Nonlinear dimension reduction on motion capture data

Daniel Perry

Overview

0.1 Summary

In the field of computer animation, animating characters quickly and effectively can be a challenge. The simplest approach is to key frame the characters motion and then allow the computer to interpolate the in-between frames. However, key framing can be tedious and time consuming.

One approach that attempts to alleviaate some of the manual posing and key framing, is to use motion capture data to record positions and movement in the real world, and then apply that data to a computer animation. However using motion capture data can be somewhat restrictive - using motion capture data without modification restricts the characters movements to those already captured. Capturing new motion can also be expensive.

There has been a substantial amount of work in morphing, connecting, and otherwise modifying motion capture data to provide more flexibility and reduce the cost of using motion capture data.

Another vein of work has investigated generating new motion through physical simulation. In that situation a model of motion is constructed with physical realism implied through constraints on the motion. The choice of a specific motion is made by formulating the goals of the motion as a constrained optimization problem.

In general when optimizing a function, unless the problem has special structure that can be used to find the solution, it's beneficial to either have a very small spece in which to search or to have a very good guess as to the solution (or both!). In character motion the optmization space can be substantial - human motion often has upwards of 60 degrees of freedom.

This means that in order to use a general optmization technique the animator either needs to provide a very good guess for the final optimum or constrain the degrees of freedom substantially. The first option is undesirable, because this places the burden of providing a good initial guess on the animator, which can be time consuming. The second option seems preferable, assuming it can be done in a reasonable way.

One interesting approach to constraining the character motion optimization problem was proposed by [7], where motion capture data and PCA is used to restrict the motion to a smaller but useful subspace.

However [7] restricted their dimension reduction to a linear method, while character motion is better described as nonlinear. Further, specific character motion, like walking, running, jumping, etc. appears to lie on a lower-dimensional manifold within all possible motion for a given model. This seems to indicate that a non-linear dimension reduction such as manifold learning or kernel PCA could reduce the dimension further and provide substantial benefits.

0.2 Contribution

Here I present some initial investigation into both linear and non-linear dimension reduction on motion capture data.

I first present some results similar to those shown in [7] in dimension reduction. I then show two approaches to non-linear dimension reduction that could be used, and some initial results.

1 Project

1.1 PCA

Because I'd like to extend the approach suggested in [7], I've attempted to duplicate some of their results.

The first component of that is to reduce the dimension of the motion data using PCA.

PCA is a linear dimension reduction technique that computes the subspace that maximizes the variance in the data.

This can be done a number of ways, but is generally done using the SVD for systems that aren't too large. For those that are too large, iterative methods can be used, such as the power method.

In order to capture the general subspace of a specfic motion, [7] recommends computing the subspace on a set of motions.

I did this by bringing in n different motions, $M_{i i=1}^{n}$, where each motion consists of m poses of d dimension - where d is dictated by the degrees of freedom of the character model:

$$m_i = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{pmatrix}$$

where

$p_j = \big(b_1, \ldots, b_l\big)$

and b_k are the local orientation of the corresponding bone.

A single motion capture data can then be represented by a single matrix with dimensions $m \times d$, $d = 3l$, if each bone's orientation is represented with euler angles.

The motion matrices are combined by contatenating them together, so that the set of motion data is $nm \times d$:

$$M = \begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix}.$$

The PCA is then take of the full matrix M. This is normally done by computing the SVD of $M = U\Sigma V^T$, and then using the first r columns of V as the basis vectors the subspace: To get a low dimensional representation y_i of a single motion pose p_i, the data is projected onto the PCA subspace:

$y_i = V^T p_i$

However these low dimensional pose vectors can't be used to pose a model, so they are reprojected back into pose space:

$\tilde{p}_i = V y_i = V V^T p_i$

Note that if the full V was used $VV^T = I$, and we would recover the original pose.

For the examples shown in the accompanying movie, I imported 8 similar walking videos from the CMU Database, and computed the PCA subspace from those examples.

I then projected each of the example data sets onto the selected PCA subspace and then reprojected back onto pose space. As the movie shows when the subspace is too small, the resulting motion is severely restricted.

1.2 Manifold Learning

One appealing approach is to use Manifold Learning to reduce the dimension of the motion capture data.

Motion captures data appears to lie on a manifold, and when focused on a single motion type - like walking, running, or jumping - it seems like the dimension could become quite small.

However there are two significant limitations to manifold learning - one is that generalizing the transformations to data points outside of the original data set is non-obvious [4]. Another is that computing the pre-image of a data point after transformation is complicated.

For this project I implemented the Isomap manfold learning method and ran it on the motion matrix, M, described above. However I had difficulty in finding a precise description of how to compute the pre-image of a point in Isomap. This might be possible by framing the manifold learning technique as Kernel PCA

(see below) and then using a pre-image technique from that body of literator. However some details are still fuzzy (as the pre-imag methods used below a specfic to the Gaussian and related kernels).

Because of these difficulties, I postpone the application of these methods to future work.

1.3 Kernel PCA

Kernel-based PCA is another form of PCA that attempts to find the non-linear relationship of the data by projecting into a higher dimensional space.

Most manifold learning techniques can posed as a Kernel PCA problem, by defining the kernel appropriately [3].

However, more generally Kernel PCA involves selecting a specific kernel $K(x_1, x_2)$ that corresponds with a function $\phi(x)$, such that $K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$. $\phi(x)$ can be thought of as lifting the point x into a higher dimension space, with the hope that non-linear relationships in input space become linear in the higher dimensional space.

Kernel PCA consists of performing PCA on the data set $\Phi(X) = \begin{pmatrix} \phi(x_1) \\ \vdots \\ \phi(x_n) \end{pmatrix}$, to find the sub-space that maximizes the variance of the data in the higher dimensions. In other words we are now analyzing $\Phi(X)^T \Phi(X)$ instead of $X^T X$.

One drawback of Kernel PCA is that sometimes it can be difficult to select the right kernel and even the right kernel parameters.

In the experiments shown in the video, I used the Gaussian kernel with $\sigma = 10$. I attempted projecting onto a non-linear subspace of dimension 10 and 20, however neither result appears very good visually. I assume this is because I need to find the appropriate kernel parameters or possibly need to use a different kernel function.

The Gaussian kernel is a common kernel choice because of it's many nice properties and has an intuitive interpretation of it's effect on neighborhoods. For this project, the choice of Gaussian was also beneficial because of work done on the pre-image problem specifically for the Gaussian kernel [6].

1.4 Implementation

The motion capture data data used here was downloaded from the CMU motion capture database [1]. To read their data formats I downloaded and used their data structures and data readers provided in the ASF/AMC viewer [5]. I also used a modified version of their viewer for the visualization in the accompanying movie, as shown in 1.

Once the motion data is loaded I convert it into a matrix form and use a popular matrix library [2] for the actual analysis. Once the analysis is done, I convert the motion data back into the motion format and use their same ASF/AMC viwer library for writing out the AMC files.

2 Conclusion

In the accompanying video, I've demonstrated some results of using a linear dimension reduction technique on motion capture data.

Additionally, I've shown some initial attempts at using a non-linear dimension reduction technique, Kernel PCA, to compute some lower dimensional embedding of the data.

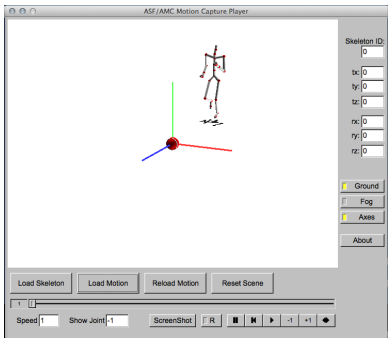

Figure 1: User interface in mocapPlayer.

References

[1] Cmu motion capture database. [http://mocap.cs.cmu.edu/ Online; accessed 20-October-2014].

[2] Eigen: C++ template library for linear algebra. http://eigen.tuxfamily.org/index.php?title=Main_Page [Online; accessed 20-October-2014].

[3] Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux, Jean-François Paiement, Pascal Vincent, and Marie Ouimet. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation*, 16(10):2197–2219, 2004.

[4] Yoshua Bengio, Jean-François Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, and Marie Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. *Advances in neural information processing systems*, 16:177–184, 2004.

[5] Yili Zhao James McCann, Jernej Barbic. Asf/amc viewer: motionplayer. [http://mocap.cs.cmu.edu/tools.php Online; accessed 20-October-2014].

[6] JT-Y Kwok and Ivor W Tsang. The pre-image problem in kernel methods. *Neural Networks, IEEE Transactions on*, 15(6):1517–1525, 2004.

[7] Alla Safonova, Jessica K Hodgins, and Nancy S Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (TOG)*, 23(3):514–521, 2004.