Interpolating images between video frames using non-linear dimensionality reduction

Sébastien ROBASZKIEWICZ ROBINIO@STANFORD.EDU
Sammy EL GHAZZAL SELGHAZZ@STANFORD.EDU

Abstract

We present a new method for interpolating images between frames of a video. By applying Isomap, we map each frame of the video to a low dimension feature space, which extracts meaningful features from the sequence of images. To insert new images between 2 frames, we select images from the original dataset that are mapped in the same area as those 2 frames. We assess the performance of our algorithm by confronting it to 2 other interpolation algorithms on 2 different datasets. We delete every other frame from the original datasets, and reinterpolate the deleted images with 3 different methods. By comparing the similarity of the interpolated image and the deleted image with measures based on SIFT and the Hough Transform, we determine which algorithm leads to the lowest error. Our results suggests that the method based on Isomap performs better than the two other algorithms when the video contains repetitive motion.

1. Introduction

Video compression has always been an important challenge, and it is now more crucial than ever due to the wide use of streaming services. A way of achieving compression is to delete images, and interpolate them back during the decompression process. In other words, the compression reduces the frame rate of the video and the decompression is aimed at bringing the frame rate back to its original value. As suggested by (Pless, 2003), we investigate whether or not dimensionality reduction can be used to solve this problem.

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

2. Method

2.1. Insuffisance of a linear approach

In this project, we intend to take advantage of the underlying pattern in a video sequence to interpolate images between two consecutive frames. In order to find the dimension of the underlying structure in a sequence of images, we confronted two algorithms: Principal Component Analysis (PCA) and Isomap (Tenenbaum et al., 2000), a nonlinear dimensionality reduction algorithm. The main difference in principle between these two algorithms is that while PCA uses Euclidean distances, Isomap computes a good approximation of pairwise geodesic distances uses them in a spectral method to compute a low-dimensional embedding.

We found out that PCA needed 59 principal components to capture 95 % of the variance in the dataset (which consisted in a teapot revolving around the z-axis), whereas Isomap reaches a minimal residual variance of 4 % for a two dimensional embedding, therefore suggesting that the data lies in the neighborhood of a two dimensional manifold (cf. Figure 1).

2.2. Using Isomap

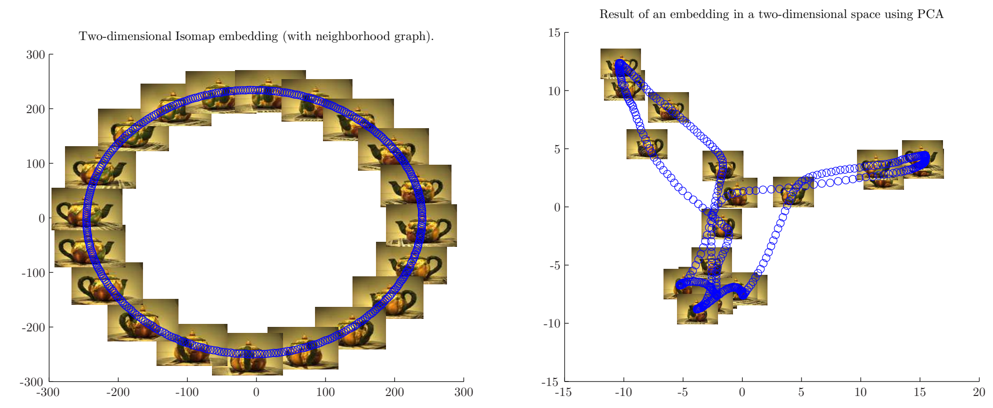
Given a point in $\mathbb{R}^d$, we are not able *a priori* to find its pre-image(s)[1]. There is no guarantee either that the pre-image will be unique (injectivity of ϕ) or that such a pre-image even exists (surjectivity of ϕ).

Knowing the mapping function ϕ would allow us to find an approximation of the pre-image by using a regression technique. We would compute x^* such that:

$$x^* = \arg\min_{x \in \mathcal{X}} \|\phi(x) - y\|$$

where y is the point for which we are trying to find a pre-image. Initializing x to some value, we would

[1]We define the pre-image of a point $y \in \mathbb{R}^d$ in the feature space to be a point x such that $y = \phi(x)$ where ϕ denotes the mapping function between the input space and the low-dimensional space.



(a) Isomap (b) PCA

Figure 1. Comparison between the two dimensional embeddings of PCA and Isomap on the teapot data set.

apply gradient descent (for instance) to minimize the objective. The point x^* reached at convergence would be considered to be a good approximation of a pre-image of y.

However, in practice, the map ϕ between the input space and the feature space is *a priori* not known. Consequently, given a new training point x, it is impossible to compute either its embedding $\phi(x)$ or the gradient of a function involving $\phi(x)$. These remarks motivate the need for an interpolation method not relying on the mapping function ϕ, which we detail in the next section.

2.3. The method

For an illustrated explanation of the method, please refer to Figure 2.

2.3.1. Mapping the dataset to a low-dimensional space

We use Isomap to project each image from the original video to the low-dimensional space, which is also called *feature space*. In that space, the data points $\left(\phi(x^{(i)})\right)_{k\in[\![1;n]\!]}$ are represented by their coordinates and by their timestamp in the video (which corresponds to the index i).

2.3.2. Interpolating a curve between the data points in the feature space

We interpolate a smooth curve between the points in the low-dimensional space. The curve equation is given by $t \mapsto \alpha(t)$ and is constrained to go through the m data points, *i.e.* $\exists(t_1,\dots,t_m) \in \mathbb{R}^m$ s.t. $\forall i \in [\![1;m]\!]$, $\alpha(t_i) = \phi(x^{(i)})$, while minimizing the curvature of the curve. This optimization problem can be solved by a regression (Dam et al., 1998).

2.3.3. Location of the interpolated images in the feature space

Given 2 images $x^{(i)}$ and $x^{(i+1)}$, we want to insert n images $\left(p^{(i,k)}\right)_{k\in[\![1;n]\!]}$ between them. Let us denote by $\left(d^{(i,k)}\right)_{k\in[\![1;n]\!]}$ the n points *in the feature space* evenly spaced on the curve linking the two consecutive images $\phi(x^{(i)})$ and $\phi(x^{(i+1)})$. More formally, if $\alpha^{(i)}$ is the truncation of α between $\phi(x^{(i)})$ and $\phi(x^{(i+1)})$ such that $\alpha^{(i)}(0) = \phi(x^{(i)})$ and $\alpha^{(i)}(1) = \phi(x^{(i+1)})$, we have, for any $k \in [\![1;n]\!]$, $\alpha^{(i)}\left(\frac{k}{n+1}\right) = d^{(i,k)}$. We consider the $(d^{(i,k)})_{k\in[\![1;n]\!]}$ to be the embedding of the interpolated images. As we discussed in section 2.2, it is not possible to find preimage of those points. As a result, we next detail an alternative method to find a reasonable interpolation in the input space.

2.3.4. Inserting new images

To address the problem of the preimage approximation, we choose the image corresponding to $d^{(i,k)}$ to be its nearest neighbor in the low-dimensional space. Specifically, for each $d^{(i,k)}$, we seek for the data point $\phi(x^{(j)})$ that minimizes the Euclidean distance between the two of them, and we then set $p^{(i,k)} = x^{(j)}$. In other words, if we have $u^{(i,k)} := \arg\min_{1\le j\le m} \left\|d^{(i,k)} - \phi(x^{(j)})\right\|_2$, then we set $p^{(i,k)} := x^{(u^{(i,k)})}$.

2.3.5. Final result

Each pair of consecutive original images $(x^{(i)}, x^{(i+1)})$ is now augmented with other images from the dataset as follows: $(x^{(i)}, x^{(u^{(i,1)})}, \dots, x^{(u^{(i,n)})}, x^{(i+1)})$.

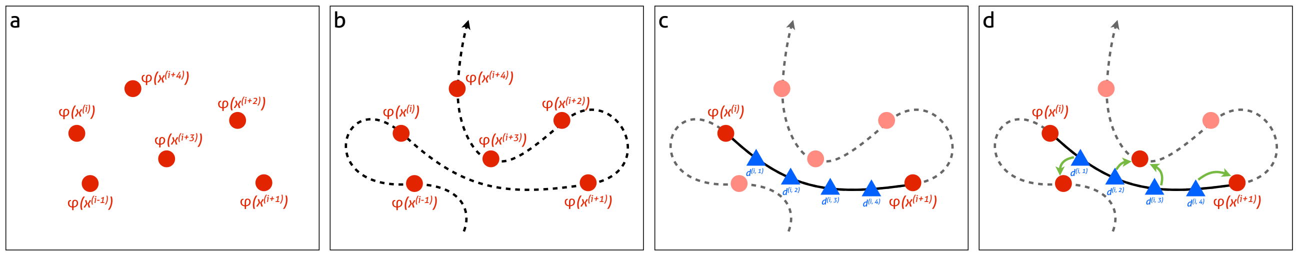


Figure 2. Our interpolation method: (a) We project the images from the video on the low-dimensional space (LDS) with Isomap. (b) In the LDS, we interpolate a smooth curve between the embedded data points. (c) In the LDS, we estimate the location of the n images we want to insert $(d^{(i,k)}, \dots, d^{(i,n)})$ by placing them evenly on the curve between $\phi(x^{(i)})$ and $\phi(x^{(i+1)})$. (d) We set each of the interpolated images to be equal to the closest point from the original data.

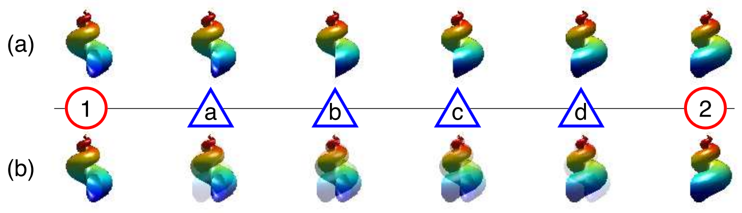


Figure 3. Comparison between two interpolation methods: (a): the method we discussed in section 2.3.4. (b): weighted average interpolation between the two images. The red circles (two extremal images) are the images between which we want to insert new images. The blue triangles are the inserted images.

3. Experiment

3.1. Protocol

To test the validity of our method, we used the following protocol:

1. From an original video, create a compressed version of the video by removing every other image.
2. Interpolate frames between each pair of consecutive images in the compressed video.
3. Compare our interpolated frames with the frames deleted in step 1.

3.2. Measuring the performance of our method

3.2.1. Two other interpolation methods

In order to measure the performance of our method, we confronted it with 2 other interpolation methods:

- averaging the pixel values of the adjacent images
- Motion Estimation - Motion Compensation (MEMC), a common interpolation method used for instance in the MPEG4/H.263 standard. More specifically, we used the toolbox discussed in (Barjatya, 2004), which consists in two steps:

1. Motion estimation: this task is performed by a block-matching algorithm (we mainly used extensive search). The latter produces one motion vector for each block in the image (a block being a square of pixels).
2. Motion compensation: using the motion vectors found in the previous step, the algorithm interpolates an intermediary image by translating blocks of pixels according to the motion vector

3.2.2. Assessing the performance of each interpolation method

To compare the similarity between the image that was deleted during compression and the interpolated image, we implemented two comparison procedures.

Procedure 1, based on Hough Transform
We first derived a comparison method from the Hough transform (Duda & Hart, 1972). Specifically, given a discretization $(\rho_k, \theta_l)_{(k,l)} \in \Delta$ of the Hough space, we computed the simple Hough transform of both the original $HT^{(d)}$ and the interpolated images $HT^{(i)}$. We then defined the Hough error ϵ to be:

$$\epsilon = \sum_{(k,l)\in\Delta} \left(HT^{(d)}(\rho_k, \theta_l) - HT^{(i)}(\rho_k, \theta_l) \right)^2 .$$

Overall, this method consists in comparing the set of edges of the two images.

Procedure 2, based on Scale-Invariant Feature Transform
Scale-Invariant Feature Transform (SIFT) (Lowe, 2004) is usually used to detect the presence of objects in images. It computes keypoints which are represented by *descriptors* and is able to match key points from an image to those of another image (see Figure 4). To come with a measure of similarity using this algorithm, we combined two criteria:

- The number of matches between the descriptors of the two images[2]. This measures how similar the two objects are.
- The difference in position between the matching descriptors of either image. This allows us to know if the object is in the right position (location and orientation) and gives an idea of how the interpolated image respects the flow of the video.

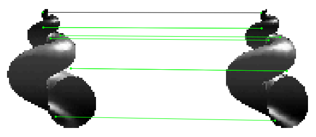


Figure 4. Matching keypoints between the interpolated image and the deleted image using SIFT.

3.3. The datasets

Finally, we decided to compare the performance of the 3 interpolation methods on 2 different datasets:

- For the first one, we built a video by rotating a 3D shell. The frame rate and the revolving speed are slightly desynchronized so that each rotation is different from the previous one. This video is an example of a repetitive footage where similar actions are seen at different points in the video.
- For the second one, we used a CIPR sequence representing a toy train going from right to left, a board with an upwards movement, and 2 spinning atoms. This video is therefore much more complex that the previous one. In particular there is no repetitive movement.

[2]Descriptors are usually represented as a 128-dimensional features, which ensures that two matching keypoints are actually referring to the same element with good probability.

4. Results

4.1. Shell dataset (video with repetitive motion)

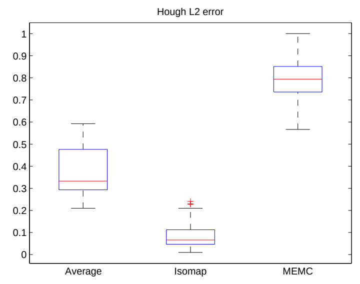
- The Hough error estimation indicates that our method using Isomap performs significantly better than the two others on the shell dataset (Figure 5-(a)).
- The SIFT error estimation was not exploitable due to the number of outliers and too small differences in descriptors positions, probably because some images do not have enough key points.
- For each of the 300 interpolated images, we found that the minimum error was given by Isomap.

4.2. Caltrain dataset (video without repetitive motion)

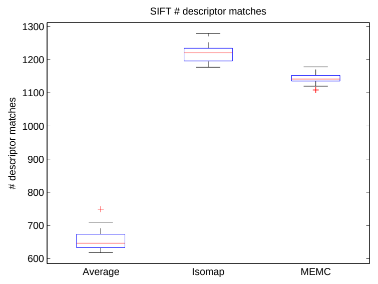
- The Hough error estimation indicates that the average interpolation performs significantly better than the two others. The method using Isomap has no significant difference with the MEMC interpolation method.
- The SIFT error estimation indicates that the method with Isomap has significantly more descriptor matches than the two other methods (see Figure 5-(b)). This shows that the Isomap interpolation method is better at preserving the original shape of the object. Isomap and MEMC also perform significantly better than the average interpolation in the position change of the matching descriptors (see figure 5-(c)). However, there is no significant difference between Isomap and MEMC on that aspect.

4.3. Videos

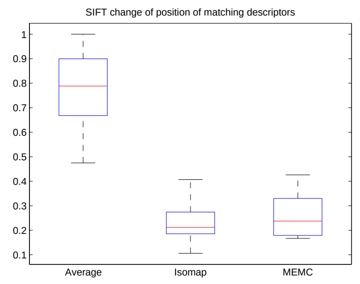
You can see the videos at the following URL: `http://stanford.edu/~robinio/cs229project/`. On the shell dataset, see how the interpolation with Isomap yields a video which is very close to the original video.



(a) Hough error - Shell dataset



(b) SIFT number of matches - Caltrain dataset



(c) SIFT change in position of key points - Caltrain dataset

Figure 5. Comparison of the results of 3 interpolation methods using the procedures described in Section 3.2.2. On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually.

5. Conclusion

Our results suggest that our method is particularly efficient on footages with repetitive motion (for instance, chen the camera is rotating around an object, or when a person is waving his hand at someone, or if the camera is shooting a closeup of a flying bird). It has several advantages:

- The interpolated images have the same quality as the rest of the video, which often results in a much sharper video
- In the worst case scenario, the effective frame rate of the decoded video is the same as the compressed video (the interpolated frames between $x^{(i)}$ and $x^{(i+1)}$ will be equal to $x^{(i)}$ or $x^{(i+1)}$).

6. Limitations & further work

The main limitation of our method is that we only use images from the data set. As a consequence, the method works only for videos that contain a repetitive motion. An interesting improvement of the method would be either to find a way of computing preimages with Isomap or to use a nonlinear dimensionality reduction algorithm that is invertible (or pseudo invertible). Indeed, we would then be able to create images from scratch, which would probably result in a systematic increase the effective frame rate of the video.

Another direction of research would be to find a way to combine our method with another interpolation method such as multi-reference MEMC. The principal challenge would then be to find a systematic way of deciding which algorithm generates which images.

References

Barjatya, Aroh. Block matching algorithms for motion estimation. *IEEE*, 2004.

Dam, Erik B., Koch, Martin, and Lillholm, Martin. Quaternions, interpolation and animation. Technical report, 1998.

Duda, Richard O. and Hart, Peter E. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972. ISSN 0001-0782. doi: 10.1145/361237.361242.

Lowe, David G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2): 91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94.

Pless, Robert. Image spaces and video trajectories: Using isomap to explore video sequences. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pp. 1433–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1950-4.

Tenenbaum, Joshua B., de Silva, Vin, and Langford, John C. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500): 2319–2323, December 2000.