Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Cartoon Textures

Christina de Juan[†] and Bobby Bodenheimer[‡]

Vanderbilt University

Abstract

In this paper we present a method for creating novel animations from a library of existing two-dimensional cartoon data. Drawing inspiration from the idea of video textures, sequences of similar-looking cartoon data are combined into a user-directed sequence. Starting with a small amount of cartoon data, we employ a method of nonlinear dimensionality reduction to discover a lower-dimensional structure of the data. The user selects a start and end frame and the system traverses this lower-dimensional manifold to re-sequence the data into a new animation. The system can automatically detect when a new sequence has visual discontinuities and may require additional source material.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

The process of traditional cel animation has seen a number of enhancements in recent years, but these have focused on such tasks as texture mapping the cels [CJTF98], creating shadows [PFWF00], or retargeting the motion of one character onto another character [BLCD02]. However, cel animation remains a very tedious and time-consuming task, requiring twenty-four hand drawn frames per second of animation. For a typical animated TV series, artists bring life to familiar cartoon characters for every episode, yet no method exists that would allow them to reuse their drawings for future episodes.

Software packages such as Toon Boom Technologies, [FBC*95], can create simple inbetweens based on vector animation. Although an animator could reuse the original models of the characters, the basic animation still has to be created, and these animations tend to lack the expressiveness of familiar styles, such as the distinctive style of animations by Chuck Jones. The same issues arise when creating 3D models for cartoon characters and 'toon-rendering them. 'Toon-rendering is a technique that can render 3D scenes in styles that have the look of a traditionally animated film; it is often called 'toon shading. Adding a great deal of deformation, like squash and stretch or incorporating other principles of animation [Bla94] to a 3D model of a character is often challenging, requiring the skills of a talented artist. Even when 'toon-rendering a 3D character, one cannot expect it to look like the traditionally hand drawn Wile E. Coyote getting flattened or stretched in a visually extreme manner.

This work presents a method for creating novel animations from a library of existing cartoon data. Drawing inspiration from the idea of video textures [SSSE00], sequences of similar-looking cartoon data are combined into a user-directed sequence. Our goal is re-sequencing cartoon data to create new motion from the original data that retains the same characteristics and exposes similar or new behaviors.

The number of new behaviors that can be re-sequenced is restricted by the amount of data in our library for each character. Starting with a small amount of cartoon data, we use an unsupervised learning method for nonlinear dimensionality reduction to discover a lower-dimensional structure of the data. The user selects a desired start and end frame and the system traverses this lower-dimensional manifold to re-sequence the data into a new animation. Our method is model-free, i.e., no a priori knowledge of the drawing or character is required. The user does not need the ability to animate, or know what an acceptable inbetween is, since the data is already provided. The system can detect when a transition is abrupt, allowing the user to inspect the new animation and determine if any additional source material is needed. Minimal user input is required to generate new an-

† email:cdejuan@vuse.vanderbilt.edu
‡ email:bobbyb@vuse.vanderbilt.edu

imations, and the system requires much less data than the video textures method for re-sequencing.

2. Previous Work

2.1. Animation-Based Methods

The issue of generating inbetweens for cartoon animation has been studied. Reeves [Ree81] presented a method for creating inbetweens by using moving-point constraints. A moving-point is a curve in space and time that provides a constraint on the path and speed of a specific point on the keyframe for a character. These moving-points are manually specified, and allow for multiple paths and speeds of interpolation. While this method provides control in creating a new animated sequence by generating the inbetweens "automatically," a great deal of manual effort is involved.

Sederberg and Greenwood [SG92] studied how to smoothly blend between a pair of 2-dimensional polygonal shapes. By modelling a pair of contours as thin wires, [SG92] minimize equations of work for deforming thin wires to achieve smooth shape transformation between the two contours. They address the problem of vertex correspondences by specifying a small number of initial corresponding point pairs on the input contours. While their results show nice shape blending, the shapes must be polygonal, therefore using existing animations would require polygonalizing every image. Their results also depend on the initial manual placement of the corresponding vertex pairs.

Bregler et al. [BLCD02] reused cartoon motion data by capturing the motion of one character and retargeting it onto a new cartoon character. This approach does not generate a new cartoon motion. Their system requires a great deal of expert user intervention to train the system and a talented artist to draw all the key-shapes. Each of the key-shapes must be manually specified for the source and target character, and parameterized by hand to find the affine deformations that the source key-shapes undergo before applying them to the target key-shapes. Their work provides a method for reusing the overall motion of the cartoon data, but it does not look at the structure of the data itself and therefore cannot re-sequence the data to expose meaningful new behaviors.

We are motivated by the work of Schödl et al. [SSSE00] on video textures to retain the original images in motion sequences but play them back in non-repetitive streams of arbitrary length. Video textures is most similar to our goal of re-sequencing cartoon images, specifically the "video-based animation" section of their work, although it is not user-directable. They use the L2 distance to compute the differences between frames for building the video structure. We want to compare the differences between frames in a similar fashion to analyze the data for re-sequencing. [SSSE00] assume a large data set with incremental changes between frames. Their methods do not extend well to cartoon data, which is inherently sparse and contains exaggerated deformations between frames. In their follow-up work [SE01], they use user-directed video sprites for creating new character animations. However, the examples shown require a vast amount of video data: 30 minutes of video footage for a hamster yielding 15,000 sprite frames. In our work, the largest cartoon data set we use has 2,000 frames, yet we still achieve good results with sparser data of 560 frames.

Recently, other researchers have found inspiration from video textures and have applied it to motion capture data. Sidenbladh et al. [SBS02] employ a probabilistic search method to find the next pose in a motion stream and obtain it from a motion database. Arikan and Forsyth [AF02] construct a hierarchy of graphs connecting a motion database and use randomized search to extract motion satisfying specified constraints. Kovar et al. [KGP02] use a similar idea to construct a directed graph of motion that can be traversed to generate different styles of motion. Lee et al. [LCR*02] model motion as a first-order Markov process and also construct a graph of motion. They demonstrate three interfaces for controlling the traversal of their graph. In our work, once the structure of the data is learned, the manifold that represents the data can be traversed to re-sequence the data.

2.2. Dimensionality Reduction

Dimensionality reduction for image data sets consisting of a large number of images has been used to represent a mean image, or subset of images, that are representative of the entire data set. A commonly used dimensionality reduction method is Principle Component Analysis (PCA) [Jol86], a linear embedding technique that will generate a mean image and eigenvectors that span the principle shape variations in the image space. However, this technique does not retain the spatio-temporal structure in the data that we are seeking. We assume our data have some underlying spatial surface (manifold) for which we wish to discover an embedding into a lower-dimensional space. Multidimensional scaling (MDS)[KW78] is another approach that finds an embedding that preserves the pairwise distances, equivalent to PCA when those distances are Euclidean. However, many data sets contain essential nonlinear structures that are invisible to PCA and MDS.

Two techniques for manifold-based nonlinear dimensionality reduction are Isomap [TdSL00] and Locally Linear Embedding (LLE) [RS00]. Both methods use local neighborhoods of nearby data to find a low-dimensional manifold embedded in a high-dimensional space. However, neither of these methods account for temporal structure in cartoon data. A modified version of Isomap, called Spatio-Temporal Isomap (ST-Isomap) [JM03], can account for the temporal dependencies between sequentially adjacent frames. We borrow the idea of extending Isomap using temporal neighborhoods from [JM03], and use ST-Isomap for dimensionality reduction of cartoon data to maintain the temporal structure

in the embedding. [JM03] focuses on synthesizing humanoid motions from a motion database by automatically learning motion vocabularies. Starting with manually segmented motion capture data, ST-Isomap is applied to the motion segments in two passes, along with clustering techniques for each of the resulting sets of embeddings. Motion primitives and behaviors are then extracted and used for motion synthesis. This type of analysis and synthesis also requires more data than is typically available for cartoon synthesis. Thus, we adapt the methods of [JM03] to use images as input, and use only one pass of ST-Isomap for creating the embedding used for re-sequencing.

3. Technical Approach

Since we are not generating new frames, the types of new motions that can be re-sequenced are restricted by the amount of data in our library for each character. Our method is model-free, requiring no a priori knowledge of the cartoon character. First, the cartoon data is pre-processed. Next, nonlinear dimensionality reduction is used to learn the structure of the data. Finally, by selecting a start and end frame from the original data set, the data is re-sequenced to create a new motion.

3.1. Pre-Processing Cartoon Data

Our input data comes from 2D animated video or 'toon-rendered motion capture. The video is pre-processed to remove the background and register the character relative to a fixed location throughout the sequence. There are a number of video-based tracking techniques that can be used for background subtraction, although currently we manually segment the images. Since our representation of the data is model-free, we do not need to identify any specific region of the character, i.e., limbs or joints, so it does not matter that the characters may undergo deformation. The registration is done using the centroid of the character in each frame and repositioning it to the center of the image, facilitating the computation of a distance matrix later.

We examine four cartoon sequences with different characters, a *gremlin*, *Daffy Duck*, the *Grinch*, and *Michigan J. Frog*. For the *gremlin* data set, there are 2,000 images of size 320 by 240 that are cropped and scaled to 150 by 180. The *gremlin* data set is created from three clips of motion capture of free-style dancing performed by the same subject, which is played through a gremlin model and 'toon-rendered on a constant white background. There are 560 images in the *Daffy* data set, with images of size 720 by 480, cropped and scaled to 310 by 238. There are 295 images in the *Grinch* data set and 146 images in the *Frog* data set, both sets with images of size 640 by 480. For these sequences, the characters are segmented and placed on a constant blue background. Figure 1 shows examples of the frames from the original data along with the corresponding segmented images. Our focus in this work is primarily with the *gremlin* and *Daffy Duck* data sets because of their larger size. We later discuss the issues with the smaller data sets.

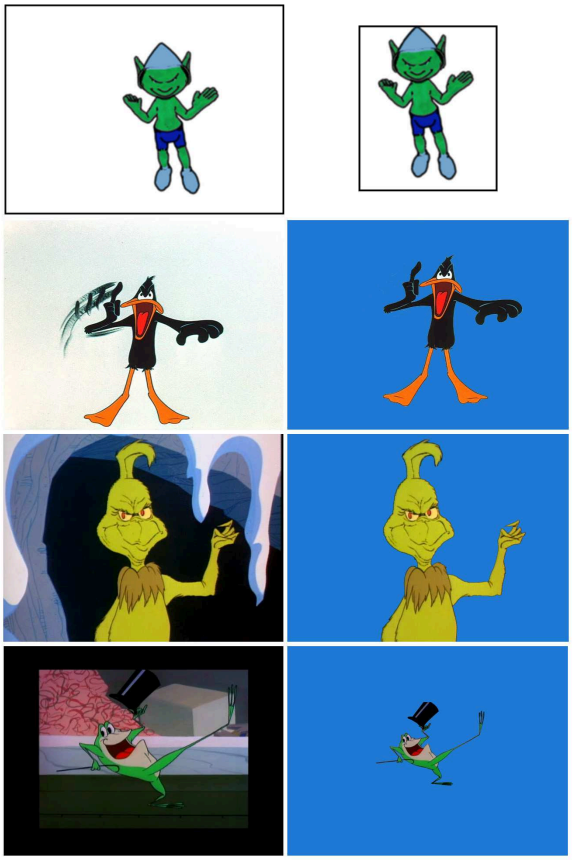


Figure 1: *The top row shows a frame from the* gremlin *data set before and after processing. The second row shows an original and cleaned up frame from the* Daffy Duck *data. An example frame from the* Grinch *data in the third row, and* Michigan J. Frog *in the last row.* Daffy *and* M.J. Frog ™& ©Warner Bros. Entertainment Inc. (s04)., Grinch ©Turner Entertainment Co.

3.2. Dimensionality Reduction

Nonlinear dimensionality reduction finds an embedding of the data into a lower-dimensional space. We use a modified Isomap, ST-Isomap, to perform the manifold-based nonlinear dimensionality reduction. Like standard Isomap, ST-Isomap preserves the intrinsic geometry of the data as captured in the geodesic manifold distances between all pairs of data points. It also retains the notion of temporal coherence, which is critical to the resulting output for cartoon data. ST-Isomap uses an algorithm similar to Isomap, as follows:

1. Compute the local neighborhoods based on the distances

$D_X(i,j)$ between all-pairs of points i, j in the input space X based on a chosen distance metric.
2. Adjust $D_X(i,j)$ to account for temporal neighbors.
3. Estimate the geodesic distances into a full distance matrix $D(i,j)$ by computing all-pairs shortest paths from D_X, which contains the pairwise distances.
4. Apply MDS to construct a d-dimensional embedding of the data.

The difference between Isomap and ST-Isomap is in step 2, where the temporal dependencies are accounted for.

One issue with Isomap is determining the size of the spatial neighborhoods. If the data is sufficiently dense, Isomap can form a single connected component, which is important in representing the data as a single manifold structure. The connected components of a graph represent the distinct pieces of the graph. Two data points (nodes in the graph) are in the same connected component if and only if there exists some path between them.

Our experimental results found that varying the size of the neighborhood (step 1) will ensure that a single connected component is formed regardless of the sparseness of the data. However, depending on the distance metric used and the sparseness of the data, the spatial neighborhoods need to be increased to a point such that no meaningful structure will be found. This issue arises with Isomap since its main objective is in preserving the global structure and preserving the geodesic distances of the manifold. ST-Isomap, by including adjacent temporal neighbors, remedies this deficiency, allowing a smaller spatial neighborhood size while forming a single connected component. Having all of the data points in the same embedding is desirable for re-sequencing. Using from one to three temporal neighbors and a small spatial neighborhood results in a meaningful structure that is usable for re-sequencing.

3.3. Distance Metrics

The key to creating a good lower-dimensional embedding of our data is the distance metric used to create the input to Isomap. When computing the local neighborhoods for $D(i,j)$, we examined three different distance metrics: the L2 distance, the cross-correlation between pairs of images, and an approximation to the Hausdorff distance [DHR93]. As mentioned previously, video textures uses the L2 distance for computing the similarity between video frames. Although this works well for densely sampled video, it is insufficient for dealing with sparse cartoon data.

3.3.1. L2 Distance

The first distance metric is the L2 distance between all-pairs of images. Given two input images I_i and I_j:

$$d_{L2}(I_i, I_j) = \sqrt{\|I_i\|^2 + \|I_j\|^2 - 2 * (I_i \cdot I_j)}$$

Only the luminance of the images is used for the L2 distance. The distance matrix $D_{L2}(i,j)$ is created such that

$$D_{L2}(i,j) = d_{L2}(I_i, I_j)$$

This metric is simple and works well for large data sets with incremental changes between frames, but is unable to handle cartoon data, which is inherently sparse and contains exaggerated deformations between frames.

3.3.2. Cross-Correlation Distance

The second distance metric is based on the cross-correlation between a pair of images. This metric also uses only the luminance of the images. Given two input images I_i and I_j:

$$c_{i,j} = \frac{\sum_m \sum_n (I_{i_{mn}} - \bar{I_i})(I_{j_{mn}} - \bar{I_j})}{\sqrt{(\sum_m \sum_n (I_{i_{mn}} - \bar{I_i})^2)(\sum_m \sum_n (I_{j_{mn}} - \bar{I_j})^2)}}$$

where $\bar{I_i}$ and $\bar{I_j}$ are the mean values of I_i and I_j respectively. This equation gives us a scalar value $c_{i,j}$ for the correlation coefficient between image I_i and image I_j in the range $[-1.0, 1.0]$. However, we want the correlation-based distance metric to be 0.0 for highly correlated images and 1.0 for anti-correlated images. Therefore the correlation-based distance matrix between images I_i and I_j is $D_{corr}(i,j) = (1.0 - c_{i,j})/2.0$.

3.3.3. Hausdorff Distance

The third distance metric is an approximation to the Hausdorff distance. This metric uses an edge map and a distance map of each image. The edge map E is computed using a standard Canny edge detector [Can86]. The distance map X is the distance transform calculated from E, and represents the pixel distance to the nearest edge in E for each pixel in X. Then, the Hausdorff distance between a pair of images I_i and I_j is:

$$D_{Haus}(i,j) = \frac{\sum_{(x,y) \in E_i \equiv 1} X_j(x,y)}{\sum_{(x,y) \in E_i \equiv 1} E_i(x,y)}$$

where E_i is the edge map of image I_i, X_j is the distance map of image I_j, and (x,y) denote the corresponding pixel coordinates for each image. Figure 2 shows an example of the edge map and distance map for a given frame.

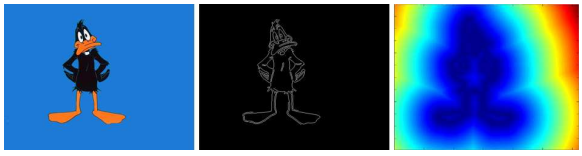
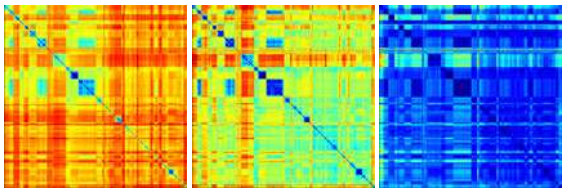


Figure 2: *An edge map in the center, and distance map on the right, for a frame from the* Daffy Duck *data set.* ™& ©Warner Bros. Entertainment Inc. (s04).

Figure 3 shows an example of the L2, correlation-based distance and the Hausdorff distance matrices for the *Daffy*



L2 Correlation-Based Hausdorff

Figure 3: *L2, Correlation-based and Hausdorff distance matrices for the* Daffy *data set.*

data set. A value of zero corresponds to similar images, while a value of one corresponds to dissimilar images. Note that the diagonal is zero as expected, and the banding indicates structure in the data. We found that the Hausdorff distance metric works best for all data sets.

3.4. Embedding

Once the distance matrix for a data set is computed, we apply ST-Isomap to obtain the lower-dimensional embedding of the cartoon data. The dimensionality of the embedding space must be determined. Choosing a dimensionality too low or too high results in incoherent re-sequencing.

Estimating the true dimensionality of the data using ST-Isomap is different than with PCA. In PCA, picking the dimensionality of a reduced data set can be done automatically such that the proportion of variance (shape variations) retained by mapping down to n-dimensions can be found as the normalized sum of the n-largest eigenvalues. This residual variance is typically chosen to be greater than 80% (usually 90%), while the remaining variance is assumed to be noise. PCA seeks to maximize the principal shape variations in the data, while minimizing the error associated with reconstructing the data from the lower-dimensional representation. The intrinsic dimensionality of the data estimates the lower dimensional subspace where the high dimensional data actually "lives".

In ST-Isomap, the residual variance is computed using the intrinsic manifold differences, which take into account possible nonlinear folding or twisting. We pre-select the number of dimensions in which to embed the data, from one to 10 dimensions. The true dimensionality of the data can be estimated from the decrease in residual variance error as the dimensionality of the embedding space is increased. We select the "knee" of the curve, or the point at which the residual variance does not significantly decrease with added dimensions. Figure 4 shows the residual variances and the 2-dimensional projections of the neighborhood graphs for all the data sets. The neighborhood graphs represent the manifold structure of the data, but only 2-dimensional embedding spaces are shown in the figure. Notice that the *gremlin* and the *Daffy* data sets are reduced to about five dimensions (Figure 4(i,k,m,o)), as indicated by the variance plots. The *Grinch* data can be reduced down to a three dimensional manifold (Figure 4(a,c)). The *Frog* data set is very sparse, and can at best be reduced to a five dimensional manifold.

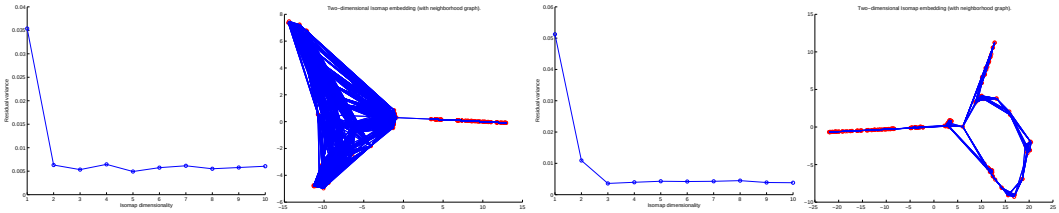
The differences in the variances and neighborhood graphs for the data sets in the figure are also influenced by varying the spatial neighborhood size for creating the original Isomap and the ST-Isomap embeddings. More spatial neighbors are included in the original Isomap to ensure that a single connected component is embedded for all data sets except the *gremlin*, which is sufficiently dense. The *Daffy* data set requires 20 spatial neighbors using original Isomap. For the *Grinch* data, 74 spatial neighbors are needed to generate a single connected component using original Isomap. Similarly, the *Frog* data requires 10 spatial neighbors to generate a single connected component using original Isomap. All data sets required seven spatial neighbors to generate the single connected component using ST-Isomap, while the temporal neighbors varied from one to three.

3.5. Re-sequencing New Animations

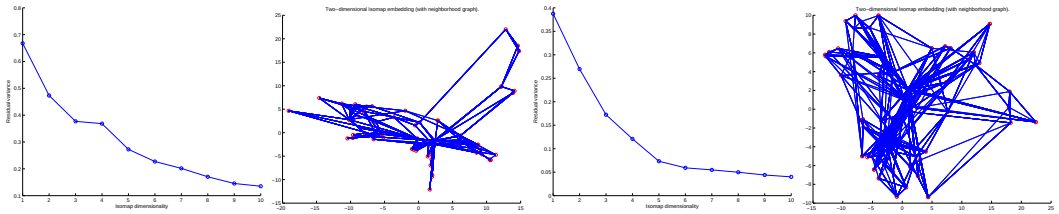
To generate a new animation, the user selects a start frame and an end frame, and the system traverses the Isomap embedding space to find the shortest cost path through the manifold. This path gives the indices of the images used for the resulting animation, which is created by re-sequencing the original, unregistered images. Dijkstra's algorithm [Sed02] is used to find the shortest cost path through the manifold. The dimensionality of the embedding space used for re-sequencing, i.e., for traversing the neighborhood graph, varies for each data set. The *Daffy* data set and the *Frog* data set use a 5-dimensional embedding space, the *gremlin* data set uses a 4-dimensional embedding space, and the *Grinch* data set uses a 3-dimensional embedding space.

3.5.1. Post-Processing

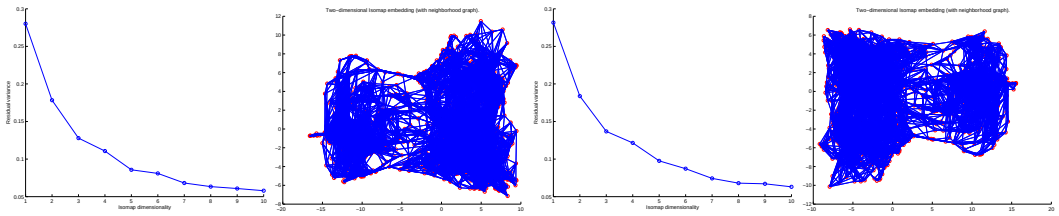
To ensure the smoothest looking re-sequenced animations, we add a small amount of automatic post-processing. Only the start and end keyframes for each re-sequenced segment are specified, but currently there are no restrictions on the number of inbetweens that the path should have. As such, the shortest cost path may not visit all temporally adjacent frames in the embedding space. To improve the re-sequenced animation, we process the frames specified from the path using the following automatic techniques. First, any missing sequentially adjacent frames within eight frames are inserted, helping to smooth some of the choppiness associated with skipping the missing frames. Sequentially adjacent frames are those that are adjacent in the original sequence. For example, if the re-sequenced path selected is [20 24 60 70] before inserting the sequentially adjacent frames, the resulting path becomes [20 21 22 23 24 60 70]. Using up to eight sequentially adjacent frames does not significantly change the overall re-sequenced path since the temporally



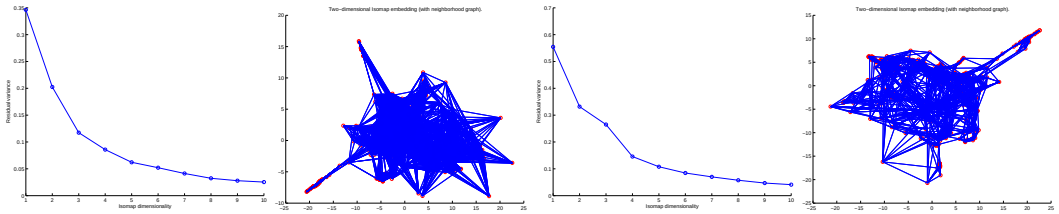
(a) *Grinch* original Isomap, variance plot (b) *Grinch* original Isomap, 2D graph (c) *Grinch* ST-Isomap with three temporal neighbors, variance plot (d) *Grinch* ST-Isomap with three temporal neighbors, 2D graph



(e) *Frog* original Isomap, variance plot (f) *Frog* original Isomap, 2D graph (g) *Frog* ST-Isomap with three temporal neighbors, variance plot (h) *Frog* ST-Isomap with three temporal neighbors, 2D graph



(i) *Gremlin* original Isomap, variance plot (j) *Gremlin* original Isomap, 2D graph (k) *Gremlin* ST-Isomap with two temporal neighbors, variance plot (l) *Gremlin* ST-Isomap with two temporal neighbors, 2D graph



(m) *Daffy* original Isomap, variance plot (n) *Daffy* original Isomap, 2D graph (o) *Daffy* ST-Isomap with two temporal neighbors, variance plot (p) *Daffy* ST-Isomap with two temporal neighbors, 2D graph

Figure 4: *Results showing the residual variance and 2-dimensional projection of the neighborhood graph generated with original Isomap and ST-Isomap using the Hausdorff distance matrix on the* Grinch, Michigan J. Frog, *the* gremlin, *and* Daffy Duck. *The number of temporal neighbors used is indicated in the figure.*

adjacent frames are usually near each other in the embedding space.

After adding these frames, we further improve the smoothness of the re-sequenced animations by matching the velocity of the centroid of each character from frame to frame in the new path. The new sequence was found based on the distance metric using registered images, described in Section 3.1. The registered images thus no longer possess any offset of the character within the frame. In post-processing, the original, unregistered images are used. For each original image in the data set, the character's centroid is calculated and stored. Then a velocity vector is computed based on each frame's previous and next temporal neighbor in the original (unregistered) sequence. When given a path for re-sequencing, the position and velocity of the centroid for the character in every frame are known. The position of the character is adjusted from one frame to the next in the new sequence based on the projected position indicated by the first frame's velocity vector from the original sequence. This adjustment is done whenever the path jumps from one single frame or subsequence in the path to another. Subsequences in the path are handled such that the first frame in the subsequence has its character repositioned based on the previous frame's projected position, while the remaining frames in that subsequence are adjusted to the first frame's new position.

Finally, if the character translates along the z-axis then the figure often changes in size within the frame. The final re-sequenced frames are adjusted using a scale factor based on the average pixel volume in the sequence. The scale factor s is defined as $s = \sqrt{\frac{Vol_a}{Vol_s}}$ where Vol_a is the average pixel volume in the entire path, and Vol_s is the average pixel volume of a subsequence (or just the pixel volume of a single frame). Then s is applied to each frame of the subsequence (or single frame) in the path.

3.5.2. Threshold Detection

In re-sequencing cartoon data, the transitions from the shortest cost path may result in a visual discontinuity. A small cost would indicate a good transition, while a large cost would indicate a bad transition. The system can automatically identify when the cost of a transition is too large. A threshold is determined for each data set, and notifies the user of abrupt transitions in the re-sequenced animation. The threshold is currently determined manually for each data set by examining the embedding structure and its associated costs. This notification allows the user to decide if additional source material or inbetweens are needed to produce a more visually compelling sequence.

4. Results

A demonstration of re-sequencing cartoon data with ST-Isomap can be seen in the accompanying video. The Hausdorff distance metric works best for all of our cartoon data. We found that for the *gremlin* data set and the *Daffy* data set, using two or three temporal neighbors yielded the best results. The *gremlin* data set is well populated with only a few large jumps at the transitions between motion capture clips, but the Hausdorff distance metric is an improvement over the L2 distance. For the *Daffy* data set, there are also a few large jumps in the original data resulting from the camera cuts for those scenes. The Hausdorff distance metric is significantly better than the L2, and reasonable paths are found through the embedding space.

We are able to re-sequence the *gremlin* data into a short motion clip that retains the same characteristics of the original dance motion, but shows a new dance behavior. This result was achieved by selecting six keyframes (sets of start and end frames) and applying ST-Isomap with two temporal neighbors, and post processed as described in Section 3.5.1. The result is a sequence with a total of 57 frames.

We also re-sequence the *Daffy* data into two short motion clips, each retaining the original characteristics of the gesturing motion, but showing a new gesturing behavior. The clips were created by selecting six and seven keyframes and applying ST-Isomap with two temporal neighbors. The first clip was minimally post-processed, only the missing temporally adjacent frames were inserted, and resulted in a sequence with a total of 59 frames. The second clip was post-processed by including any missing temporally adjacent frames and velocity-matching the centroids, resulting in a sequence with a total of 98 frames. Both clips show new gesturing behaviors.

Daffy	$246 \rightarrow 235$	0.413511	good
Daffy	$326 \rightarrow 77$	6.173898	bad
Daffy	$99 \rightarrow 243$	3.010666	accept
Daffy	$235 \rightarrow 236$	0.094055	good
Daffy	$98 \rightarrow 99$	7.270829	bad

Table 1: *Examples of the distance values between pairs of frames using the Hausdorff distance metric on the* Daffy *data set. Adjacent frames in the original data set may not always have a low distance value, as shown in the table. The transition from frame 98 to 99 is an abrupt transition according to the distance metric.*

After generating several re-sequenced animations for a particular data set, we inspect the cost values associated with the transitions and determine a threshold value for abrupt transitions. Once the threshold is determined, the system can use threshold detection to indicate to the user when a large transition cost has occurred. Our findings indicate that a threshold value of $D_{Haus} < 2.2$ represents a good transition while $D_{Haus} > 3.9$ represents an abrupt transition for

Figure 5: *An example of good, bad, and acceptable transitions for the* Daffy *data set from a path generated using ST-Isomap with two temporal neighbors. The pairs of frames shown correspond with the values shown in Table 1.* ™& ©Warner Bros. Entertainment Inc. (s04).

the *Daffy* data set. Table 1 shows some of the distance values associated with the transitions for a re-sequenced animation, while Figure 5 shows the frames referred to in the table. The transition from frame 99 to 243 has a value $2.2 \leq D_{Haus} \leq 3.9$, representing a region that should be inspected by the animator before accepting or rejecting. In this case it is accepted.

To test the system's ability to detect a large transition, an example is generated with three images from the *Daffy* data set removed. ST-Isomap is applied using two temporal neighbors and seven spatial neighbors. In the path generated from the data set with missing frames, the transition cost exceeded the pre-set threshold and resulted in a sequence with visual discontinuities. Inserting inbetween frames at the point of highest transition cost generates an improved sequence. Figure 6 shows the two paths without any post-processing. The sequence generated from the data set with missing images differs from the other sequence only in the transition from the first frame 326 to the second frame 77, which is where the inbetweens were added.

The *Michigan J. Frog* data set illustrates the challenges in re-sequencing cartoon data. This data set has 146 frames, of which only 73 are unique. Although ST-Isomap can reduce the data to approximately five dimensions, traversing the resulting embedding space for re-sequencing yields jumpy motion. A transition threshold can still be found even though the data set is so sparse. A threshold value $D_{corr} \leq 0.58$ represents a good transition. Figure 7 shows examples of good and bad transitions for the *Frog*, and the corresponding transition costs, for a path generated using ST-Isomap with three temporal neighbors.

5. Conclusion and Future Work

We are able to re-sequence cartoon data to new animations that retain the characteristics of the original motion. Our method is model-free, i.e., no a priori knowledge of the drawing or cartoon character is required. The keys to the method are the identification of a suitable metric to characterize the differences in cartoon images and the use of a non-linear dimensionality reduction and embedding technique, ST-Isomap. The system can characterize when a novel re-sequencing requires additional source material to produce a visually compelling animation.

We foresee that this system will be useful as an aid to artists charged with generating inbetweens in cel animation. If a sufficient body of prior animation is available, the inbetween artist could use the system to match keyframes in a new animation and generate inbetweens from existing data. Only if the keyframes were sufficiently novel or the transition cost too high would the inbetween artist be required to generate new art.

We would like to address the issue of synthesizing new

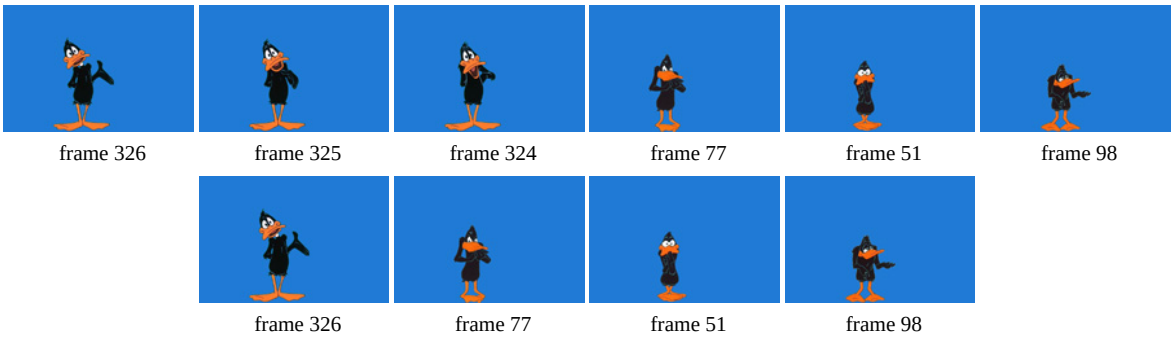


Figure 6: *A filmstrip of two paths without any post-processing. The bottom row shows the path generated from the* Daffy *data set with three frames removed. The top row shows the same path with inbetweens inserted at the point of highest transition cost, in this case between frames 326 and 77.* ™& ©Warner Bros. Entertainment Inc. (s04).

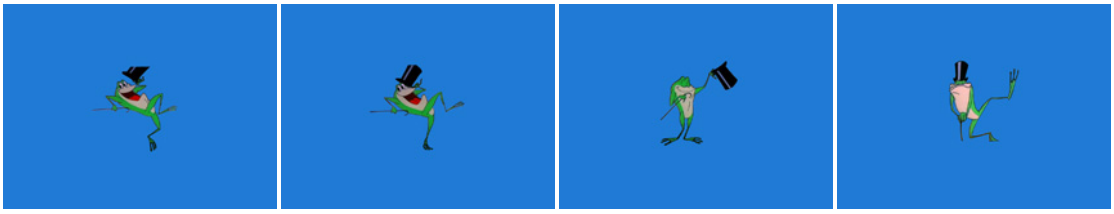


Figure 7: *An example of good and bad transitions for the* Frog *data set. The first pair of images demonstrates a good transition from frame 22 to 27 with a cost of 0.198132. The second pair of images demonstrates a bad transition from frame 12 to 109 with a cost of 0.609729.* ™& ©Warner Bros. Entertainment Inc. (s04).

data, i.e., generating transitions with blending or interpolating. Currently, the system will only determine that an abrupt transition has been made, but it cannot automatically generate the necessary inbetweens. Using optical flow for generating the inbetween frames may work if the two frames are not significantly dissimilar. Another technique to investigate is smoothing the transitions by adding motion blur [BE01].

Other error metrics more specific to cartoon images, such as perception-based image metrics, may reveal how the human visual system accepts certain types of transitions in an animated character, while other transitions are obviously bad. Even though some of the data is sparse, it was originally drawn that way, and when playing back the animation, some of the frames that are considered abrupt transitions by our system may actually be visually acceptable by the viewer. User studies may provide some insight into this behavior.

Another improvement would be to explore other methods of traversing the Isomap embedding space. The shortest cost only represents the similarity between two frames. Some cartoon motions that are very expressive and exaggerated may call for a quick transition between dissimilar frames. In this case, the lowest cost would not be appropriate. Post-processing the re-sequenced animations helps produce smoother results, but the results may still be too discontinuous. We would like to investigate how adding a component of velocity (similar to the post-processing) to the distance metric may change the embedding space, and thus change the resulting re-sequencing. One possible way of incorporating the "velocity" into the distance metric would be to calculate the optic flow of the edge map of each image and use it to estimate a velocity term.

Finally, the sparseness of the data is an issue because of the slow acquisition of clean and segmented images of cartoon characters. To acquire a larger amount of data more quickly, we are looking into automatic methods of background segmentation. One method we have begun using is a level set method that looks at the character as regions of specific color values.

6. Acknowledgments

The authors thank Chad Jenkins for his insights in using ST-Isomap. We thank Steve Park and the Graphics, Visualization, and Usability Center at the Georgia Institute of Technology for supplying the motion capture data used in this study. We also thank Robert Pless for his insightful suggestion on how to quickly compute the Hausdorff metric. We wish to thank Gary R. Simon at Warner Bros. Entertainment Inc. for giving us permission to use the images of *Daffy Duck*

and *Michigan J. Frog*. We also thank the reviewers for their helpful comments. This research was supported by National Science Foundation Grant IIS-0237621.

References

[AF02] ARIKAN O., FORSYTH D. A.: Synthesizing constrained motions from examples. *ACM Transactions on Graphics 21*, 3 (July 2002), 483–490. 2

[BE01] BROSTOW G. J., ESSA I.: Image-based motion blur for stop motion animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 561–566. 9

[Bla94] BLAIR P.: *Cartoon Animation*. Walter Foster Publishing, Inc., 1994. 1

[BLCD02] BREGLER C., LOEB L., CHUANG E., DESHPANDE H.: Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics 21*, 3 (July 2002), 399–407. 1, 2

[Can86] CANNY J.: A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 6 (1986), 679–698. 4

[CJTF98] CORREA W. T., JENSEN R. J., THAYER C. E., FINKELSTEIN A.: Texture mapping for cel animation. *Computer Graphics 32*, Annual Conference Series (1998), 435–446. 1

[DHR93] D.P. HUTTENLOCKER G. K., RUCKLIDGE W.: Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 9 (September 1993), 850–863. 4

[FBC*95] FEKETE J., BIZOUARN É., COURNARIE É., GALAS T., TAILLEFER F.: TicTacToon: A paperless system for professional 2-D animation. In *SIGGRAPH 95 Conference Proceedings* (1995), Cook R., (Ed.), Addison Wesley, pp. 79–90. 1

[JM03] JENKINS O. C., MATARIC M. J.: Automated derivation of behavior vocabularies for autonomous humanoid motion. In *Proc. of the 2nd Intl. joint conf. on Autonomous agents and multiagent systems* (2003), pp. 225–232. 2, 3

[Jol86] JOLLIFFE I.: *Principal Component Analysis*. Springer-Verlag, New York, 1986. 2

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Transactions on Graphics 21*, 3 (July 2002), 473–482. 2

[KW78] KRUSKAL J. B., WISH M.: *Multidimensional Scaling*. Sage Publications, Beverly Hills, 1978. 2

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3 (July 2002), 491–500. 2

[PFWF00] PETROVIC L., FUJITO B., WILLIAMS L., FINKELSTEIN A.: Shadows for cel animation. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), ACM Press / ACM SIGGRAPH / Addison Wesley Longman, pp. 511–516. 1

[Ree81] REEVES W. T.: Inbetweening for computer animation utilizing moving point constraints. In *Siggraph 1981, Computer Graphics Proceedings* (1981), pp. 263–269. 2

[RS00] ROWEIS S., SAUL L.: Nonlinear dimensionality reduction by locally linear embedding. *Science 290* (2000), 2323–2326. 2

[SBS02] SIDENBLADH H., BLACK M. J., SIGAL L.: Implicit probabilistic models of human motion for synthesis and tracking. In *Computer Vistion — ECCV 2002 (1)* (May 2002), Heyden A., Sparr G., Nielsen M.,, Johansen P., (Eds.), Springer-Verlag, pp. 784–800. 7th European Conference on Computer Vision. 2

[SE01] SCHÖDL A., ESSA I.: Controlled animation of video sprites. In *Symposium on Computer Animation* (2001), ACM Press / ACM SIGGRAPH, pp. 121–127. 2

[Sed02] SEDGEWICK R.: *Algorithms in C: Part 5 Graph Algorithms*, 3 ed. Addison-Wesley, 2002. 5

[SG92] SEDERBERG T. W., GREENWOOD E.: A physically based approach to 2-D shape blending. In *Proceedings of ACM SIGGRAPH 1992* (1992), Catmull E. E., (Ed.), vol. 26, pp. 25–34. 2

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), ACM Press / ACM SIGGRAPH / Addison Wesley Longman, pp. 489–498. 1, 2

[TdSL00] TENENBAUM J. B., DE SILVA V., LANGFORD J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science 290* (2000), 2319–2323. 2