Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2008)
M. Gross and D. James (Editors)

DrivenShape - a data-driven approach for shape deformation

Tae-Yong Kim† and Eugene Vendrovsky‡

Rhythm and Hues Studios, USA

Abstract

DrivenShape is a data-driven technique that exploits known correspondence between two sets of shape deformations (e.g. a character's pose and her shirt). It allows users to drive deformation of secondary object simply by animating the pose shape. The tool is especially useful when the corresponding shapes are highly correlated and the space of all the possible shapes is limited. We have successfully used this technique in our recent productions, and it enabled artists to save on both computation time and man hours.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

DrivenShape is a technique that exploits known correspondences between two sets of shape deformations. It aims at providing an efficient alternative to expensive deformation computations (e.g. cloth simulations) by reusing precomputed deformations. For example, one may notice that in tightly fitted clothings, formation of wrinkles and foldings are highly correlated with the underlying pose (try raising your arm may times, and observe how your shirts respond to the pose in very similar ways).

This correlation between the pose and secondary objects allows us to develop an efficient algorithm to reuse precomputed data over and over for new poses. The tool is especially useful when the shape pairs 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.

We formulate the problem as that of interpolation. If the new pose can be interpolated from a set of *key poses*, then we can use the same interpolation weights to interpolate the target (driven) shapes. In this paper, we show how one can reuse precomputed shape data once we create appropriate databases.

For all of our examples, our system runs in real-time, providing rapid feedback to the animator. This allows users to indirectly control driven shape (e.g. cloth) while using animation method of their choice for the pose shape (e.g. character's body) deformation. This way, computationally expensive deformation can be intuitively controlled as if they were simple deformation tools.

We have used the technique in our recent productions, and this allowed artists to save both on computer time and man hours. In many shots, shirts and pants of animated characters were entirely animated with the presented technique.

1.1. Related Work

Data-driven approaches have been actively explored in recent research work. Examples include a method to precompute dynamic scenes and reconstruct simulations among cyclic events via model reduction [JF03], and a method to ease facial animation task by constructing plausible sequences of facial expression with a model learned from motion capture database [LCXS07], and a method to stitch together and reuse related mesh deformation sequences [JTCW07], reusing existing deformations on one shape to other shapes [SP04] and many algorithms to reuse motion capture data (e.g. [KGP02]).

† tae@rhythm.com
‡ eugene@rhythm.com

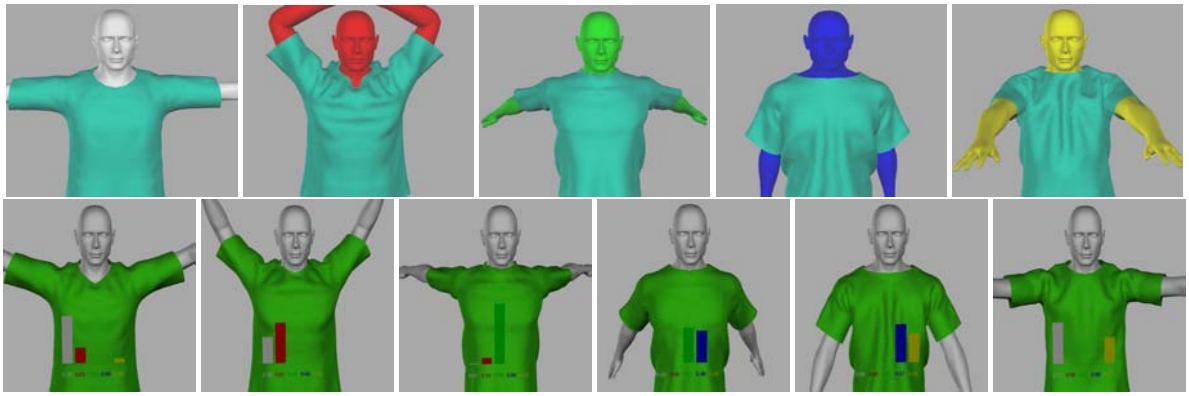

Figure 1: *Top) we build a database of 5 key shapes, marking correspondence between the shirt(driven shape) and the character's pose. Bottom) When a new (arbitrary) pose shape is given , we compute weights from the key pose shapes, and reconstruct cloth shape as weighted combination of the key cloth shapes. The graphs illustrate weights from key poses of matching colors.*

Conceptually similar to ours, [BLCD02] extract blending weights from existing cartoon animations through an optimization process on affine transformations, and then use those weights to generate new animation sequences. We follow similar procedures (we apply the weights computed from pose shapes to deform secondary objects), but in a different context of shape deformations.

Algorithms exist that exploit user supplied data to assist shape deformations. Among others, the Pose Space Deformation (PSD) technique [LCF00] explored ways of using user-supplied example skins while augmenting skeletal-driven deformation. The MeshIK system [SZGP05] support mesh deformations (without skeleton) via user-specified constraints, through non-linear optimizations on meshes.

Weber et al. presents a system similar to ours that use example shapes in the context of joint-based shape deformations [WSLG07]. Although promising, the method requires specialized skeleton setups for shape deformations. In production environments, deformations do not always use skeletons, and are often combined with other methods even when they do. We do not need any skeletons for the pose shape.

It is illuminating to note that while the pose shape can be deformed by any existing technique mentioned above, our technique can provide a back-end to deform the secondary (driven) shapes (e.g. complex wrinkle formation of clothing) that users may not wish to manually animate.

Secondary objects (e.g. clothings and skins) are often computed by expensive simulations since underlying deformation models are highly non-linear and complex [BW98] [BMF03]. Alternative methods include techniques to construct cloth/skin's wrinkles directly by mapping wrinkle curves on the deformed surface [CGW*07] [LC04]. One can also directly capture complex motion of cloth deformation [WCF07]. Such data can be also edited through a multi-resolution mesh editing framework [KG06].

We envision that our technique can provide a useful abstraction of data acquired and edited from such sources. For example, if character's pose was captured together with the cloth deformation, motion captured data can be re-processed in our framework and can provide an intuitive animation framework to the animators. We do not the deformation data to be subject to real physics. Any artistically carved deformations can be equally abstracted via our system as well. In practice, we have used both approaches to construct shape databases.

2. Algorithm

We start by anaylizing reference animations that contain both pose shape (driver) and its corresponding deformed (driven) shape. These can come from existing simulations or keyframed animations. We only require that correspondences between two shapes are known. From these correspondences, we extract N most distinct pairs of key shapes and store them as a database. Once the database is constructed , the pose shape is the only input to the system. Our system then reconstructs corresponding deformation of the driven shape.

2.1. Database Construction

Given reference animations, we extract N key pairs that would sufficiently cover all the possible shape variations. Often these shapes are at the extrema of underlying deformations. For example, if chracter's arm is bending along one axis, the shape at the beginning and the shape at the end of bending will provide reasonable key shapes as we can interpolate all the in-between shapes as weighted combination of the two shapes.

Key shapes can be extracted automatically or hand picked. We provide a tool to automatically extract N most distinct pairs of shapes based on a greedy process. We first pick any (usually the rest) shape and add that to the set. We then expand the set of pose shapes by adding at each step a shape that's the most different from all the shapes contained within the set. After N iterations, we have a set of N shapes that are sufficently different from each other. The differences in shapes are measured in a simple euclidean measure - $\sum \parallel P_i - Q_i \parallel^2$, where P_i and Q_i are corresponding vertex positions of the two shapes to compare.

We also allow users to directly pick the pair of shapes they want to add. Usually, users start with small number of automated picks (5-10) and add to the database as they wish. These additions are mostly out of artistic need. For example, similar poses can often result in significantly different shapes for the driven shape (e.g. due to non-linearity in simulations). In that case, decision on whether to add these *features* becomes totally an artistic call. In practice, we found about 10-20 shapes are usually sufficient to cover entire space of pose deformation without generating too much visual quality degradation.

Once key shapes are extracted, these are stored as a database. At this point, reference animations are no longer needed. Note that one can create as many databases as needed. In our experience, users have created dozens of databases, varying simulation parameters such as friction on collision object, stiffness of cloth, etc.

2.2. Weight Computation

For a database of N shape pairs, let us call the pose shape P_i, and the driven shape D_i, for $1 \leq i \leq N$. We find the weights from the given target pose shape P_{target} such that blending the pose shapes P_i with these weights would result in a shape as close as possible to P_{target}. Then, using these weights, we compute the desired driven shape D_{target} as a weighted combination of driven shapes D_i from the database. See Figure 1.

In a mathematical form, we wish to minimize $\parallel A*w-b \parallel^2$, 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_{target}.

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

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

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

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

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

, subject to

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

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

.

Equation 6 ensures that we avoid artifacts from negative weights. We solve the above equations using the standard quadratic programming (QP) techniques [GI83]. Once weights are found, one can plug these values to create the new deformation $D_{target} = \sum w_i * D_i$.

2.3. Correction of Residual Displacement

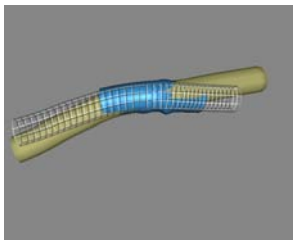

Figure 2: *The $\widetilde{D}_{target} = \sum w_i * D_i$ (blue) matches the blended pose $\widetilde{P}_{target}$ (wire), but penetrates the given pose target P_{target} (yellow)*

The basic algorithm outlined above works well for linear interpolation problems. However, for highly deformable characters, a simple linear blend would not match the target shape. Let $\widetilde{P}_{target} = \sum w_i * P_i$. In general, $P_{target} \neq \widetilde{P}_{target}$.

Note that when the targeted pose shape is outside the space of shapes spanned by original pose shapes, we get projection of weights onto the convex space of the shapes, due to the non-negativeness constraint we impose in 6. In other words, we do not extrapolate the shape - we rather clamp when the input is out of bounds. This has a desirable effect of keeping interpolated shapes always plausible, but the blended shape D_{target} may not match P_{target} well. Noticeably, this introduces an artifact. For example, driven shape could penetrate the pose shape (Figure 2).

For improved accuracy, we use the residual displacement from the blended pose shape $\widetilde{P}_{target}$ to the target pose shape P_{target}. Note that if we apply the same weights to the driven shape ($\widetilde{D}_{target} = \sum w_i * D_i$), the resulting blended (driven) shape $\widetilde{D}_{target}$ will be roughly in line with $\widetilde{P}_{target}$ as long as the two deformations are correlated, since our weights are always interpolative. So, by mapping a residual from $\widetilde{P}_{target}$ to P_{target} to $\widetilde{D}_{target}$, we can get a new deformation for the driven shape D_{target} that is close to the target pose P_{target}.

The mapping function $F(\widetilde{P}_{target} \rightarrow P_{target})$ can be defined in many ways. Requirements to ensure smooth transfer of deformation include 1) the funciton be continuous inside and outside the pose shape within reasonable distance 2) the function be smooth. The problem is well addressed in previous work such as Harmonic Coordinates [JMD*07],

Mean Value Coordinates [JSW05] and Deformation Transfer [SP04].

Although these existing techniques may produce superior results, they require substantial setup time before deformation transfer can be computed. In our case, the deformation mapping should be updated on the fly (since the *rest* configuration is $\widetilde{P}_{target}$ that changes every frame). For our purpose, significant degradation in performance is less favored since it defeats our purpose - providing fast alternative to simulations.

Noting that the residual artifacts diminish as we add more examples to the database, we settle for a simple proximity based deformation transfer algorithm. The presented technique works well for reasonable amount of deformation transfer. For limitations and more discussions, see Figure 8 and section 4.1.

2.4. Proximity-based Deformation Transfer (Figure 3)

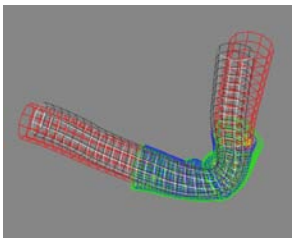

Figure 3: *For each point of $\widetilde{D}_{target}$ (blue), we find the closest point (yellow line) on $\widetilde{P}_{target}$ (gray). Then we apply displacement from $\widetilde{P}_{target}$ (gray) to P_{target} (red), to get the final shape D_{target} (green).*

For each point d_i in $\widetilde{D}_{target}$, we find the closest triangle from $\widetilde{P}_{target}$ and construct a coordinate system centered at the closest point cp_i, with y-axis being the triangle normal, and another axis being one edge of the triangle. Local coordinate d_i^{local} is then computed by projecting d_i on the coordinate system.

Given a barycentric coordinate of cp_i, we can reconstruct the point on P_{target}, and similary the triangle normal and coordinate system. We use d_i^{local} to reconstruct the d_i^{target} on the new coordinate system. The most time consuming step is the closest point computation, and we have optimized it with spatial acceleration structure (e.g. k-d tree) on the pose shape. The whole process then runs in real-time for shapes with over 5000 vertices.

2.5. Penetration Handling

Although $\widetilde{D}_{target}$ is usually consistent with $\widetilde{P}_{target}$, small penetrations do occur since deformation models of $\widetilde{D}_{target}$ and $\widetilde{P}_{target}$ are not completely correlated. Our proximity based transfer has an added bonus for this situation, since collision detection can be easily done by checking if y (normal) value of d_i^{local} is negative. If so, we simply set it to zero (or any added offset value) before we reconstruct d_i^{final} to remove the penetration.

2.6. Avoiding Cross-Talk in the Transfer

When the pose shape has two sections that are orignally apart and get close to each other (e.g. legs crossing each other), the deformation transfer can come from the wrong side and points in the driven shapes can incorrectly snap to the wrong area in the pose shape (Figure 4). These are solved by supplying additional mapping to exclude such unnecssary binding. Users controls which area of driven shape maps to which area of the pose shape by assigning groups or painting (e.g. left pants should only map to left leg).

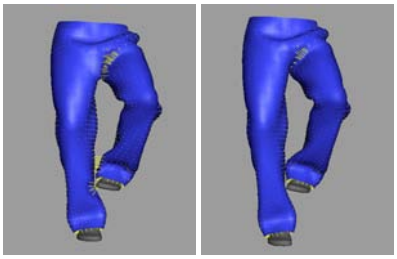

Figure 4: *Left: when right foot is close to the left leg, some points on the pants are incorrectly transferred by left leg. Right: by assigning additional mapping, the points now find the closest points only on the right leg, eliminating the artifacts.*

2.7. Normalizing Animation Data in a Moving Frame

In a practical environment, animation data almost always have global transformations in it. Normalizing such transformations yields the best results for the *DrivenShape* (Figure 5). When we construct the database, we find such a transformation matrix T_i for each pose P_i either from animation data or orientation fitting(e.g. [HS88]). We apply its inverse T_i^{-1} to all the vertex data. When a new target pose P_{target} associated with T_{target} comes in, we first normalize the pose by T_{target}^{-1}, and use it to find weights. The driven shape target D_{target} is constructed in this normalized frame, and the final result is transformed back by T_{target}.

3. Examples

For validation purposes, we constructed a database of 6 key shapes for the cylinder example (Figure 6), and reconstructed the cloth shape using the pose that was contained in the original animation/simulation (see accompanied video as well). Although results do not exactly match, the reconstructed shapes are visually plausible. In this example,

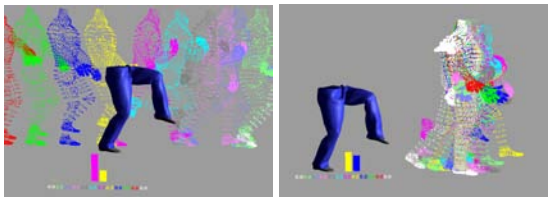

Figure 5: *Left) key shapes are extracted from animations in a moving frame. Note that weight computation is dominated by the translation. Right) after nomaliztion, we find a better pair of shapes to interpolate, yielding more realistic results.*

DrivenShape runs at 40 fps, whereas orignal simulation took about 2 secs per frame on a single CPU AMD 1.9Ghz machine. Cloth mesh had about 5000 vertices.

Figure 7 shows snapshots of shapes at a number of poses and show how adding more examples improve the reconstructed results. This example uses the normalization steps described in section 2.7.

Figure 8 shows what happens when there is not enough pose in the database. Upon small deviation off the pose space, the system reasonably constructs the cloth shape by mapping the deformation from the closest pose to the target pose via the deformation transfer. When the pose deviation is too extreme, the deformation transfer fails and indicates that we need more examples there.

Finally, Figure 9 and accompanied video show production examples that used *DrivenShape*. In these productions, we could eliminate the need for cloth simulations for about 70 percent of the shots and these shots were completely animated with *DrivenShape*.

4. Limitations and Discussions

4.1. Artifacts of Linear Blending

It is known that linear blending of shapes are not best suited for rotational deformation such as in articulated characters [WSLG07], and thus generate artifacs such as shrinking shapes when blending shapes of different orientations.

As databases become reasonably dense (10-20 examples), artifacts diminish (since rotational deformations can be approximated by piecewise linear deformation), so this has not been a major issue in our production environments. Nevertheless, we anticipate that there may be better approaches.

We tried computing the delta shapes (local cooridnates) of the driven shape w.r.t the pose shape, and blend the deltas to reconstruct the final driven shape. This approach failed since a single delta becomes ambiguous as most key pose shapes are at the extrema of deformation space. Globally smooth deformation transfer [SP04] may help in this regard. We could transfer deformation of driven shape from each key pose to the target pose and then perform blending in the more localized space. Note that this is essentially an inverse deformation transfer problem (instead of going from rest to deformed shape, we need to go the other way). A better shape interpolation scheme such as PSD [LCF00] can also help in getting better intermediate shapes.

Another approach is to gain knowledge in the non-linear deformation model of pose shape, either by construction [WSLG07] or by optimization [BLCD02]. However, we are not yet aware of appropriate fitting technique that can be used to fit interpolation weights of non-linear deformations models to arbitrary shapes in an efficient (hopefully real-time) way.

4.2. Secondary Dynamics

DrivenShape does not try to recover any secondary dynamics. This is often not an issue for tightly fitted clothings, but when secondary motions are desired, we lose substantial amount of realism.

In our experiences, users turn to regular simulations when secondary motions are desired. Even in this case, *DrivenShape* rapidly provides the initial draping of cloth - a process that often consumes significant amount of simulation time by itself. Often users partition the geometry and apply *DrivenShape* to stiffer parts, and simulate more dynamic part (such as the hood of a sweater) with regular cloth simulations.

In other instances, users would animate clothing with *DrivenShape* as a rapid first pass, and use the result to drive actual cloth simulations (e.g. with spring constraints, etc.). This saved man hours spent on tweaking cloth parameters needed to get the right look. Hence, we believe that *DrivenShape* can be a useful tool to jump start cloth control techniques such as [BMWG07].

5. Conclusion and Future Work

The proposed framework works the best when we have a few characters with clothings, and those characters appear in many (in our cases, over 200 shots) animations. This greatly justifies the time spent on creating the database since new animations can be interactively created without the need for expensive simulations.

One useful extension will be to apply the technique to different characters with different proportions. These expanded set of databases could then prove useful for massive simulation of crowd characters with clothing, or interactive applications such as games.

One may be able to extend the shape matching to the problem of motion matching. In this case, the notion of pose would include motion history (N frames of pose deformation would make one *pose* in the database) as well as history of deformed geometry. Then, we could apply similar

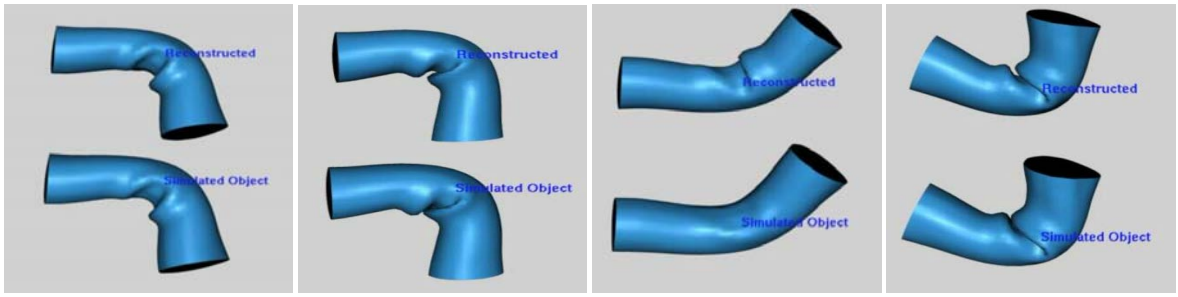

Figure 6: *We compare reconstructed results to original simulations. Top row) reconstructed cloth by* DrivenShape. *Bottom row) original simulations. Note that reconstructed shape has temporally smoother feature than orignal simulations. This is the most evident in the third figure from the left, where reconstructed wrinkles appear earlier than in the orignal simulation.*

techniques to compute weights. This will heavily expand the size of database, so efficient data reduction/pruning technique may be necessary.

References

[BLCD02] BREGLER C., LOEB L., CHUANG E., DESHPANDE H.: Turning to the masters: Motion capturing cartoons. *ACM Transactions of Graphics 21*, 3 (2002), 399–407.

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. *ACM Symposium on Animation* (2003).

[BMWG07] BERGOU M., MATHUR S., WARDETZKY M., GRINSPUN E.: Tracks: Toward directable thin shells. *ACM Transaction on Graphics 26*, 3 (2007).

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. *ACM SIGGRAPH Proceedings* (1998), 43–54.

[CGW*07] CUTLER L., GERSHBIEN R., WANG X. C., CURTIS C., MAIGRET E., PRASSO L., FARSON P.: An art-directed wrinkle system for cg character clothing and skin. *Graphical Models 69*, 5-6 (2007), 219–230.

[GI83] GOLDFARB D., IDNANI A.: A numerically stable dual method for solving strictly quadratic programs. *Mathematical Programming 27* (1983), 1–33.

[HS88] HIGHAM N. J., SCHREIBER R. S.: Fast polar decomposition of an arbitrary matrix. *Technical Report TR88-942* (1988).

[JF03] JAMES D., FATAHALIAN K.: Precomputing interactive dynamic deformable scenes. *ACM Transaction on Graphics 22*, 3 (2003).

[JMD*07] JOSHI P., MEYER M., DEROSE T., GREEN B., SANOCKI T.: Harmonic coordinates for character articulation. *ACM Transactions on Graphics 26*, 3 (Aug. 2007).

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics 24*, 3 (2005), 561–566.

[JTCW07] JAMES D., TWIGG C., COVE A., WANG R.: Mesh ensemble motion graphs: Data-driven mesh animation with constraints. *ACM Transaction on Graphics 26*, 4 (2007).

[KG06] KIRCHER S., GARLAND M.: Editing arbitrarily deforming surface animations. *ACM Transaction on Graphics 25*, 3 (2006), 1098–1107.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Transaction on Graphics 21*, 3 (2002).

[LC04] LARBOULETTE C., CANI M.-P.: Real-time dynamic wrinkles. *Computer Graphics International* (2004).

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. *ACM SIGGRAPH Proceedings* (2000), 165–172.

[LCXS07] LAU M., CHAI J., XU Y.-Q., SHUM H.-Y.: Face poser : interactive modeling of 3d facial expressions using model priors. *Symposium on Computer Animation* (2007), 161–170.

[SP04] SUMNER R., POPOVIC J.: Deformation transfer for triangle meshes. *ACM Transaction on Graphics 23*, 3 (2004), 399–405.

[SZGP05] SUMNER R., ZWICKER M., GOTSMAN C., POPOVIC J.: Mesh-based inverse kinematics. *ACM Transaction on Graphics 24*, 3 (2005).

[WCF07] WHITE R., CRANE K., FORSYTH D. A.: Capturing and animating occluded cloth. *ACM Transaction on Graphics 26*, 3 (2007), 1098–1107.

[WSLG07] WEBER O., SORKINE O., LIPMAN Y., GOTSMAN C.: Context-aware skeletal shape deformation. *Eurographics 26*, 3 (2007).

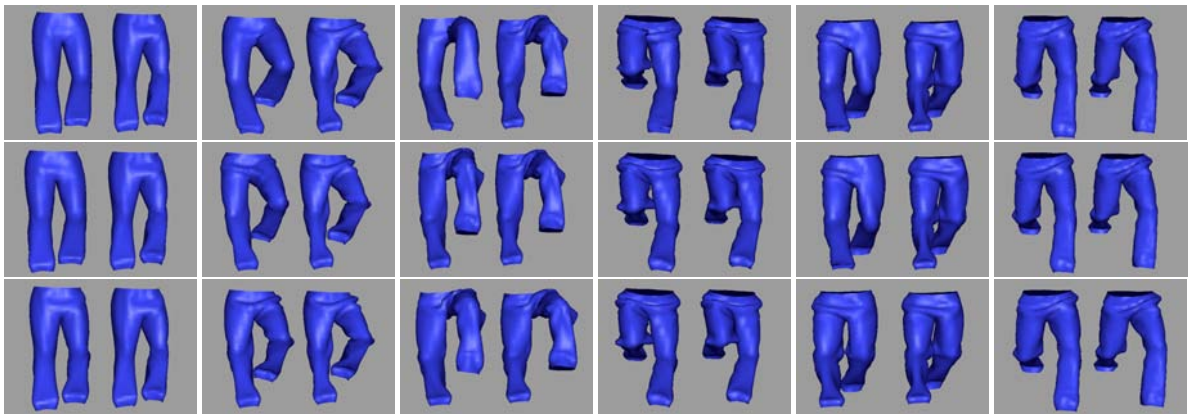

Figure 7: *Comparison on number of key shapes in the database. For each shape pairs, left side is the reconstructed shape and the right side is original simulations. Top row) N = 4 Middle Row) N = 6 Bottom Row) N = 12.*

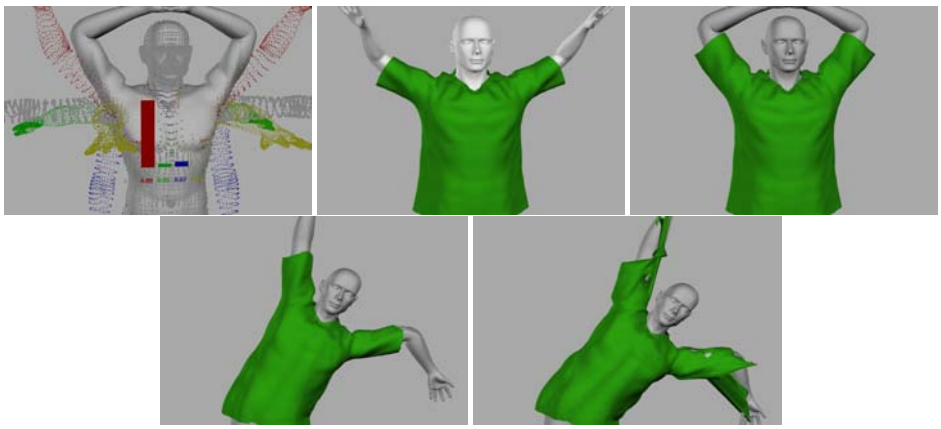

Figure 8: *(Top : left) The given pose (gray) is outside the space spanned by known poses. We find the closest pose (red), and Middle) blend cloth shape on the pose. Right) We transfer the blended shape to the target shape. (Bottom : left) the pose is farther away from the pose space, and* DrivenShape *still copes with the pressure. Right) oops.* DrivenShape *finally gives up!*

Figure 9: *Our recent productions used* DrivenShape *extensively. No simulations were done except for database creations, and these shots were finalized without additional simulations. Image courtesy of 20th Century Fox and Rhythm and Hues.*