Image Spaces and Video Trajectories: Using Isomap to Explore Video Sequences

Robert Pless
Department of Computer Science and Engineering,
Washington University in St. Louis,
pless@cse.wustl.edu

Abstract

Dimensionality reduction techniques seek to represent a set of images as a set of points in a low dimensional space. Here we explore a video representation that considers a video as two parts – a space of possible images and a trajectory through that space. The non-linear dimensionality reduction technique of Isomap, gives, for many interesting scenes, a very low dimensional representation of the space of possible images. Analysis of the shape of the video trajectory through these image spaces gives new tools for video analysis. Experiments with natural video sequences illustrate methods for the very different tasts of classifying video clips and temporal super-resolution.

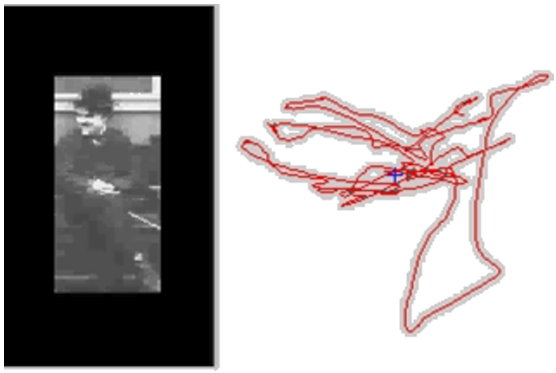

Figure 1: A frame from a short video clip of Charlie Chaplin in a skit where his hat lifts from his head three times, and he fixes it once. This paper considers the problem of creating "video trajectories" — a representation of changes in a video sequence, based upon the non-linear dimensionality reduction technique called Isomap.

1 Introduction

Faster computation and more memory have opened up new avenues of research in image and video analysis. One method that strays particularly far from the classical bottom-up or geometric approach to computer vision is "image similarity based image analysis". This is a family of techniques which considers the image as a whole as the fundamental atom of analysis. A collection of related images defines an image space, and the subject of this paper is an exploration of the image spaces formed by video sequences.

All $m \times n$ pixel images exist in the mn-dimensional space where each dimension is the intensity at a particular pixel. However, considering a set of images as a set of points in this space does not give natural tools for image understanding or analysis. Dimensionality reduction algorithms seek to map each image to a point in a lower dimensional space, as a tool for analysis or image compression.

This contribution here is to illustrate a technique for video representation using the (relatively) new non-linear dimensionality reduction technique of Isomap [13]. This representation considers video as: (1) a collection of unordered images which (via Isomap) define an image space, and (2) an ordering of these images which specifies a trajectory through that image space. Figure 1 gives an illustration of the goal, to transform a short video clip into a trajectory. The algorithm is essentially a straight forward extension of Isomap to the domain of time-sequenced imagery. What is new is: (a) the demonstration that this gives a useful representation, (b) a description of the algorithmic choices required to implement this algorithm on different video data sets, and (c) the illustration of two preliminary, but quite different, sample applications that show the benefits of this approach.

These methods work best when there exists a two or three dimensional underlying parameterization of

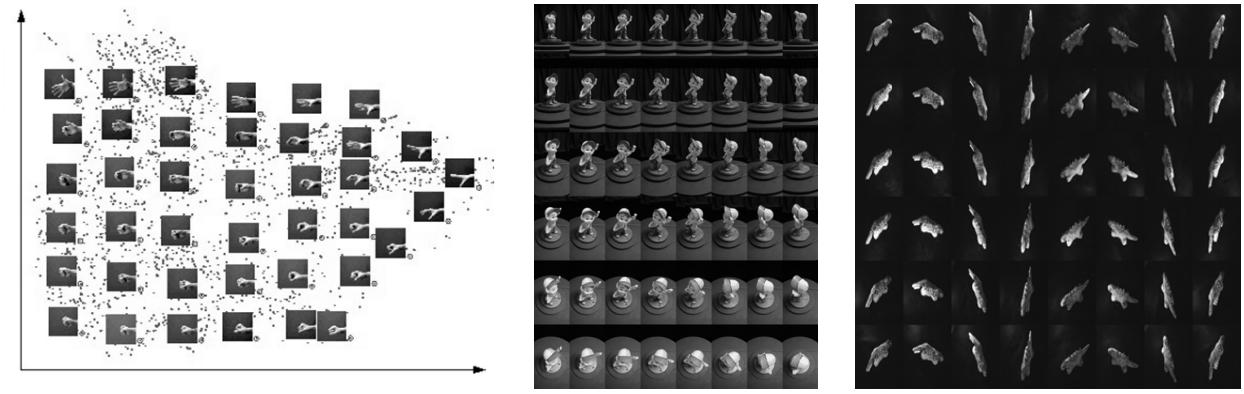

Figure 2: Examples of image spaces defined by non-linear dimensionality reduction of large image sets. In all three cases, the algorithm input is an un-organized set of images. (left) The original isomap algorithm gives a qualitative organization of images of gestures into axes of wrist rotation and finger extension [13]. (middle) This method can be modified to give quantitative pose-estimates for (object rotation angle and camera elevation angle) for thousands of images of an object, with a mean error on the order of 3° [7]. (right) The MDS algorithm has also been modified to embed images in cyclic dimensions [6].

the data set. For instance, a video of a (perfectly tracked and centered) bird flying consistently across the sky can be parameterized by the phase of the periodic action ("where are the birds wings are in the flap-cycle"), and the relative orientation of the bird and the camera. It may seem that few scenes are defined by so few parameters, but a great deal of video data is taken in highly limited, stylized, or unchanging environments — many surveillance videos or short clips of movies (within one scene) fit this model quite well.

2 Background

A classical dimensionality reduction technique for a set of images is Principle Component Analysis (PCA) [5]. For any desired dimension k, PCA represents each image in the set as a weighted sum of k basis images, and these basis images are chosen to minimize the total (sum of the squared intensity) difference between each image and its reconstruction. PCA is an ideal dimensionality reduction technique because there is a natural function mapping an image to a point in the space (projection onto the basis images), and a natural function mapping points in the space to images (weighted sum of the basis images). Independent Component Analysis (ICA) [4] uses different techniques to find the set of basis images, but still uses projection and linear combination to map between images and the low-dimensional points representations of images.

However, many natural sets of images are not well represented by a linear combination of few basis images. Even image sets that can be indexed by a single parameter (many instances of a clock pendulum, for instance) may require many basis images to reconstruct the images with reasonable accuracy. Furthermore, the simple structure of such an image set may not be obvious from the set of coefficient weights which define each image.

In this paper we consider a different set of methods for dimensionality reduction. These methods do not rely on linear combinations of basis images and do not have natural functions which map between images and points in the abstract representation space. However, they find low-dimensional structures which correspond to natural parameterizations of the image set. These dimensionality reduction techniques first find images which are similar to each other, and then seek to discover a global structure by solving a system which combines these similarity constraints. Two methods of this form were recently proposed, LLE [8] and Isomap. This work concentrates on the Isomap approach, which is briefly defined in Section 3. Isomap is successful in automatically finding parameterizations for many image sets, including the perceptual organization of gesture images, and quantitative pose estimates for images of an unknown object (see Figure 2). The analysis of the image set after is has been parameterized using Isomap is simplified

because the parameterization often has the same dimension as underlying data set.

When an image set arises from a video sequence, there is additional information available in the order of the images. One application of video analysis has been to find patterns in scenes with consistent but non-periodic motions such as waving grass, streams, smoke [3, 12]. These approaches can both be characterized as a two step process. First use PCA to define a low dimensional image representation based upon several hundred frames of a video sequence. Second, consider the video as a set of points in that space, look at the trajectory of the image sequence, and find methods for extending that trajectory through time. For some types of dynamic textures, this gives very natural and believable continuations of a video sequence, but new video images are often blurry because they are fundamentally a weighted sum of images in the original set.

There is related work which uses similarity measures in the context of video imagery. Embedding a set of images with MDS into a 2D space and then using K-means to find a set of exemplar images has been used as a video summarization technique [11]. Video Textures [10] considers the video as a linear structure and seeks to find similar images in different parts of video sequence. Similar images serve as good looping points to extend video clips indefinitely without a perceptual break. Additionally, the analysis of periodicity in a video sequence has been used to distinguish between videos of human walking motions, running dogs, and vehicular motions [2].

3 Isomap and MDS

In this section we give a brief overview of the mathematics behind Isomap and MDS. A complete description is available in the original articles [13], and longer tutorials on these methods and MDS in general give more specific implementation details [1].

Multi-Dimensional Scaling:
Input: D = $n \times n$ matrix of all squared pairwise distances
Output: Point coordinates which best approximate pairwise distances.
Method: Solve an eigenvalue problem with a matrix easily created from D. This produces the best k-dimensional point set for any dimension k. This is "best" is the following sense: let D_k be the distance matrix defined by the k-dimensional point set computed from MDS. MDS returns a k-dimensional point set that minimizes the Frobenius norm of $D - D_k$ (i.e., of all possible k-D point sets, MDS returns on whose distance matrix is closest to D, measured by sum of the squares of the element differences).

Isomap
Input: an $n \times n$ matrix pairwise distances with some (perhaps most) distances unknown.
Output: Point coordinates such that the pairwise distances are best approximated.
Method: Define a graph whose vertices are the set of points, and whose edges are the known pairwise distances. Compute all-pairs shortest path distances in this graph, which defines a distance between every pair of nodes. Use MDS to find point coordinates which satisfy these (now complete) distance constraints.

4 Image Similarity Based Video Analysis

A video is an ordered set of images. If a set of images has an associated image space, the video can be represented as a trajectory through that space. The shape of the image space and the form of the trajectory provide new tools for the automatic analysis of video imagery. The following algorithm generates a video trajectory, and is graphically illustrated in Figure 3. Implicit in the algorithm are a collection of choices. These choices are required to specialize the algorithm for a particular application. These are marked with a label [*], and are discussed after the algorithm.

1. Define a distance measure between pairs of images, $f : I \times I \longrightarrow R$. *[1]*

2. Define a threshold function to determine which image distances to ignore. Let $\theta(i, j) = 0$ if the image distance between image i and image j is to be ignored, and 1 otherwise. *[2]*

3. Define a sparse matrix M whose i,j entry is $f(\text{image}_i, \text{image}_j)$ if $\theta(i, j) = 1$ and undefined otherwise.

4. Embed the sparse distance matrix M using Isomap. Choose the appropriate dimension for the embedding. [3]

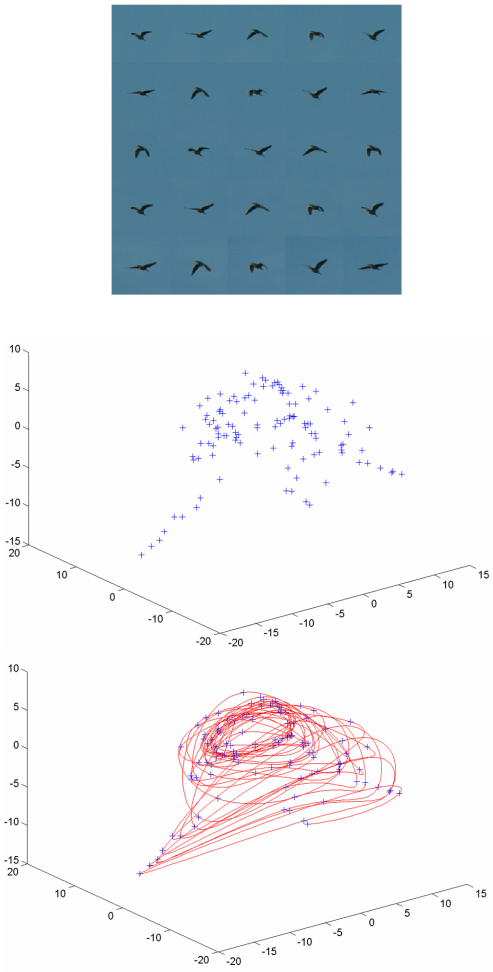

Figure 3: An illustrated outline of the procedure for creating a video trajectory. (top) Consider the input video as a set of images. (middle) compute the distance between each pair of images, and use MDS to embed the images as a point set in a low dimensional space. (bottom) Draw a spline curve connecting the points in the order the corresponding images appeared in the original video.

5. Draw a spline curve connecting the points in the order the images were in the original video.

6. Use the video trajectory representation for specific applications. [4]

This general framework is used for the construction of the video trajectories for all the examples shown in Section 4. Steps one through four define the image space intrinsic to the set of images in the video. Step five uses the ordering of the images to define the path that the video sequence takes through the image space.

To specialize this framework for particular applications, it is necessary to specify the choices of distance measure, threshold, and embedded dimension. Properties of each of these choices are discussed in turn.

0. Pre-processing. Before a distance measure between images is computed, it is often useful to pre-process the images. Often, application specific pre-processing can greatly reduce the underlying dimensionality of the input set. In the "bird" experiments discussed later, the initial images are of a bird against a constant background. The center of mass of the non-background pixels was calculated and the image was cropped to a constant sized box around this point. Then, the image differences are calculated between between images of a cropped and centered bird. Without this pre-processing, the parameterization of the bird scene requires two additional parameters. This would force the algorithm to embed the points in a higher dimension and require far more input data. For the Fountain clip used in Figure 4(right) and the Chaplin clip in Figure 5, no pre-processing was necessary.

1. Distance Measure. The first choice that needs to be made is the image distance measure. This function is initially applied to every pair of images and returns a distance between these images Surprisingly, the exact distance measure used often affects the final result very little. This is because in the next stage we disregard all but the smallest distance measures. The power of a method like Isomap derives in a large part from the (graph) structure of which images are similar rather than the actual similarity measures themselves. Image distance measures that we have used successfully include normalized cross-correlation, correlation of edge-maps, naive distance measures such as sum of squared pixel differences, sophisticated general distance measures such as "Earth-Movers Distance" [9], and distance measures designed for specific applications.

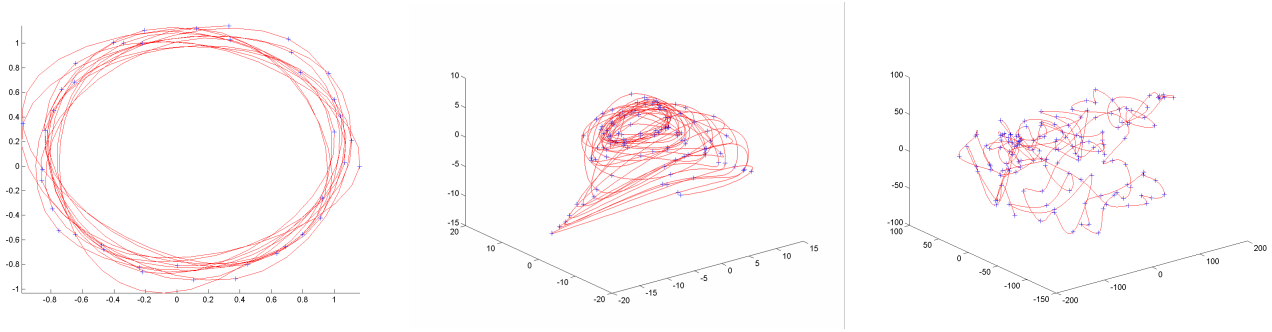

Figure 4: Video Trajectories have characteristic forms for different types of video imagery. (Left) The video trajectory of a sequence viewing only the pendulum of a clock. (Middle) A camera panning to view a bird in flight. (Right) A fountain sequence used in the dynamic textures work [12]

2. Threshold. The distance measure is a function defined on all pairs of images. Especially for images, distance measures are only reliable when the images are very similar — the reason that Isomap is a good procedure for embedding images is that it only needs a sparse set of distance measures. There are several option in how to choose which image distances to keep. One is for each image, to keep the distances to the k-closest neighbors[1]. Another options is to specify a threshold and choose to use all distances lower than that threshold.

3. Embedding Dimension. The Isomap embedding proceeds by first completing the distance matrix M and then using MDS to embed this matrix and find points. For an images set with n images, the MDS algorithm can embed the images into an arbitrary dimension k, for $1 \leq k \leq n$. For each dimension there is a residual error — this error is the difference between the distances specified in the sparse distance matrix and the distances between those point pairs after they have been embedded.

4. Applications. Once the images have been embedded in a low dimensional space, the order of the images defines a path through this space. Together, the image space and the trajectory define useful information about the sequence, although how this trajectory is interpretted is application dependent. In the next section, we illustrate some preliminary results in this direction.

[1]It is possible for image i to be one of the k-closest neighbors to image j, but image j not to be one of the k-closest neighbors of image i. To keep our sparse matrix symmetric, we include both.

4.1 Implementation and Results

For a video sequence with 200 images, a matlab implementation for this complete procedure takes approximately 4 minutes to execute on a 800 MHz laptop. Most of this time is spent in the initial computation of all pairs of image distances. The form of the resulting trajectory depends upon the input video sequence, and there are a number of patterns that often occur in the trajectory. Examples of actual trajectories created from a variety of real video sequences are illustrated in Figure 4. The "bird" videos used a pre-processing step as described in the previous section, and all videos used sum-of-squared differences of pixel intensity as the distance measure (always between images of identical size). The distances chosen as acceptable and included in the sparse distance matrix as an input to Isomap were the 10 smallest distance for each point. The computed trajectories fall into five natural categories:

Cyclic Completely repetitive video sequences have trajectories which are embedded as ellipses in a two-dimensional space. Images which are separated arbitrarily in time may be identical because the scene in view is changing in a periodic manner.

Helical A periodic action being viewed by a moving camera is characterized by similarity between local images separated by the period of the action, but a drift over time in appearance of the object even in the same phase. This leads to a helical structure in the trajectory. Both the helical and the cyclic trajectories can be detected based upon a Fourier analysis of the distance matrix (for example, see [2]).

Knotted Dynamic textures (fountains, smoke, flames, and natural motions of trees in the wind) are

characterized by a non-periodic sampling from a limited image space.

Linear Videos which are smoothly changing but not repetitive (such as a slow pan across a landscape) tend to have a linear structure because frames from different parts of the video are rarely judged to be similar.

Combinations Video sequences may have smooth transformations between pieces that fit cleanly into the one of the above categories.

4.2 Sample Application: Video Segmentation

The video trajectory is a low dimension description of the video sequence. For many sequences, this form of this trajectory highlights important transitions in the scene or deviations from the normal scene appearance. Figure 5 shows video trajectories and their interpretation for two different video sequences.

The first clip is of an isolated bird flying, then gliding. The transition between these different dynamic activities is clearly visible in the trajectory. This trajectory can be broken into periodic regions (shown in the thinner, red line) and non-periodic regions (shown in the yellow, thicker line). The transition point exactly segments the clip into when the bird is flying and when the bird is gliding.

The second clip is a more complicated scene, part of a clip with Charlie Chaplin in a skit where his hat comes up several times and he moves his arm to adjust it once. This trajectory was created with exactly the same code as the trajectories created of the bird videos. The trajectory has four major features that exactly correspond to the four events in the video clip, the hat coming up (three times), and Chaplin raising his left arm to adjust this hat (once). This video clip and its trajectory is included in the video supplement to this submission.

4.3 Sample Application: Temporal Super-Resolution

For video sequences where similar images occur at different times (the cyclic, helical, or knotted cases listed above), the video trajectory can be used to intelligently interpolate between consecutive images. The trajectory is represented by a spline curve fit through the points corresponding to the images in the video. In order to create a higher frame rate video, it is necessary to create images between subsequent images

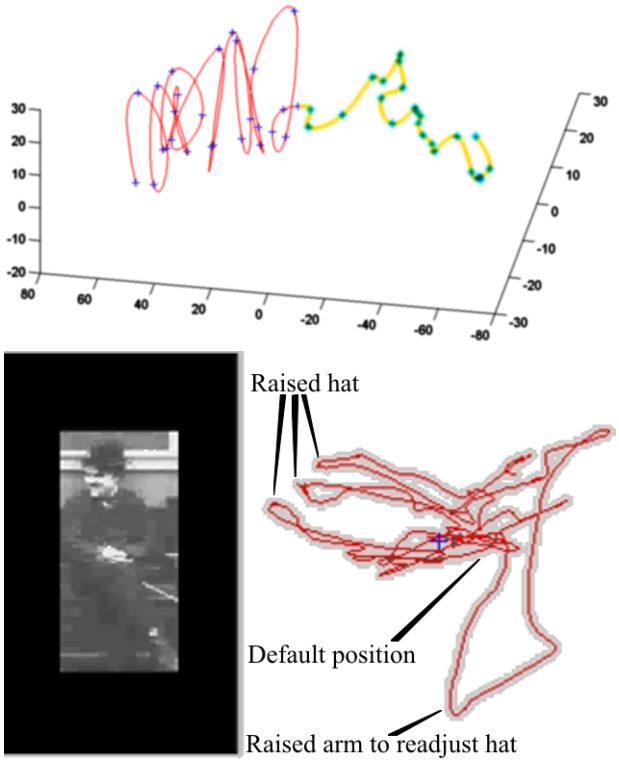

Figure 5: Video Interpretation. (Top) A clip of an isolated bird flying, then gliding. The transition between these different dynamic activities is clearly visible in the trajectory. (Bottom) The Charlie Chaplin video discussed in the first figure. The three parts of the clip where his has comes up are clearly defines, as is the long period when he uses his left arm to readjust his hat. Please see video included with this submission. The segmentation of the top trajectory is automatically determined based on a threshold of the curvature of the spline curve. The labelling of points on the lower trajectory was provided by the author.

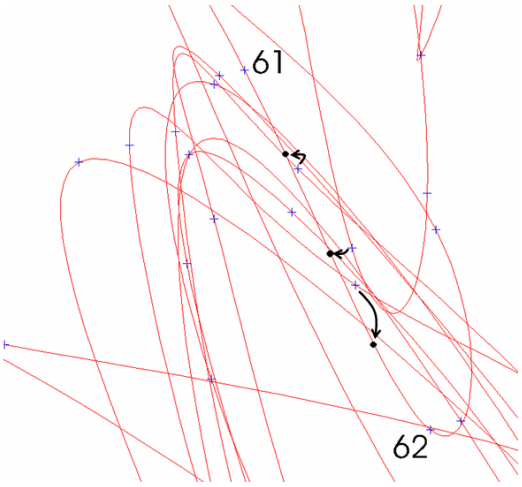

Figure 6: Image resampling: An expanded view of the trajectory shown in Figure 3. Shown are the points corresponding to each original frame, with two consecutive frames labelled, and the intermediate points along the spline. A smooth transition between consecutive images is made by choosing the images whose embedded points are closest to the intermediate points along the spline.

in the original video.

Consecutive frames are connected by a continuous path through the image space. Intermediate points on this path correspond to images that smoothly interpolate between frames. A weakness of Isomap (relative to PCA, for instance) is that there is no function which maps points in image space to images. Instead it is necessary to search to find the image whose point in image space is closest to the intermediate point. For this to be effective, it requires that the original video sequence provide a relatively dense sampling of the image space. Figure 6 shows an example of two consecutive frames from a video trajectory, intermediate points on that trajectory, and the frames from elsewhere in the video which are closest to the intermediate frames.

An example of the frames chosen between two frames from the original sequence of a bird flying is shown in Figure 7. These frames are contrasted against an alternative method of interpolating between images which consists of a weighted average of the original frames. The video files included with this submission give several examples of long sequences of video that are smoothly interpreted with 5, 10, and 20 images inserted between each original image.

This re-sampling algorithm may duplicate images — applying the interpolation algorithm to a linear trajectory (a non-repetitive video) simply uses the original images multiple times, if that original frame is the closest to the interpolating points. A hybrid algorithm which uses image re-sampling when possible and image morphing otherwise may give better performance in general.

5 Conclusion and Future work

A video defines an image space and the trajectory through that image space. This trajectory depends on the pattern of similarity between images, and analysis of this trajectory is new tool for video manipulation. This tool applies to natural imagery of non-rigid motions, does not require a prior model of object shape, and takes advantage of the increased computational power and memory resources available in modern computers.

Future work should concentrate on two fronts. First, the algorithm begins by computing a similarity measure between every pair of images. The Isomap procedure uses only the image similarity measurements which are most similar, so it is not necessary to compute all pairs of image distances. Second, there are significant differences between the dimensionality reduction techniques of Isomap and LLE, these differences may affect their applicability to video analysis tasks.

Included Video. Included with this paper is a zip file containing two videos. The first is a long sequence of the temporal-super-resolution algorithm applied to a flying bird, adding 19 images between each original image. This right of this side-by-side video shows an alternative method of adding frames by smoothing blurring between frames of the original sequence. The second file show the Charlie Chaplin video side by side with the video trajectory, with a marker that shows the position of the current frame in the trajectory.

References

[1] Ingwer Borg and Patrick Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag, 1997.

[2] R. Cutler and L. Davis. Robust real-time periodic motion detection, analysis and applications.

Figure 7: Temporal Super-Resolution: From a sequence of a bird flying, the following shows 19 images inserted between two frames of the original sequence using the process described in Figure 6 (some original images are inserted multiple times). This is contrasted with a smooth blurring between the original images (the bottom layer).

IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 781–796, August 2000.

[3] Andrew Fitzgibbon. Stochastic rigidity: Image registration for nowhere-static scenes. In *Proc. International Conference on Computer Vision*, pages 662–670, 2001.

[4] A Hyvrinen, J Karhunen, and E Oja. *Independent Component Analysis*. John Wiley and Sons, 2001.

[5] I T Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[6] Robert Pless and Ian Simon. Embedding images in non-flat spaces. In *Conference on Imaging Science Systems and Technology*, pages 182–188, 2002.

[7] Robert Pless and Ian Simon. Using thousands of images of an object. In *CVPRIP*, 2002.

[8] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000.

[9] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Proc. International Conference on Computer Vision*, pages 59–66, 1998.

[10] Arno Schdl, Richard Szeliski, David Salesin, and Irfan Essa. Video textures. In *Proceedings of SIGGRAPH*, pages 489–498, 2000.

[11] Haim Schweitzer. Computing content-plots for video. In *Proc. European Conference on Computer Vision*, pages 491–501, 2002.

[12] S Soatto, G Doretto, and Y N Wu. Dynamic textures. In *ICCV*, pages 439–446, 1998.

[13] Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.