

# DrivenShape - a data-driven approach for shape deformation

Tae-Yong Kim\*  
Rhythm and Hues Studios

Eugene Vendrovsky†  
Rhythm and Hues Studios.



Figure 1: Theodore’s shirt animated with DrivenShape.

*DrivenShape* is a technique developed at Rhythm and Hues Studios, aimed at providing an efficient alternative to expensive deformation computations (e.g. cloth simulations). It allows users to drive deformation of one object from the shape of another object. For example, we may wish to drive shape (wrinkles and folding) of a pants based on how the character’s legs are posed.

The tool is especially useful when the paired shapes are highly correlated and the space of all the possible shapes is limited. Such cases are common in animation, such as deformation of tight fitting clothing or muscle flexing, where deformation is almost determined by the pose of the character. (Try raising your arm many times, and observe how your shirts responds to the pose).

## Overview

We start by analyzing reference animations that contain both pose shape (driver) and its corresponding deformed (driven) shape. For a database of  $k$  frames, let us call the pose shape  $P_i$ , and the driven shape  $D_i$ , for  $1 \leq i \leq k$ .

At runtime, we have a new pose  $P_{new}$  and wish to reconstruct corresponding deformation  $D_{new}$ . We express the new target shape  $D_{new}$  as weighted sum of  $D_i$  in the database,  $D_{new} = \sum w_i * D_i$ , and use  $P_{new}$  to compute weights such that  $P_{new} = \sum w_i * P_i$ . In a mathematical form, we wish to minimize  $\| A * w - b \|^2$ , where  $A$ ’s column vectors are filled with vertices of  $P_i$ ,  $w$  is the weights, and  $b$  is a column vector with vertices of  $P_{new}$ .

We seek for  $w_i$  that minimizes the above norm, resulting in the following equations.

$$G = A^T * A \quad (1)$$

$$g^T = -2 * A^T * b \quad (2)$$

$$b0 = b^T * b \quad (3)$$

$$\| A * w - b \|^2 = w^T * G * w + g^T * w + b0 \quad (4)$$

, subject to

$$\sum_{i=1}^k w_i = 1 \quad (5)$$

$$0 \leq w_i \leq 1 \quad (6)$$

\*e-mail: tae@rhythm.com

†e-mail: eugene@rhythm.com

, which can be solved with the standard quadratic programming (QP) techniques [Goldfarb and Idnani 1983]. Equation 6 ensures that we don’t get artifacts from negative weights.

## Database Construction

We provide a tool to automatically extract  $N$  most distinct shapes based on a greedy process. We expand the set of pose shapes by adding at each step a shape that’s the most different (by Euclidean measure) from all the shapes contained within the set. After  $N$  iterations, we have a set of  $N$  shapes that are sufficiently different from each other. We also allow users to directly pick the pair of shapes they want to add. Usually, users start with small number of automated picks and add to the database as they wish.

## Collision Detection

For highly deformable characters, a simple linear blend would not match the target shape (in general,  $P_{new} \neq \sum w_i * P_i$ ). One noticeable artifact is a deformation that intersects with the pose shape. Let  $\tilde{P}_{new} = \sum w_i * P_i$ , and  $\tilde{D}_{new} = \sum w_i * D_i$ . After weights are computed, we apply additional mapping  $\tilde{P}_{new} \rightarrow P_{new}$  to  $\tilde{D}_{new}$  as follows. For each point  $d$  in  $\tilde{D}_{new}$ , we find the closest triangle from  $\tilde{P}_{new}$  and construct a coordinate system on the closest point with one axis being the normal, and another axis being one edge of the triangle. Local coordinate of  $d^{local}$  is then computed, and used to reconstruct final position  $d^{final}$  after we move points of the triangle to  $P_{new}$  and update the coordinate system.

When two geometries are close (e.g. legs crossing each other), the closest triangle can come from wrong side and cause popping artifact. We let users supply additional mapping to exclude unnecessary binding (e.g. left pants maps to left leg only).

## Secondary Motion and Layering

Since output is directly mapped from the pose, we lose secondary motion that was contained in the original animation. When secondary motions are desired, we turn back to regular simulations, but users still use *DrivenShape* to guide the simulation (e.g using spring constraints) or to rapidly provide the initial draping of cloth. It also seamlessly works with additional deformations such as noise-based wind effects. Often users partition the geometry and apply *DrivenShape* to more rigid part, and simulate more dynamic part (such as hood of the sweater) with cloth simulator.

## Conclusion

For 20-30 shape pairs with 5000 vertices, the system runs in real-time, providing rapid feedback for animators. This technique was extensively used in our recent production of *Alvin and the Chipmunks*, and we could eliminate the need for expensive cloth simulations for about 70 percent of the shots. Although it was originally developed to speed up the cloth simulation pipeline, users have expanded its use to anything that deforms such as muscles, facial structure, etc.

## References

GOLDFARB, D., AND IDNANI, A. 1983. A numerically stable dual method for solving strictly quadratic programs. *Mathematical Programming* 27, 1–33.