# Motion Texture

Harriet Pashley
Advisor: Yanxi Liu
Ph.D. Student: James Hays

## 1. Introduction

Motion capture data is often used in movies and video games because it is able to realistically depict human motion. One of its greatest limitations is the inflexibility for reuse in different projects. To get the exact shot that you would like requires an individual motion capture session. This is not only costly but time consuming as well. We would like to be able to easily take existing data and synthesize more of it or be able to modify it slightly to suit our needs. Motion capture databases such as Carnegie Mellon's [http://graphics.cs.cmu.edu] divide clips into different actions. It is difficult to piece these clips together in a reasonable way to create a desired longer motion.

There have been many successful results in texture synthesis for a variety of textures using different approaches. Some synthesize texture pixel by pixel based on the pixel's neighborhood [Efros and Leung, 1999], while others use patches which are overlapped and then sewn together along a certain optimal seam [Efros and Freeman, 2001]. Recently there have been many advances made in the texture synthesis realm which can be applied to the growth of motion capture data, namely the specific treatment of near-regular textures [Liu, Lin, Hays, 2004]. This specific classification of textures has regularities and symmetries with the addition of a limited amount of random noise and imperfections. Near-regular textures are seen all over in nature, including cloth patterns, brick walls, and even gait patterns [Liu, Collins, Tsin, 2002]. Algorithms have been tailored to near-regular textures to exploit their regularity while still preserving their subtle differences [Liu, Lin, Hays, 2004]. These synthesis methods use PCA to find the average tile and then deviate from it to form new tiles. These tiles are then pieced together to form a larger texture. We extend this concept to stride cycles in motion capture data for motion capture data. In Fig. 1 the repetitive nature of the foot and arm joints are apparent over the course of three different cycles. Texture synthesis algorithms are able to faithfully reproduce arbitrary amounts of a given input texture.

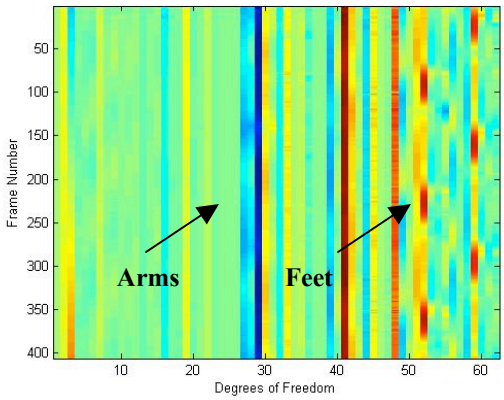


Figure 1: The frame of the motion is viewed as a row of the graph. Time is increasing from top to bottom. The periodicity can be observed in the feet and the arm joints across three different cycles.

## 2. Related Work

There have been many papers published on the subject of motion synthesis. Researchers have been able to piece together segments of motion capture data by specifying descriptions of motion in a specific order, such as "walk, run, jump" [Arikan, Forsyth, and O'Brien, 2003]. To reduce an animator's workload, research has been done to key-frame a small number of joints and use motion capture data to fill in the remaining joints based on the correlation between the joints [Pullen and Bregler, 2002]. Our work deviates from previous work which synthesizes human motion by segmenting data together which is statistically similar. Motion is viewed as a set of linear dynamic systems and their accompanying matrix of transitions which contains how likely one system will transition to another [Li, Wang and Shum, 2002]. The most widely used method to transition between two

motion clips is linear interpolation. The physical correctness of interpolated motions has been justified by evaluating it in terms of standard physical properties such as linear and angular momentum, the contact between feet and the floor and the continuity of velocity. However, it is still not guaranteed to look natural. [Safanova et al., 2005].

In this paper we will examine each category of texture synthesis algorithms as mentioned above as well as explore the importance of correlation between joints.

## 3. Texture Synthesis Algorithms

### 3.1 Pixel-Based Methods

In Efros and Leung's 1999 paper they were able to create arbitrary amounts of texture by creating a data structure of textons (see Fig. 2). Textons define the basic unit of a texture, which contain the color information of the center pixel and of those pixels in its immediate vicinity which are called neighbor pixels. Textons can be created for all pixels of the input texture. When synthesizing, the texton *t* composed of a neighborhood of size *n* around the current pixel *p* is compared with each texton of the input texture *t'* by minimizing the distance between them using distance metric *d*. The center pixel of texton *t'* with the smallest distance from the center of *t* is the value of *p*. This process is repeated for each pixel to be synthesized. Applying this to motion capture data, we equated each "pixel" to each angle value *a* at frame *f* and degree of freedom of joint *x*. Our "texton", which we will refer to as a *motion texton* is the neighboring angles around the current angle. This is not the same definition used by Li, Wang, and Shum (2002). We deviate from Efros and Leung's method only in the search space for each angle *a*. If we searched over the entire original motion we could synthesize a head joint angle with a foot joint angle. To avoid this, we minimize the distance between *a* and *a'*, where $a' \in a_1 \ldots a_z$ and *z* is the length of the input motion sequence. This produced a visually pleasing result when given a sufficiently large neighborhood size. The neighborhood size was dependent on the gait cycle, about 20 frames on average.

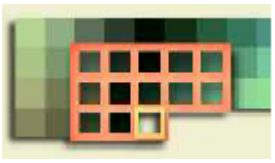

Figure 2: A visualization of a texton.

### 3.2 Patch-Based Methods

Another approach for texture synthesis is the image quilting method of Efros and Freeman. They create a set of blocks from the input image which are "all such overlapping blocks in the input texture image" of a size specified by the user. The initial block is selected and placed in the larger image. The patch set is searched for the block which has the least variation in the overlapped region between the two neighboring blocks (see [Efros and Freeman, 2001). This process is repeated for each patch as they are placed in the larger image. An optimal path is computed in the overlapped region to seamlessly join the patches. Relating this to motion capture data, we take two motions *m1* and *m2* (which may be the same motion) and search for the best overlap between them over *n* frames. This occurs at frame *t1* of *m1* and *t2* of *m2*. We compute the optimal seam between m1 and *m2* joint by joint, where *x* is each joint as seen in this pseudocode:

```
for i = t1 to t1 + n
  for j = t2 to t2 + n
    calculate distance between m1x[i] and m2x[j]

return i and j of minimum distance computed
```

This means that over the *n* transitional frames each joint *x* could transition between *m1* and *m2* at different frames for a smoother transition. When *m1* and *m2* are the same motion, a threshold needs to be set for how quickly the motion can transition (the minimum value of *t1*). Without this threshold the best overlap will be computed at frame 1.

### 3.3 Tile-Based Methods

When using a pixel based texture synthesis algorithm the size of the neighborhood has a large impact on the quality of the results. A small neighborhood will not incorporate the structure of the texture into the distance calculation. Even an arbitrarily large window does not capture the correct regularity. For all of the infinite number of regular textures there are only 17 different lattice/tile structures that define the symmetry of the texture. The lattice can be detected in a texture and the tiles used for synthesis. Tiles are placed and then blended together using dynamic programming. This preserves the intensity/color variations of the texture as well as the regularity of the texture [see Liu, Tsin and Lin, 2005].

While the structural regularity is not preserved when using a large neighborhood size, the correct choice of neighbors is an important part of patch-based texture synthesis methods.

## 4. Correlation Between the Degrees of Freedom of Joints

The format of motion capture data is that each joint is given a number and these numbered joints are laid out in sequential order in an array. Thus adjacent neighbors in the data structure may not be logical numbers. For example, in Fig. 3 the lower back joint is stored next to the right toe joint. When using Efros and Leung's method (1999) these kinds of neighborhoods may be meaningless. To make the neighborhood meaningful we look at the correlation score between joints. We quantify this correlation using Pearson's correlation coefficient, ρ. This coefficient is a value between -1 and 1 which measures the linear relationship between two sets of numbers *s1* and *s2*. 1 is a perfect correlation between *s1* and *s2*, 0 is no relationship between the two sets, and -1 is an inverse relationship between the two.

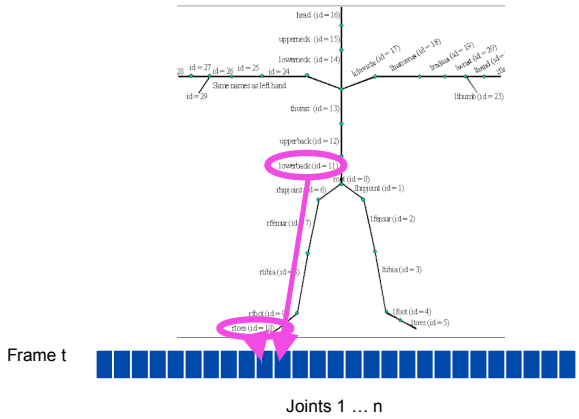


Figure 3: The physical neighbors of a joint are not logical neighbors of a joint. The lower back and right toe are not correlated joints, yet using Efros and Leung's method (1999) their values would affect one another when synthesizing a joint.

Applying this to our motion synthesis, we compute ρ for every pair of degrees of freedom in motion *m*. If $\sigma_{xy} = (\overline{xy}) - (\bar{x})(\bar{y})$, $\rho = \frac{\sigma_{xy}}{\sigma_x \cdot \sigma_y}$. We store these values in an array and use them to weight the value of the physical neighbors of the degrees of freedom of joint *x* during synthesis. The same algorithm is used as described in the previous section with the exception of the distance metric *d*. For each degree of freedom of joint *x*,

$$d = \sum_{i=1}^{n} abs(neighbor\_weights \cdot frame_i^{\,T}),$$

where *n* is the size of the neighborhood. The absolute value is taken because we want to weigh positively and negatively correlated joints equally as they provide the same amount of information about the motion.

## 5. PCA Tiling of Motion

A near-regular texture [Liu, Lin and Hays, 2004] can be decomposed into two parts, a regular texture and its deformation field. When synthesizing a brick wall the authors find the average brick which they call the mean tile. They then use PCA to construct the bases of the tile and create a linear combination of the mean tile and its bases. We look at motion capture data in this light we manually segment a walk into strides. We define a stride to start at the first frame at which the right food touches the

ground and the end of the stride to be the last frame before the right foot touches the ground at the end of the walking cycle. These strides are then averaged together to form the mean stride. The bases are constructed and new motion can be created by forming a linear combination of the mean stride and its bases. The strides created by this method can be pieced together using any of the three methods described in Section 3.

One of the difficulties of this approach is to create motions without foot sliding. To compute the mean stride, cycles must be interpolated to be the length of the longest stride. This causes the shorter cycles to be stretched, thus causing feet to slide. We are currently working on ways to mitigate this problem as explained in our future work. Also, it would be instructive to find a means of computing a threshold for eigenvector coefficients to produce realistic motions.

## 6. Future Work

We are looking to expand our work firstly by acquiring better data for PCA. By using many cycles of a motion a more meaningful average stride can be computed. We are going to also examine our method with data where the feet are fairly regular (similar stride lengths and times) and the hands are uncorrelated. We would like to expand the autonomy of our method, by being able to automatically detect the beginning and end of each cycle in a motion. This will greatly reduce the pre-processing steps of our current method for synthesis using PCA.

Currently, our method needs a post-processing step to clean up foot sliding. We are hoping to use correlation to automatically detect sliding feet. When feet begin to slide we hypothesize that the correlation between the velocity of the feet and velocity of the root will change. The velocity of the root can be calculated by finding the difference between the different positions of the root over the duration of the motion. Finding the velocity of the feet in world coordinates would allow for a comparison between the velocities of the two joints.

We are also aware that it may be better to synthesize each joint of the motion instead of synthesizing the degrees of freedom of the joint separately. This is not straightforward as different joints have different numbers of degrees of freedom. It involves turning each joint from a Euler angle representation into a quaternion representation. This affects how the correlation between joints is computed as the correlation of two vectors is yet to be defined precisely.

Our computational treatment of motions as near-regular textures may provide valuable insights into the improvement of motion transplants [Ikemoto and Forsyth, 2004]. Using graph cuts [Kwatra, Schödl, et al, 2002] we will be able to stitch together subsets of different motions across time, hopefully improving existing results.

## REFERENCES


Arikan, O., Forsyth, D.A., and O'Brien, J.F. 2003. Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):402:408.

Efros, A.A. and Freeman, W.T. 2001. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, pp. 35-42.

Efros, A.A. and Leung, T.K. 1999. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision.*

Ikemoto, L. and Forsyth, D.A. 2004. Enriching a motion collection by transplanting limbs. *ACM Transactions of Graphics,* pp. 99-108.

Kwatra, V., Schödl, A, et al. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. *ACM Transactions of Graphics,* 22(3):277:286.

Li, Y., Wang, T., Shum, H.Y. 2002. Motion Texture: a two-level statistical model for character motion synthesis. *ACM Transactions of Graphics,* pp. 465-472.

Liu, Y., Lin, W.C., and Hays, J. 2004. Near-regular texture analysis and manipulation. *ACM Transactions of Graphics*, 23(3):368-376.

Liu, Y., Collins,R.T. and Tsin, Y. 2002 ``Gait Sequence Analysis using Frieze Patterns", *European Conference on Computer Vision*. Copenhagen, Denmark. May 28-31, 2002.

Liu, Y., Tsin, Y., Lin, W. The Promise and Perils of Near-regular Texture. *International Journal of Computer Vision*, Vol. 62, No. 1-2, April, 2005, pp. 145 - 159.

Pullen K., Bregler C. 2002. Motion capture assisted animation. *SIGGRAPH* 02.

Safanova, A., Hodgins, J.K. 2005. Analyzing the physical correctness of interpolated human motion. *ACM Transactions on Graphics*, pp. 171 – 180.