects should be pushed apart by a maximum of $\delta_t o$ where δ_t is an arbitrary coefficient. We can write it in a more formal way as (Equation 28). If we develop it, we end up on (Equation 26.2)

3.4.4 Apply impulses to our model

The collision detection algorithm detect collisions between any point of the cloth and is therefore treating the cloth as some sort of a continuous problem. However, the model we use to represent our cloth is discrete, as we said in section 3.1. Thus we need to convert impulses and velocities generated through our collision system to the discrete model we use.

As well as the position of a point p in a triangle (x_1, x_2, x_3) can be expressed as $w_1x_1 + w_2x_2 + w_3x_3$ where (w_1, w_2, w_3) are the barycentric coordinates of p for that triangle, we can express the velocity v_p of a point p using:

$$v_p = w_1v_1 + w_2v_2 + w_3v_3 \tag{29}$$

where v_i is the velocity of the point x_i. Using the same reasoning, we can spread an impulse I between our points (x_1, x_2, x_3, p) using:

$$\forall i \in [1..3],\ v_i^+ = v_i + w_i\frac{I}{m_i|\Re_i^c|} \qquad v_p^+ = v_p - \frac{I}{m_p|\Re_p^c|} \tag{30}$$

where v_i^+ is the velocity of the point x_i after impulse. Since a point i can suffer multiple collisions, we balance the impulses applied to a point i by $|\Re_i^c|$ the number of collisions this point is implicated in.

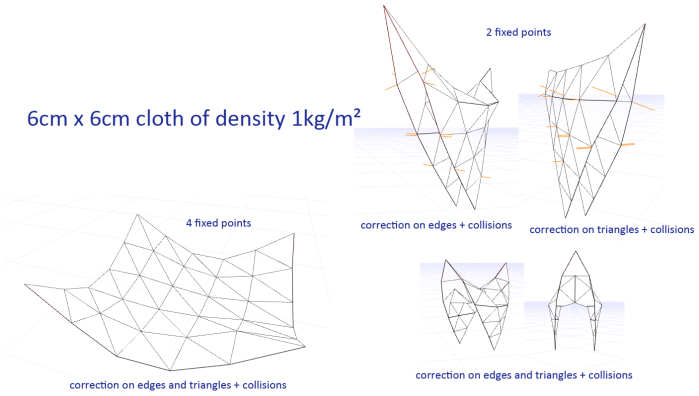

Figure 3: **Screen Captures** - Shows the bending and collisions.

4 Experimental Results

Our system was tested on the following configuration: (i5-5200U @2.20GHz 3MB cache - 4.00GB PC3-12800 @1600MHz - SSD 256GB). Table (1, 2, 3) shows the performances of our implementation of the cloth behavior without collisions and with a fixed physics update timestep of 1.0E-4s. The test scenes performed consisted in a piece of cloth of successively 4x4cm^2, 6x6cm^2, 10x10cm^2 and 25x25cm^2 with a mesh resolution of 1cm, and a density of 1 $kg.m^{-2}$, attached by two corners and subject to gravity (see figure 3). We can rapidly see the limitations of our implementation. Indeed, the processing time of the update function exceed quickly the defined update timestep, preventing the simulation to perform. One of the solutions could be to increase this timestep, but it results in a lose in accuracy and stability for the model.

Regarding plausibility, we can see that our approach provides reasonable results. If the edge corrections are faster than triangle corrections, the visual result in term of plausibility and stiffness is better with triangle corrections, and more specifically when coupled with internal collision detections (see figure 3). When used together, the corrections creates a very specific bending behavior visible in figure 3. This bending behavior varies with the update timestep, and we suspect it is an expression of the instability of our approach.

triangles	18	50	162	1152
edges	33	85	261	1776
max	7.62E-4 s	9.17E-4 s	4.74E-3 s	1.16E-2 s
aver	3.18E-6 s	6.93E-6 s	2.26E-5 s	1.18E-4 s
min	1.87E-6 s	5.13E-6 s	1.40E-5 s	9.84E-5 s

Table 2: Performances of edge deformation countering alone (in seconds per update)

5 Conclusion

To summarize, we have presented the mathematics and the implementation details for a cloth simulation technique using a simplified shape matching technique - we have focused on the practical aspect - regarding stability and mathematical sacrifices - requiring us to adapt the theoretical model to achieve a usable solution. Our solution runs in real-time and remains reasonably stable.

triangles	18	50	162	1152
edges	33	85	261	1776
max	8.05E-4 s	1.23E-3 s	4.34E-3 s	9.82E-3 s
aver	7.77E-6 s	1.93E-5 s	5.44E-5 s	3.40E-4 s
min	5.31E-6 s	1.45E-5 s	4.33E-5 s	3.01E-4 s

Table 3: Performances of triangle and edge deformation countering (in seconds per update)

Future Work Further modification and improvement are possible, such as, inter-body collisions during time-step increments, also the exploitation of parallel hardware data and algorithm optimisations to accelerate system bottlenecks (collision detection [BWD13] and the multi-resolution methods).

References

[BFA02] Bridson R., Fedkiw R., Anderson J.: Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (2002), 594–603. 1, 3

[BWD13] Bender J., Weberb D., Diziolc R.: Fast and stable cloth simulation based on multi-resolution shape matching. *Computers & Graphics 37*, 8 (2013), 945–954. 1, 2, 5

[MHTG05] Müller M., Heidelberger B., Teschner M., Gross M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (2005), 471–478. 1

[TF88] Terzopoulos D., Fleischer K.: Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *ACM SIGGRAPH Computer Graphics 22*, 4 (1988), 269–278. 1

[TPBF87] Terzopoulost D., Platt J., Barr A., Fleischert K.: Elastically deformable model. *ACM SIGGRAPH Computer Graphics 22*, 4 (1987), 205–214. 1

[TW88] Terzopoulos D., Witkin A.: Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications 8*, 6 (1988), 41–51. 1