Some recommended papers

1. Develop the methods which realistically model body shape using 3D scans of many bodies and many poses.

Reference:

HIRSHBERG, D. A., LOPER, M., RACHLIN, E., AND BLACK, M. J. 2012. Coregistration: Simultaneous alignment and modeling of articulated 3D shape. In ECCV, vol. 7577 of LNCS. Springer, 242–255.

At <https://ps.is.tuebingen.mpg.de/research_projects/3d-mesh-registration>

CHEN, Y., LIU, Z., AND ZHANG, Z. 2013. Tensor-based human body modeling. In CVPR, 105–112.

At <http://research.microsoft.com/apps/pubs/default.aspx?id=196145>

The Stitched Puppet: A Graphical Model of 3D Human Shape and Pose, in CVPR 2015

At <http://stitch.is.tue.mpg.de/>

L. Sigal, M. Isard, H. Haussecker, and M. J. Black. Loose limbed people: Estimating 3D human pose and motion using non-parametric belief propagation. International Journal of Computer Vision, 98:15–48, 2011

1. Develop the methods which automatically extract proxy bones and corresponding skinning weights from a set of example poses.

SIGGRAPH Course 2014 — Skinning: Real-time Shape Deformation: Mesh Animation Decomposition and Compression, Binh H. Le and Zhigang Deng, University of Houston

At <http://skinning.org/decomposition-methods.pdf>

Doug L. James and Christopher D. Twigg. Skinning mesh animations. ACM Transactions on Graphics (SIGGRAPH 2005), 24(3)

At <http://graphics.cs.cmu.edu/projects/sma/>

1. Develop the methods which capture motion of body based on markers and capture surface deformation at high spatial and temporal resolutions.

PARK, S. I., AND HODGINS, J. K. 2008. Data-driven modeling of skin and muscle deformation. ACM Trans. Graph. 27, 3 (Aug.), 96:1–96:6.

At <http://dasan.sejong.ac.kr/~sipark/papers/park_s2008.pdf>

Pons-Moll, Gerard and Romero, Javier and Mahmood, Naureen and Black, Michael J., Dyna: A Model of Dynamic Human Shape in Motion, ACM Transactions on Graphics, 34(4), page:120:1-120:14, (Proc. SIGGRAPH2015)

At <http://dyna.is.tue.mpg.de/>

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. SCAPE: Shape Completion and Animation of PEople. ACM Trans. Graph. 24, 3, 408–416.

At <http://robotics.stanford.edu/~drago/Projects/scape/scape.html>

1. Develop the methods for stylistic human motion editing which can automatically transform unlabelled heterogeneous motion data into new styles.

Realtime Style Transfer for Unlabeled Heterogeneous Human Motion, ACM Transactions on Graphics, 34(4): Article 119, 2015.

At <http://faculty.cs.tamu.edu/jchai/projects/SIG15/style-final.pdf>

M. Abdul-Massih, I. Yoo and B. Benes, Motion Style Retargeting to Characters With Different Morphologies, COMPUTER GRAPHICS forum, Volume 00 (2016), number 0 pp. 1–14

At <http://onlinelibrary.wiley.com/doi/10.1111/cgf.12860/epdf>

Dance annotation

1. LabAnotation (<http://dancenotation.org/>)

Okan Arikan, David A. Forsyth, and James F. O'Brien. "Motion Synthesis from Anotations". In Proceedings of ACM SIGGRAPH 2003, pages 402–408

At <http://graphics.berkeley.edu/papers/Arikan-MSF-2003-08/>

Ramanan, D., Forsyth, D. A. "Automatic Annotation of Everyday Movements." Neural Info. Proc. Systems (NIPS), Vancouver, Canada, Dec 2003

At <http://www.cs.cmu.edu/~deva/papers/annotations/index.html>

GenLaban: A tool for generating Labanotation from motion capture data, Multimedia Tools and Applications, Volume 74, Issue 23, pp 10823-10846, 2015

A Labanotation Editing Tool for Description and Reproduction of Stylized Traditional Dance Body Motion (<http://dh2011abstracts.stanford.edu/xtf/view?docId=tei/ab-250.xml;query=;brand=default>)

Laban-writer download at <https://dance.osu.edu/research/dnb/laban-writer>

Meinard Müller, Andreas Baak, Hans-Peter Seidel, Efficient and Robust Annotation of Motion Capture Data, ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2009

At <http://resources.mpi-inf.mpg.de/MocapAnnotation/>

Workshop 1 on 17 June 2016

Khiem Nguyen Minh presented the paper of “SCAPE: shape completion and animation of people”.

The proposed challenges include, (1) source codes are not available on internet; (2) implementation is hard.

Phuong Bui Dang Ha presented the paper of “Tensor based human body modelling”.

The proposed challenges include, (1) source codes are not available;

Tran Nguyen Duong Chi presented…….

Workshop 2 on 24 June 2016

Khiem Nguyen Minh presented the paper of “Dyna: a model of dynamic human shape in motion”.

The proposed goal is to apply Dyna to facial expression reconstruction based on 3D range data. Currently, he will program Dyna algorithm and run it on the provided test data for verification.

Phuong Bui Dang Ha presented the paper of “the stitched puppet: a graphical model of 3D human shape and pose”.

The proposed challenges include, (1) how to partition the incomplete 3D data; (2) coding. She will program this algorithm at first, and then consider how to match incomplete 3D data.

Tran Nguyen Duong Chi presented the paper of “Motion Synthesis from Anotations”.

The proposed goal is to apply this annotation algorithm to 3D motion data. She will program this algorithm at first.

Workshop 3 on 1st July 2016

Khiem Nguyen Minh is stuck at some VTK lib.

Phuong Bui Dang Ha needs to read another paper related part-based modelling method.

Tran Nguyen Duong Chi is required to implement the algorithm proposed in “Motion Synthesis from Anotations”

Workshop 4 on 15 July 2016.

Kripesh will deliver a presentation on his research.

Tran Nguyen Duong Chi queries him of using OpenCV.

**Presentation on:** Setting Kinect for windows in windows 7 to acquire RGB and Depth image sequences in parallel and a brief talk on the research work done so far on:

**Research Topic:** Computer vision based fall detection system for elder people living alone in an indoor environment.

**Presentation Summary:**

In this presentation, First, I want to talk about the settings of Kinect for windows sensor in windows 7 to obtain Depth and RGB image sequences in parallel. These images are actually used as dataset in my research work.

Second, I want to share my research work done so far: We aim to solve the problem of fall detection in three stages: (A) Detection of human silhouette, (B) Recognition of the pose, and (C) Calculation of the velocity of the body during the change in pose. At this stage, we are in the second stage of our approach and therefore this presentation will cover only initial two stages of our approach to fall detection which is human silhouette detection and identification of activities in the form of different poses.

**Abstract:**

Falls are a major health problem especially in the case of elderly. Increasing fall events demands a high quality of service and dedicated medical treatment which directly leads to economic burden; and serious injuries due to fall have even cost several lives in the absence of immediate care and support.

Therefore, a monitoring system that can accurately detect a fall event and generate an instant alert for immediate care is extremely necessary.

This research aims to develop a computer vision based fall detection system for elder people living alone in an indoor environment.

We aim to recognise a lying pose which may be considered as an after-fall pose that could be different from other normal activities such as standing, sitting, bending or crawling. Identifying a pose is important to understand a fall event where a change of pose defines its characteristic. Usually, a fall event ends up in lying pose. However, it is more difficult when fall does not end up in lying horizontally. That system which considers all the lying poses as falls can also generate a higher false alarm. The recognition of pose is based on convolutional neural network (CNN) which is implemented using both RGB and Depth images captured from Kinect Sensor. Convolutional neural networks have been very successful in the classification of activities using mainly large RGB image datasets and high-end GPU computation. We explore the fusion of RGB and Depth images feeding to a convolutional neural network in parallel with our own dataset. The inclusion of depth together with RGB can be a significant strategy as they complement each other to overcome their weakness respectively.

[Thanh proposal on 30th July 2016]

**Proposal**

**Name: Thanh Nguyen Date 30 July 2016**

This document firstly introduces an overview of AniAge project from my best knowledge. Then, I list three topics that I’m working on, including Key Frame Extraction from 2D video, Human Pose Estimation form 2D video and Human Motion Style Transfer. Hopefully, I can discuss with you about those topics.

# Project Overview

To my best knowledge, AniAge is very huge project, it can be considered as some connected topics, as shown in the figure 1 below. Please let me know If there are any problem with my figure, because I just draw it as my understanding.

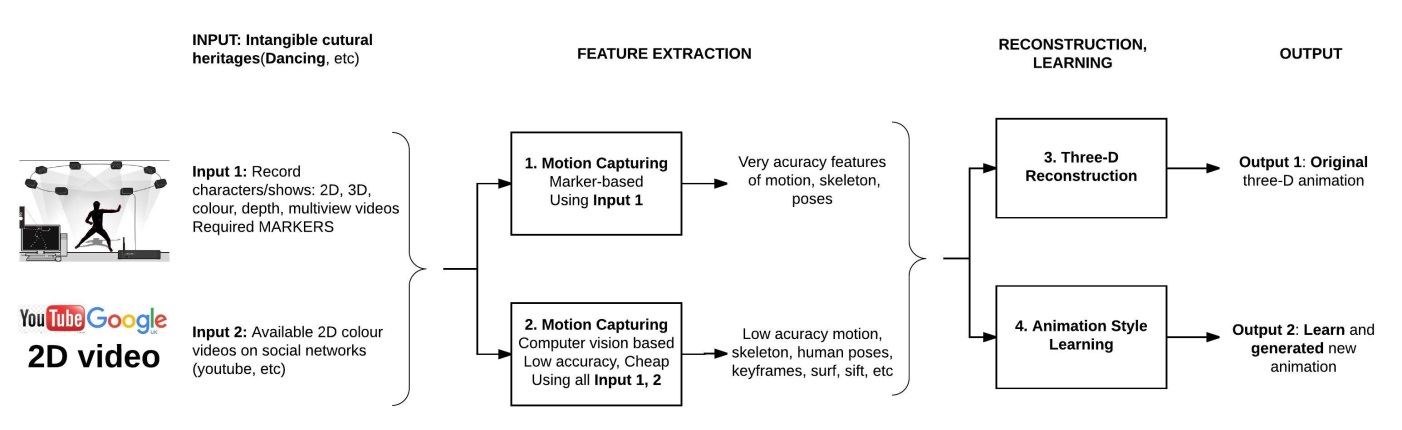


Figure 1. AniAge Overview

In details, AniAge tends to reconstruct animation, learn animation and generate new animation of intangible cultural heritages, such as dancing, etc.

* INPUT: There are two major input types. Input 1 consists of 3D, color + depth, multi-view videos, those kind of videos require manually setup and record characters with equipment (marker-based, multi-camera, stereo camera, depth camera, etc.). Input 2 includes available 2D color video shared on internet (google, YouTube).
* FEATURE EXRACTION (motion, skeleton, pose, key frame, etc.): Using input 1 requires expensive setup and equipment, but extracted features are very accuracy. Using input 2 is cheaper, but requires apply difficult feature extraction techniques, especially low accuracy. In short, using Input 1 is expensive and high accuracy; using Input 2 is cheap and low accuracy.
* RECONSTRUCTION: This phase reconstruct original animation based on extracted features.
* STYLE LEARNING: This phase learn animation from extracted features, then model them as style, and generated new animation.

# Human Pose Estimation

In this topic, I'm trying to estimate human poses/skeleton from 2D color videos. I'm currently following Max-covering method in this paper [1]:

http://www.cs.bc.edu/~hjiang/details/maxcovering/index.html

Figure 2 below shows the estimated pose from a 2D input image using Max-covering method. I have already configured Max-covering source code successfully, and reading their algorithms.

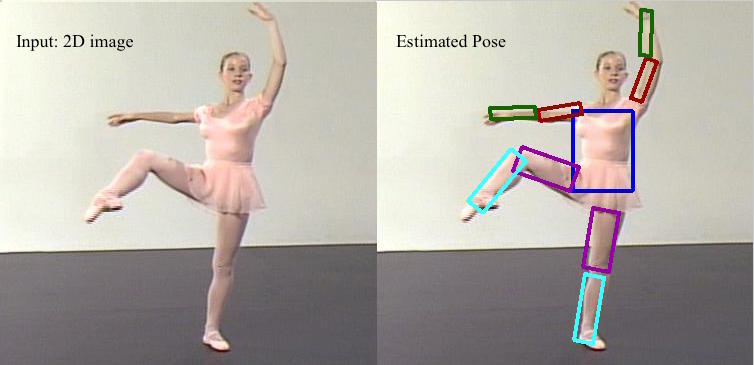


Figure 2. Estimated Pose

# Key Frame Extraction

In this topic, I still use 2D video as input to extract key frames. The key frames can be considered as one feature for the next Stage of Learning. So far, I've already implemented two key frame extraction techniques, includes histogram-based and motion-based, some result shown below, motion-based seems better so far:

In terms of histogram-based, I calculate histogram different between all consecutive frames, then consider key frames as frames that have biggest histogram diff.

In terms of motion-based idea, I estimate sum of motion vectors in each frame, then chose local minimum motion as key frames. The idea is character usually change motion direction between actions, and we capture those changes. Extracted key frames:

https://www.dropbox.com/s/ht0jioj6a8n3zdd/SmoothMotionSummary.avi?dl=0

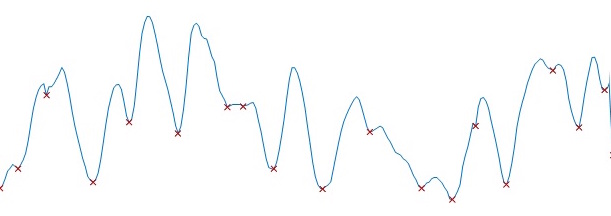


Figure 3 Motion-based method: local minimum selection

# Human Motion Style Transfer

I'm very interested in style learning and generate new animation. There are two related topic papers that I found in yours suggested papers, they are:

* **Real-time Style Transfer for Unlabeled Heterogeneous Human Motion**, ACM Transactions on Graphics, 34(4): Article 119, 2015. [2]

This paper’s objective quite similar with our project, it is transferring human motion styles. Their key idea is using **online learning** algorithm that automatically constructs a series of local mixtures of autoregressive models (MAR) to capture the complex relationships between styles of motion. They construct local MAR models **on the fly** by searching for the closest examples of each input pose in the database. Once the model parameters are estimated from the training data, the model adapts the current pose with simple linear transformations.

Compared to two previous works [4] [5], their two strong points are: First, this method can deal with heterogeneous human motions while others can only deal with some fixed motions (walking, running, jumping); Second, previous works often require labeled data before transfer it to new style while this method can deal with unlabeled data.

* **Motion Style Retargeting to Characters with Different Morphologies**, COMPUTER GRAPHICS forum, Volume 00 (2016). [3]

I’ve just have a quick look at this paper, their strong point is transferring style in different between different morphologies. For instance, they transferred a human motion style to a dragon.

# Working Task

According to your advice, I’m working my most interesting topic of Human Motion Style Transfer. Currently, I’m dig in paper [2]. In their homepage, authors said they will share data and source, but require **a senior project manager or senior researcher**, detail below. It would be very nice if you please contact them for Source + data. Thanks.

----------------------------------------------------------------------------------------------------------------

<http://humanmotion.ict.ac.cn/papers/2015P1_StyleTransfer/details.htm>

Obtaining the data and source code

On request, we make all the algorithm relative data and source code available for scientific purposes. To obtain a copy, please send an email to XSH AT ICT DOT AC DOT CN xsh@ict.ac.cn, starting:

- 1. name, title or position, and institution or affiliation;

- 2. intended use of the data, further information about you and your work;

**Please understand that we can only provide the data to you if you are a senior project manager or senior researcher at your institution.**

---------------------------------------------------------------------------------------------------------------

# References

[1] http://www.cs.bc.edu/~hjiang/details/maxcovering/index.html

[2] Real-time Style Transfer for Unlabeled Heterogeneous Human Motion, ACM Transactions on Graphics, 34(4): Article 119, 2015.

[3] Motion Style Retargeting to Characters with Different Morphologies, COMPUTER GRAPHICS forum, Volume 00 (2016).

[4] Hsu, Eugene, Kari Pulli, and Jovan Popović. "Style translation for human motion." ACM Transactions on Graphics (TOG). Vol. 24. No. 3. ACM, 2005.

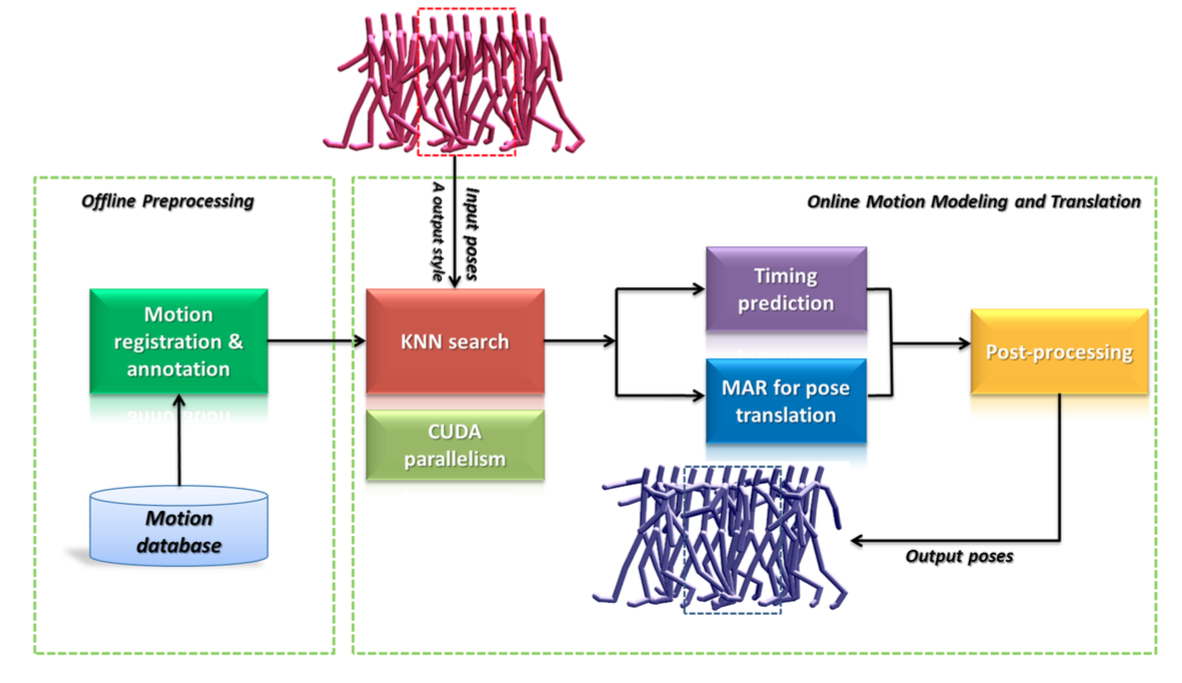
[5] Ikemoto, Leslie, Okan Arikan, and David Forsyth. "Generalizing motion edits with gaussian processes." ACM Transactions on Graphics (TOG) 28.1 (2009): 1.

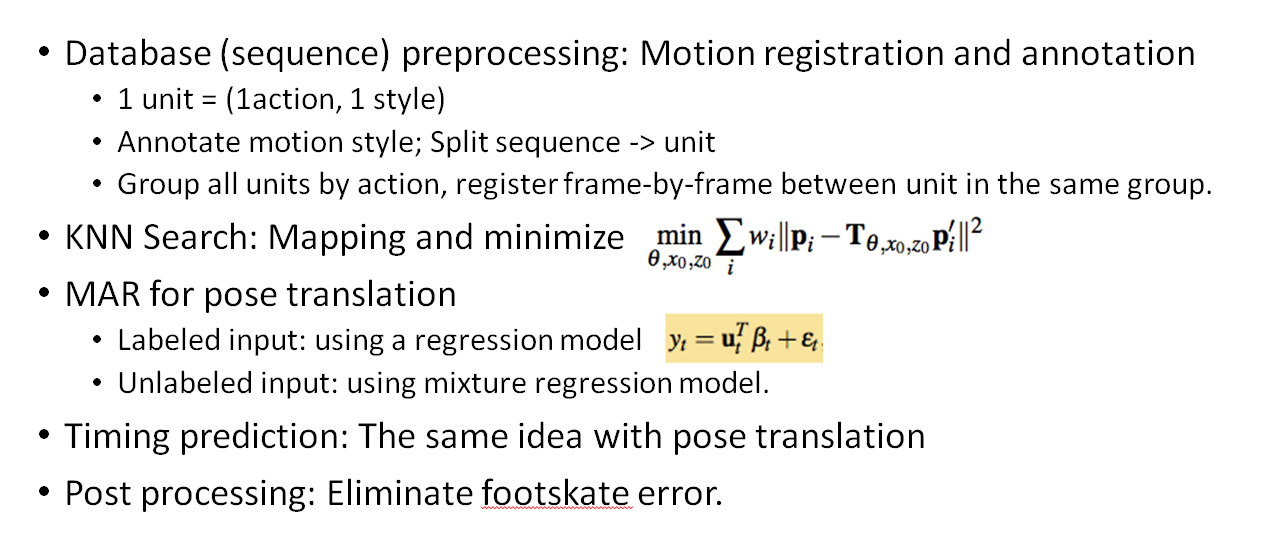
[Thanh proposal on 30th July 2016]

1. **Research Progress**

This week, I focus on Style Transfer papers [1,2,3] and the GAP of motion registration/annotation [4].

* Style Transfer based on online learning and autoregressive model (2015)





* Style Transfer using Spectral Domain (2016)

This method uses DFT to convert motion sequence to spectral domain, then extracting style in spectral domain. The method reduces effort of creating database, since it can transfer unseen action to new style.



* Annotation/registration motion

Reading paper [4].

1. **Difficulties**

I got the Local Administration for my desktop in lab, but it still not full permission. Therefore Cuda toolkit (for parallel processing) has not installed yet, “access denied” when I try to access Disk C. I requested again to IT Service Desk, hopefully they can support my account full control of my desktop soon.

1. **References**

[1] SIGGRAPH2005, Hsu, Eugene; Pulli, Kari; Popovic, Jovan -- Style translation for human motion.

[2] SIGGRAPH2015, Realtime Style Transfer for Unlabeled Heterogeneous Human Motion.

[3] SIGGRAPH2016, Spectral Style Transfer.

[4] Etemad, "Segmentation and classification of human actions and actor characteristics with 3D motion data." *International Journal of Artificial Intelligence & Applications* 3.4 (2012): 65-81.

[Yu, 20 August 2016]

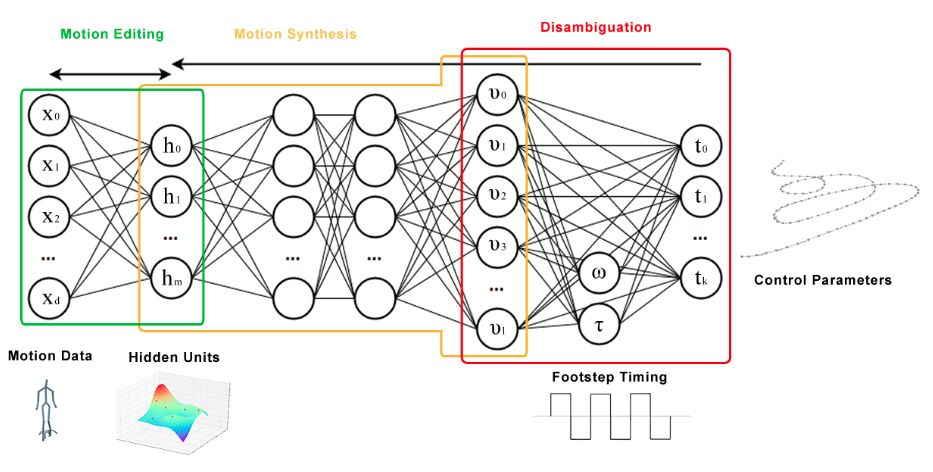
Please give out which paper you will implement by 26th August. In your report, you are required to clearly address the algorithm that you are studying on both aspects of theory and implementation, and then address your implementation in details.

[Thanh’s proposal on 2nd Sep 2016]

1. **Paper**

Holden, Daniel, et al. "A Deep Learning Framework for Character Motion Synthesis and Editing." *IEEE Transactions on Visualization and Computer Graphics* 21 (2015).

1. **Algorithm**



- Motion Synthesize:

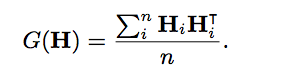
Using data from a large human motion database, a convolutional auto-encoder is trained and thus a general motion manifold is found (green box). After this training, motion can be represented by the hidden units of the network. Given this representation, a mapping is produced between high level control parameters and the hidden units via a feed-forward neural network stacked on top of the convolutional auto-encoder (orange box). The high level control parameters shown in this work are the trajectory of the character over the terrain and the movement of the end effectors. As these parameterizations can contain ambiguities, another small network is used to compute parameters which disambiguate the input (red box). Just the subset of the motion capture data relevant to the task is used to train these feed-forward networks. Using this framework, the user can produce an animation of the character walking and running by drawing a curve on the terrain, or the user can let the character punch and kick by specifying the trajectories of the hands and feet. Once a motion has been generated, it can be edited in the space of hidden units, such that the resulting motion satisfies constraints such as positional constraints for foot-skate cleanup.

* Motion style transfer:

The cost function in case of style transfer is defined by two terms relating to the content and style of the output. Given some motion data C which defines the content of the produced output, and another S which defines the style of the produced output, the cost function over hidden units H is given as the following:

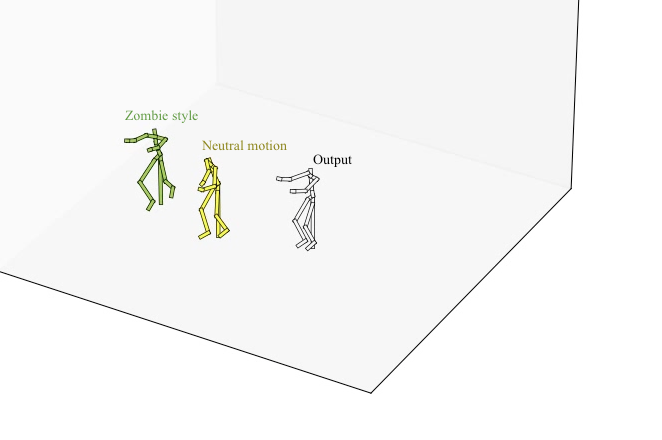
../../../../../../../../Users/hetpin/Desktop/Screen%20Shot%2020

where c and s dictate the relative importance given to content and style, which are set to 1.0 and 0.01, respectively in our experiments, and the function G computes the Gram matrix, which is the mean of the inner product of the hidden unit values across the temporal domain i and can be thought of as the average similarity or co- activation of the hidden units:



1. **Implementation**

Combining style of a zombie style motion and content of a neutral walking motion.



1. **Next week**

Continue reading paper algorithm.

[Yu’s comments, 12 September 2016]

I think your presentation is too simple to clearly introduce the implementation of the algorithm in this paper. I list a few questions below and hope you can follow them to re-draft this doc and your PPT for Friday’s workshop.

The outline of the proposed algorithm is shown in Fig.2

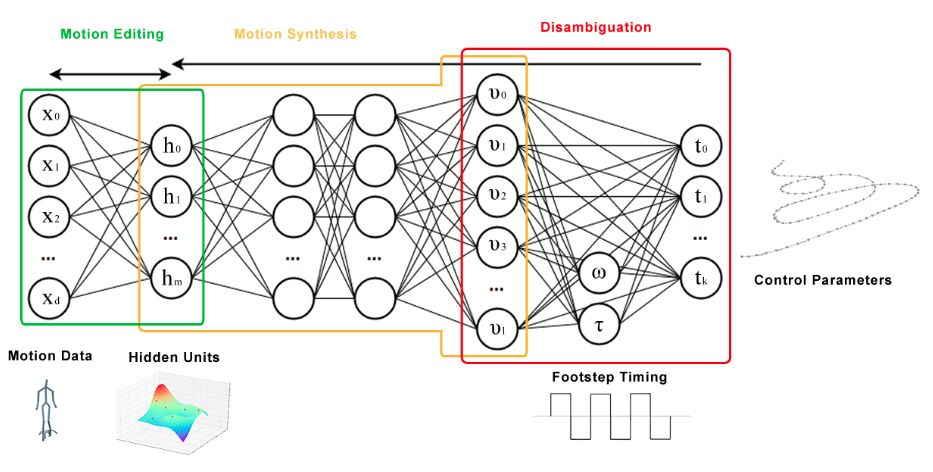
1. Data acquisition: where is the dataset from? How are they reformatted?
2. Construct Neuronal Network (autoencoder): what is the forward/backward operation? How is it constructed? How does it work? How does the training carry out?
3. Feedforward convolutional neuronal network: what is the difference compared to the above mentioned autoencoder? What is the disambiguation for locomotion?
4. (In section 7, you may list the similar questions to shape your writing), etc.

[Redraft - Thanh’s proposal on 15th Sep 2016]

**Paper**: Holden, Daniel, et al. "A Deep Learning Framework for Character Motion Synthesis and Editing." *IEEE Transactions on Visualization and Computer Graphics* 21 (2015).

# **Algorithm**

## **1.1. Overview**



Using data from a large human motion database, a convolutional auto-encoder is trained and thus a general motion manifold is found (**green box**). After this training, motion can be represented by the hidden units of the network. Given this representation, a mapping is produced between high level control parameters and the hidden units via a feed-forward neural network stacked on top of the convolutional auto-encoder (**orange box**).

## **1.2. Motion database**

**Source**: Collecting many freely available large online databases of motion capture [CMU ; Mu ̈ller et al. 2007; Ofli et al. 2013; Xia et al. 2015], as well as adding data from our internal captures.

**Normalization**: Retargeting them to a uniform skeleton structure with a single scale and the same bone lengths -> around six million frames of high quality motion capture data for a single character sampled at 120 frames per second.

**Data format for training**

* Subsample all of the motion in the database to 60 frames per second.
* The mean pose is subtracted from the data and the joint positions are divided by the standard deviation to normalize the scale of the character. The velocities, and contact labels are also divided by their own standard deviations.
* Clip motion into fixed length of n (n = 240 in experiment), overlapped by n/2 frames. Clipping is just to boost the training speed, the model can train arbitrary length.

## **1.3. Motion manifold training [Green Box]**

**Input**: Motion sequence X

**Output:** Motion manifold (network weights: theta = {wi, bi}) and hidden units H.

**Method:** Using convolutional auto-encoder to train network

- Forward operation ../../../../../../../../Users/hetpin/Desktop/Screen%20Shot%202

where:

X is input motion sequence; ReLU(x) = max(x, 0); psi max pooling function; convolutional operator \*; theta = {W, b} including weights matrix W and bias b.

- Backward operation ../../../../../../../../Users/hetpin/Desktop/Screen%20Shot%202

where:

H is hidden units; Inversed psi max pooling function; Inversed W is the weights matrix reflected on the temporal axis and transposed on the first two axes, used to inversed the convolution operation.

- Minimizing cost function using gradient descent

../../../../../../../../Users/hetpin/Desktop/Screen%20Shot%202

where: alpha is scale factor, set by 0.1 in experiment; theta = {W, b} is network

Once completed training, we have motion manifold (theta) and all hidden unit H of every X sequences: X -----(theta and phi fuction) -----> H

## **1.4. Training feed-forward network: Maps high-level para to motion**

**Input:** Parameter T (trajectory, bone length, position, etc.), is calculated directly from motion sequence X.

**Output:** A network maps T to X: phi = {w1, w2, w3, b1, b2, b3}

**Method**

* Forward operation pai maps T to X ../../../../../../../../Users/hetpin/Desktop/Screen%20Shot%202

Call network variable: phi = {w1, w2, w3, b1, b2, b3}

* Minimizing cost function using gradient descent

../../../../../../../../Users/hetpin/Desktop/Screen%20Shot%202

where: scale factor alpha; network phi; Parameter T; Motion X

Shortly, forward operation pai maps parameter T to hidden unit H; Then, inversed phi function inversed H to motion sequence Xhat; (X-Xhat)^2 is square root of difference.

## 1.5. **Disambiguation**

Mapping from high-level (low dimensional) parameters to motion will face a lot of ambiguity. For instance, a curve on a terrain alone does not give enough information to fully describe the motion that should be produced due to the ambiguity. According to authors, there is no universal solution for all ambiguity. They therefore train a model which can be used to automatically compute foot contacts from a given trajectory. **The details are unknown – reading.**

# **Applications**

## 2.1. **Motion** **synthesis**

Generating motion by define high-level parameters.

**Input**: High-level parameter T, such as trajectory, bone length, position.

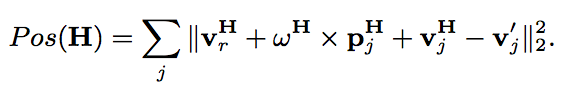
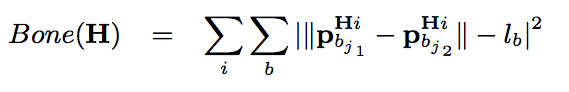
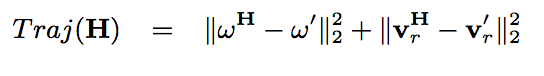
**Output**: Motion sequence X satisfied input T

**Method**

- Initializing H

- Minimizing cost function to find best hidden unit H satisfied T using gradient descent.

Cost = ../../../../../../../../Users/hetpin/Desktop/Screen%20Shot%202



- Using inversed phi function to inverse found H to motion X.

## 2.2. **Motion** **style** **transfer**

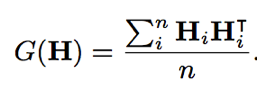
Transfer style of one motion to another motion.

**Input:** Motion XS, motion XC

**Output:** Output motion X = content of XC + style of XS

**Method**

* Using trained motion manifold to obtain hidden unit HS ,HC from XS and XC respectively.
* Define style of a motion by Gram matrix of its hidden unit



Gram matrix represents the average similarity or co-activation of the hidden units.

* Using gradient descent to minimize cost function

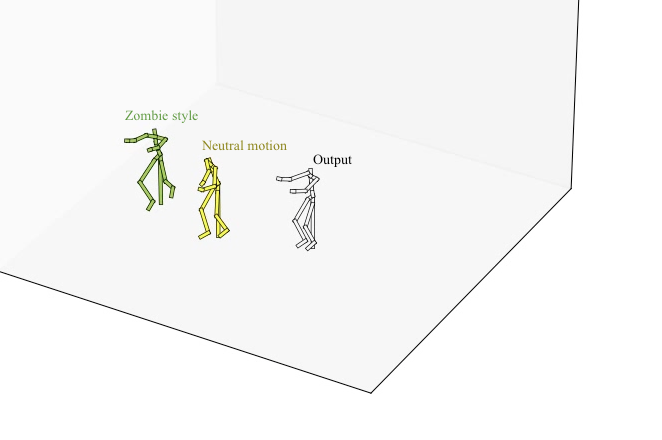
../../../../../../../../Users/hetpin/Desktop/Screen%20Shot%202

where: s and c is the coefficient of Style and Content; first square estimate the style different of Xs and X; second square estimate the content different of X and XC. Minimizing Style(H) meaning finding H that has style close to XS and content close to XC.

* Apply kinematic constraint to found H above to make motion natural.
* Finally, using backward operation of trained motion manifold to inverse the found H to motion sequence X.

# **Implementation**

Combining style of a zombie style motion and content of a neutral walking motion.



[Yu’s comment 7 November 2016]

Apply the following PCA to the pooling step of CNN network.

Given *n* images by size *h*x*w*, convert each image into a vector form with length *m*=*h*x*w*.

This forms a matrix . (if need, )

Procedure: seeking for the *k*th eigenvector ,

1. Initialize the unit
2. Compute
3. Let , update,
4. Compute ; if the function no longer increases, stop iteration and output
5. Else go to step (2).

Algorithm Greedy search

Let

For *t*=1 to *k*

In order to find , apply the above Procedure to

End

The resulting *k* eigenvectors span an eigen subspace . For a sample x, the projection is,

For the convolution layer, it is likely to apply one filter to generate a set of feature maps in an iterative way, that is, there is only one W to be trained by the training dataset.

Where each term corresponds to their individual intermediate maps. For example, corresponds to the feature map 1; corresponds to the feature map 2; and so on.

For input matrix *X*, applying 2D FFT to it yields,

Then, apply fftshift to Y so as to shift 0-frequency components to centre of spectrum, which help us to design band pass filter.

On frequency domain, the different band pass filters are determined by setting the range of (u,v) coordinates, which should cover the whole frequency domain.

Apply ifftshift to separately so as to recover the order to the original one.

After that, apply inverse FFT to respectively so as to yields,

Which is namely feature map.

For input matrix *X*, applying 2D FFT to it yields,

Then, apply fftshift to *Y* so as to shift 0-frequency components to centre of spectrum, which help us to design band pass filter.

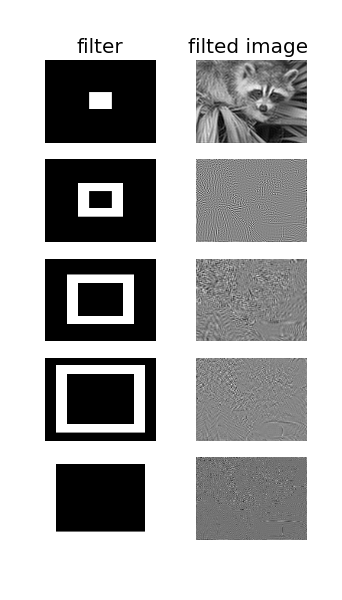
On frequency domain, the different band pass filters are determined by setting the range of (u,v) coordinates, which should cover the whole frequency domain.

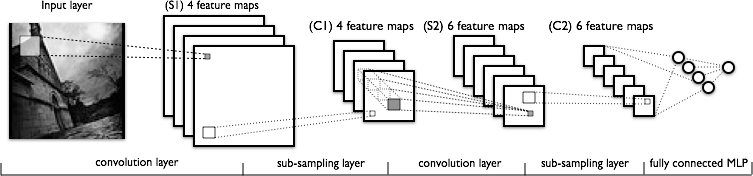
Apply ifftshift to separately so as to recover the order to the original.

After that, apply inverse FFT to respectively so as to yields,

Which is namely feature map.

Currently, the best filter set I get F=[0,0.04,0.09,0.15,0.4,0.7,1], each F[i] represents the a ratio of image size, as showed in the following picture.

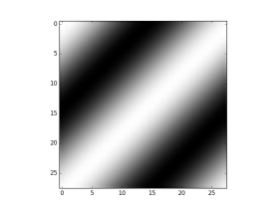
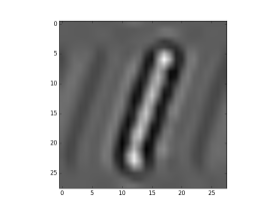
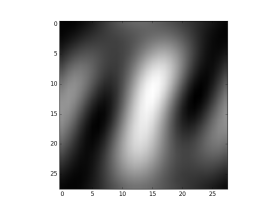
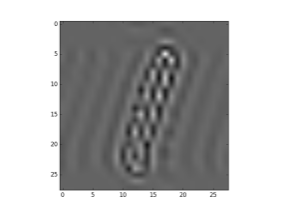


The convolution used in the original LeNet model, as showed below.

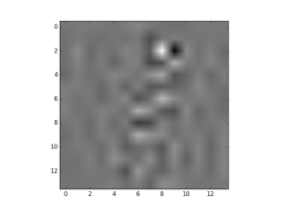
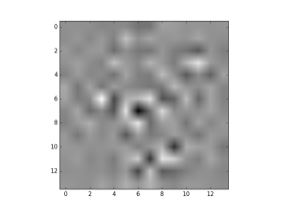
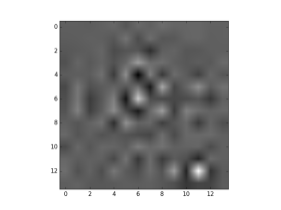
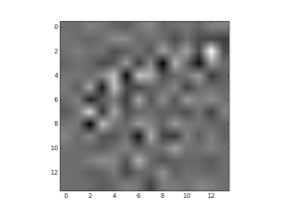
We select classifying MNIST Dataset to train the 4 layers. From the dataset we get [train\_X, train\_Y], train\_X = [Image\_batch\_size, image\_hight,image\_width] .

Since the test is so time consuming , right now we select 50 batches.

From train\_X we can get Layer0 FFT outputs( bandpass from low to high ), and in this selected batch, train\_Y should equal to 1:

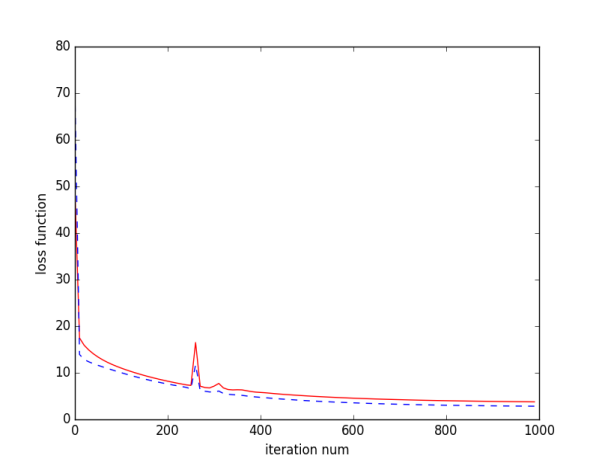
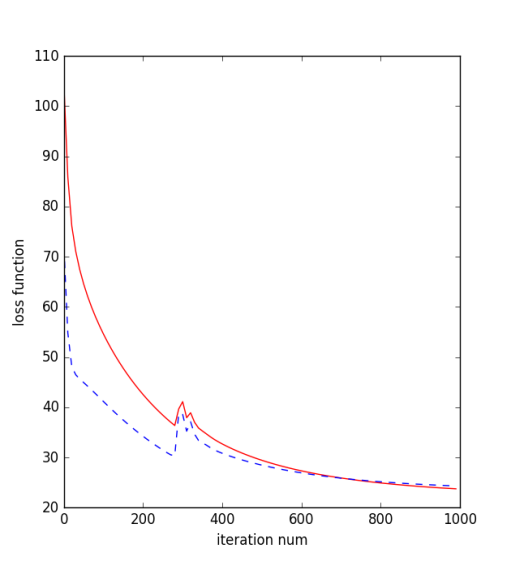
  

The output of layer0 would be processed by maxpooling function, then transported to the input of layer1 . The following pictures is showing the input of layer1 in same bandpass but different batch. As we can see the scale of the images is down sample to [14,14] while the origin size is [28,28].



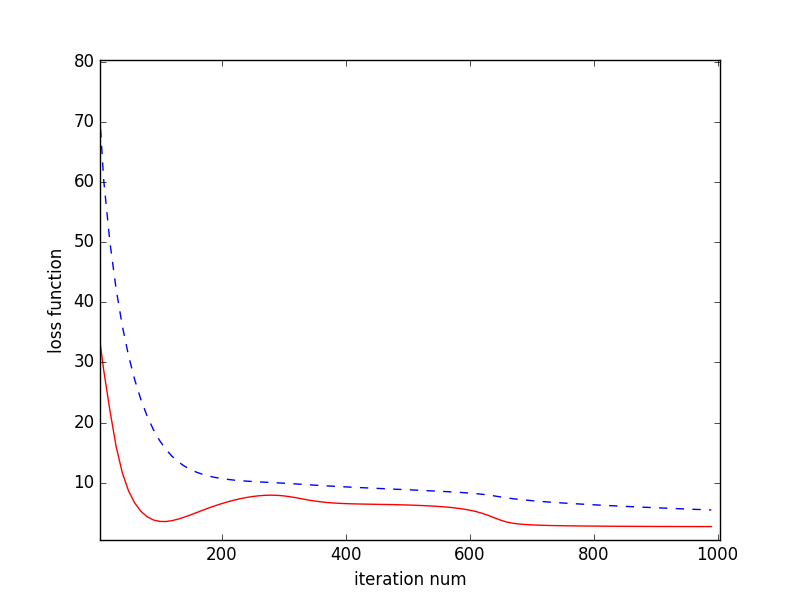
The current result we get from different band pass filters is showed on the left side. The x axis represents the iteration number and the y axis represents the error rate of the trained model. On the left side is what we got from our program. On the other side, is the cnn\_mlp result. The red line represent the result of training dataset , and the blue line is based on test dataset .

Currently , the minimum error rate in FFT in 1000 iteration is 24.421% and CNN is 3.151%.

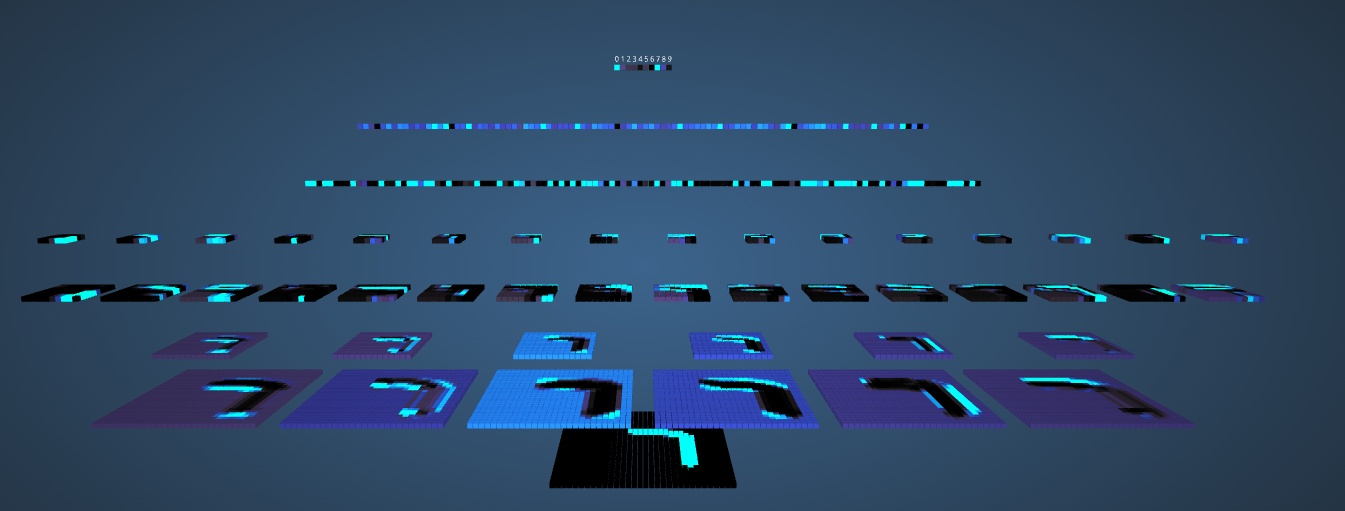


The following result is based on full fft filter, In the first layer, the image size is 28\*28, so we select 27 filters, and for the second layer we select 13 filters.

The red line represent the result of training dataset , and the blue line is based on test dataset . The minimum error rate in FFT in 1000 iteration is 4.231%



Following with the visualization of the CNN network :



1. The bottom pic is the input
2. Above the input is the first CNN layer
3. Above the CNN layer is the maxpooling outcome of first CNN layer.
4. Above the maxpooling is the second CNN layer and second maxpooling layer
5. The outcome of the maxpooling layer would be flattened into 1 dimension as showed in the pic .
6. The flattened 1d array is the input of mlp , which is fully connected.
7. The last layer is the softmax layer.



For the function of hidden layer is :

F is the activate function. It could be sigmoid or tanh .



In our case , There are 2 CNN layer and the output would be :

[YU’s comments 21 December 2016]

In terms of your current work, you may try “sparse localized deformation components”.

Download two sequences of motion clips, and .

For any set of , take decomposition as X=WC, . The component number should be the same, k.

Then, take SVD on C, , and update

After that, transfer motion style from one set to another one, i.e.

Where, I denotes an identity matrix.

Or from the 2nd set to the 1st one, vice versa.

The key points should be the relationship between local support regions and row/column vectors of U, which corresponds to styles.

[Yu’s comments 1st January 2017]

FFT based CNN,



Original CNN at <https://github.com/rasmusbergpalm/DeepLearnToolbox>



FFT based CNN seems to outperform the original CNN.

[Yu’s comments 6th January 2017]

In terms of “sparse localized deformation components”,

Download two sequences of motion clips, and .

For any set of , n is the number of frames and m is the number of joints, let , is the 1st frame.

Take decomposition as X=WC, . The component number should be the same, i.e. k. For motion data, it is unnecessary to define the local support region for every joint.

In terms of X, we may identify which joint has the biggest motion over the clip. The selected joint corresponds to the first component, . You may verify it as follows.

Let , then and . Check if removes the motion of the selected joint.

Then, take SVD on C, , and update

After that, transfer motion style from one set to another one, i.e.

Where, I denotes an identity matrix.

Or from the 2nd set to the 1st one, vice versa.

The raising issue is that the first component may correspond to motion of multiple joints. What do the other components correspond to?

Result:

C contains all components here. Take SVD on C, then blend two motions (put all components in computation), the resulting motion is normal based on Dynamic Time Warping

[Yu’s comments 17th Feb. 2017]

In terms of the positional motion data, you have shown the results of linear combination,

And SVD based combination,

Their performance looks same.

The distinct weakness is foot-skating. Your suggestion is to specify the foot path in advance. It needs to take into account how to make the body data compatible with the constraint of the given foot path.

Regarding the understanding of components, one is that one component (one row vector) corresponds to one segment of skeleton. Another one is that one component corresponds to a simple motion.

You may try to reconstruct motion by

To compare the resulting motion with the original one. Note that U is square with component number as its size and it involves the singular values here.

Furthermore, this is an approach to decompose motion into a set of simple motions. For non-linear combination, e.g. two motion data,

It may rearrange U by selecting row vectors from respectively, so as to synthesize motion, i.e. .

Result:

One component corresponds to a simple motion!

C contains all the components, Take SVD on C, and then determine components by U

After that select one component separately from two motions, blend motion?

[Yu’s comments on 3 March 2017]

Your implementation should be similar to Khiem’s work except working on different dataset. I draft the basic ideas here for your implementation,

Perform motion matrix decomposition on all the motion matrices and generate their individual components (ask Khiem for this code please).

Then, finish the following experiments,

(1) motion synthesis, given two motions, ,

For the two specified kth and sth components separately from , synthesize motion as,

(2) motion blending, for the specified kth components separately from , blend them in as,

(3) Motion Retargeting. For two specified kth and sth components separately from , retarget them into the third motion as,

(4) Test distortion. Test,

And

Check if there is distinct distortion to appear in …

Results

Since the first component occupies most motion, so all the experiments work on the first components

(1)blend motion

(2)retargeting motion

(Please add your results/rising problems/comments in this doc.)

[Thanh 09 Mar 2017]

# Support region for skeleton motion

Sparse localized deformation was designed for mesh motion, where a support region centering a max variation motion vertex. We can not define such regions in a skeleton motion. As a consequence of no support region, no L1/L2 penalty will be added, then decomposing a skeleton motion will have global effect. However, we need the localization of component to capture component as a body part motion, such as hand motion or leg motion in one component. In order to archive localize component, I pre-define a matrix of skeleton support region as bellow:

|  |
| --- |
| #Define a matrix of my support map for skeleton(22 joints)  M = np.ones(shape=(22, 22))  M[4, [3,4,5]] = 0  M[8, [7, 8, 9]] = 0  M[20, [19, 20,21]] = 0  M[16, [15, 16, 17]] = 0  M[1, [1, 2, 6]] = 0  M[9, [9,8]] = 0  M[10, [10,11,1]] = 0  M[14, [13,14,15]] = 0  M[18, [12,18,19]] = 0  M[5, [5,4]] = 0 |

This matrix represents the relationship between joints, let call this value is *R*. In case of mesh, this relationship is Euclidean distance.

For instance, M[4, [3,4,5]] = 0. This shows that relationship R from joint 4 to joint 3 4 5 is ZERO, while R from joint 4 to others is ONE. By this way, we can avoid penalty on joint 3,4,5 and apply penalty to other joints.

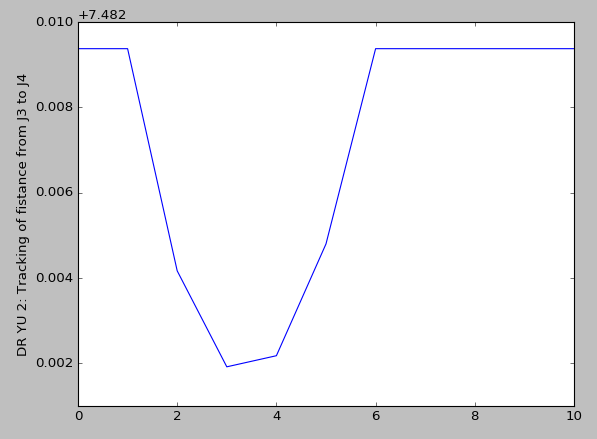
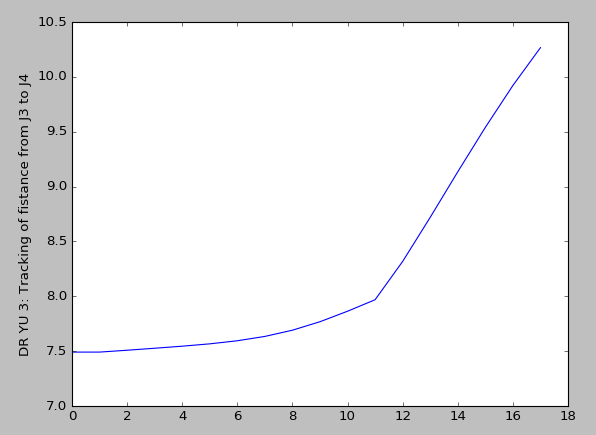
The comparison of motion decomposition with and without Skeleton Support Map shown in this video. (<https://youtu.be/JEePbEdRn3U>)

# Implementation of Motion blending and retargeting

The results shown in this video. (<https://youtu.be/jgl8JEmA-WU>)

# Distortion

Blended and retargeted skeleton motion have distortion. I attempted to observe the limb length of between some specific joints, the length varies quite a lot. Please have a look at figures below.



[Yu’s comments on 13 March 2017]

From your implementation of motion synthesis with each component corresponding to multiple specified joints and single joint (<https://youtu.be/qlHHvAUNWtU>), I guess that you only apply L1-PCA to motion matrix decomposition, i.e.

And then simply select some components for motion synthesis.

However, when specifying every component to correspond to one joint, you have to add a constraint like a mask matrix. Obviously, this is a constrained optimization problem compared to the above decomposition. The resulting components should be different from the above decomposition as well.

I hope to know if the original motion may be recovered by combining such components together, and how about the synthesized motion by such one component.

Explain what Sparse Coding is:

Sparse coding implementation

The usual iterative scheme,

Herein, rewrite it as,

i.e. to simply computation, select the smallest step-length.

Then, orthogonally projecting the resulting *x* onto sparse solution space yields,

Where , and usually depends on applications. This concludes the sparse code *x*.

The choice of the threshold determines how many joints are employed in the current recovered motion.

In motion synthesis application, y denotes the original motion matrix, D denotes the weight matrix W, x denotes the component matrix C here.

In SPLOC algorithm (of paper-Sparse Localized Deformation Components), the constraint is replaced by . The main modification is to implement this new constraint in SPLOC code.

Once threshold is fixed, the resulting components should contain lots of zeros. The reconstructed motion corresponds to a basic motion, which should be normal.

Then update the motion matrix, , repeatedly yield the components corresponding the other basic motions.

To verify the correctness, sum all reconstructed basic motion matrix and compare it with the original y.

[Yu’s comments on 22 March 2017]

First, can you describe your implementation of SPLOC codes with the change of computing component matrix, and put your experiment results here please. Is there any distortion or abnormal motion to appear in your experiments?

Second, we discussed how to define the basic motion in chat.

Suppose that the motion data *X* may be decomposed by,

Apply sparse coding to the component dimension k with L-2 norm along the 3D coordinate dimension n (i.e. L-2-0 norm),

The basic motion may be extracted from *X* by,

Set the sparsity factor *f*, and update *C*,

Which may be carried out iteratively as below,

1st basic motion,

Update *X*, i.e. residual motion,

2nd basic motion,

Update *X*,

And so on……

The sparsity factors may be either same or different depending on applications.

To verify the correctness of decomposition,

Again, return motion synthesis topic.

1. motion synthesis, given two motions, ,

For one basic motion,

Replace one basic motion with that of

1. motion retargeting,

I wonder if such retargeting really results in any distortion. ????

Result:

For sparse coding, we concern how many non-zero items there are in C. We set the same sparsity f, and determine the basic motions as,

1st basic motion,

Update *X*, i.e. residual motion,

2nd basic motion,

Update *X*,

And so on……

Reconstruction,

For the k-th basic motion,

Motion synthesis,

which needs to be pre-processed by DTW

Motion retargeting,

Results?

[Yu’s comments on 7 April 7, 2017]

You are familiar with SPLOC codes. So you are aware of how to estimate W and C matrices, and where updating C is replaced by my MatLab code.

Recover the original updating C codes without support regions.

When you generate the 1st pair of initial vectors, and , for the motion data X, apply ADMM to optimize the pair of vectors.

The resulting vector pair, W(1) and C(1), we construct the 1st basic motion as

Then, update and compute the 2nd basic motion, and so on.

It is unaware for me what is the relation of such basic motion and distinctiveness and attractiveness of emotion. You need to follow the reference papers below and figure out your findings.

FrankenFolk: distinctiveness and attractiveness of voice and motion

Jan Ondrej, Cathy Ennis, Niamh Merriman and Carol O'Sullivan.

ACM Transactions on Applied Perception, 13(4), 20, (2016).

Evaluating the Distinctiveness and Attractiveness of Human Motions on Realistic Virtual Bodies

Ludovic Hoyet, Kenneth Ryall, Katja Zibrek, Hwangpil Park, Jehee Lee, Jessica Hodgins, and Carol O'Sullivan.

ACM Transactions on Graphics (SIGGRAPH Asia 2013), 32(6)

[Yu’s comments on April 10, 2017]

Decompose basic motion as follows,

When you generate the 1st pair of initial vectors, and , for the motion data X, apply ADMM to optimize the pair of vectors.

The resulting vector pair, W(1) and C(1), we construct the 1st basic motion as

Then, update and compute the 2nd basic motion, and so on.

When synthesizing motion, we may select the first k basic motion as the principal motion of some motion clip X, that is,

Then, refer to my previous comments,

Where K is regarded as the component.

For different objects’ motions, we may exchange their individual K accordingly. The following tests are only some initial trials which need to be amended in your experiments.

1. for two objects' motion sequences with the same mood, analyse the synthesised motions through exchanging their individual components, e.g. similarity compared to the original motions, comparing their individual , comparing the before and after synthesized motion data;
2. for two objects' motion sequences with the different moods, analyse the synthesized motions through exchanging their individual , e.g. similarity compared to the original motions, comparing the synthesis motion with the original motion ;
3. for multiple objects' motion sequences with the same mood, decompose these motion data into the basic motion, that is,

And compute the component K accordingly. The resulting K may be regarded as the common component of this kind of emotion.

If substituting the component of some motion with the common K, does it work well?

For each , there are the individual . The mean may be the alternative component. Is there any difference between these two common components?

In psychology/cognition research, there is the term of “averaging face”. You may check the related papers by google, and design a test to compute “average of ONE emotion” based on the above computation. The rising question is why it may be regarded as the “average”.

Results:

Most results look fine! Using DTW does not lead distinct change!

**[Thanh 25 April 2017]**

**Implementation result of your suggestion above (date April 10, 2017)**

Both videos show the experiment of exchanging basic motion of *sad walking* and *normal walking*.

Video 1 current implementation: https://youtu.be/1YvqWH5dk2Y

According to our last discussion, the latest method (video 1) use ADMM to optimize first pair of initial vector W\_1, C\_1, then define basic motion X\_1 = W\_1\*C\*1.

Video 2 Sparse Coding: https://youtu.be/39e8t8HO7Y8

Use sparse coding to decompose motion into sequence of basic motions corresponding to sparsity (% of non zero).

From my point of view, synthesized motion of latest update looks abnormal, Sparse coding is the best so far

**[Thanh 2 May 2017]**

Synchronize input motions using (DTW) Dynamic Time Warping. Input motions may walk fast or slow, I used DTW to synchronize two motions based on their foot. Figure bellow shows the output motion Before and After DTW.

|  |
| --- |
| ../Desktop/Screen%20Shot%202017-05-01%20at%205.44.23%20PM.png |

I applied DTW to all input motion before decomposition and synthesizing process, the output motion looks a bit more natural.

**Difficulty**: How to show the relationship between component and emotion? We state that we can synthesize full body emotion between motions, but no evidence.