Flying Sparrows Capture Poses:

Using PSO to Map Appropriate Nomenclature to Motion Capture Markers

-- Final project for EECE665 Complex System & Network, Winter 2010

Chunsheng Fang[1], Yunzheng He[2], and Michael Tolston[3]

Dept of Computer Science[1], Dept of Aerospace Science[2], Dept of Psychology[3]

University of Cincinnati, Ohio, 45221

1. Introduction

Motion capture devices are used in a variety of fields, including graphics design, movies, and the sciences. It is typically the case that during the initialization of the capturing sequence all markers must be hand labeled in order for the rendering software to successfully map a skeleton to the position of the relevant markers. For example, the industry standard file formats of .c3d and .bvh require the markers to be labeled in order for viewers to be able to render the animations.[1,2] (see appendix 1 regarding how we imported our data into Motion Builder for our class presentation). As another example, Vicon, the passive motion capture system used by the large motion capture lab at the Carnegie Mellon University, also requires manual labeling [3].

Even if rendering software is not implemented, it is still necessary to be able to meaningfully index the markers for subsequent evaluation. This is especially true in passive marker systems, as individual markers are not uniquely identified until this process is done manually. This is less true in active marker system because each marker is automatically identified with an integer. However, it would be beneficial to have a system automatically index the markers with a meaningful nomenclature, as this would allow a more intuitive interpretation of the data without memorizing or referencing a list of numbers, and would also facilitate importing the data into rendering software.

For this project , we approached the problem of marker identification by utilizing a particle swarm optimization (PSO) algorithm to map a simple humanoid skeletal model to the data generated from un-tagged positions of motion sensors attached to a person. The dataset we used in testing our algorithm covers a wide breadth of full body dynamical motor action, including walking, turning in a circle, and jumping. In this paper, we also propose a novel and straight forward distance measure to serve as the objective function to cut down the searching space from 6 parameters to 3 parameters, utilizing the heuristic of Center of Mass.

2. Related Work

Research in this area has been done using a PSO algorithm to estimate human posture from camera images for use in the video conferencing [4, 5]. PSO has also been used to map geometric shapes to images captured from a camera network in order to allow communication among a network of smart cameras to estimate gesturing [6].

We could find no work which uses PSO for human posture estimation from data collected using a motion capture system. This is likely due to the fact that it is a seemingly trivial application, considering that by simply applying names to the markers one can use any number of software libraries to generate a human skeleton and map it to the position of the markers. However, as we have already mentioned, our application could streamline and automate the process of applying the nomenclature. Additionally, and likely more important, is the accurate estimation of missing data, which could be directly derived from a modification of our algorithm.

3. PSO

Particle swarm optimization (PSO) is a search algorithm based on the complex system theory, in which a group of particles simultaneously explore a solution space to find an optimal output. The solution is based on an evaluation function, which is unique to each problem. The particles actually represent values of the various dimensions in the solution space to a given problem. Each particle has a position, a velocity, and knowledge of its own best position as well as that of its neighbor's best position. These three components can be seen in the following equation for PSO:

$$v_{jk}(t) = \omega v_{jk}(t-1) + C r_c(t)\ [p_{jk} - x_{jk}(t)] + S r_s(t)\ \ [p_k - x_{jk}(t)]_{jk}(t)]$$

Note that the order of the components in the equation is velocity, own best and global best. During every iteration each particle compares its current values with its own best value and the global best value. They then update their trajectory according to the values of the parameters chosen by the programmer. It is these parameter values which weight the contribution of each factor, and thus must be carefully chosen and tested for successful implementation of PSO.

Table 1: Parameters of PSO Algorithm

Name	Meaning
ω	Inertia parameter
C	Cognitive Component: Own Best
S	Social Parameter: Global Best

4 Method

There are several methods in previous works [4,5,6] in which solve the problem of pose estimation using PSO. We used these algorithms as a basis for our own simplified version, tailored to our specific motion capture device and the project requirements.

4.1 Problem Formulation

The essential problem we approached is how to map the point clouds of the data to a known model point cloud. If we can find the optimal parameters of transforming the data to the model, with a minimum objective function, then we can transfer the model sensor ID to the data

point clouds. The result is that the data sensors are mapped with the correct sensor ID. Note that, here, model means the previous frame of data, and data means the current frame. To accomplish this, we used the PSO Toolbox for Matlab [7]. We will now identify four main points of our problem.

First, our motion capture device has a high-frequency sampling rate of 60Hz, which is sufficient even for unusual human body movement. In our experiment, the maximum rotation angle between two contiguous frames is less than 2 degrees. Therefore, we assume that between any given two contiguous frames, they can be considered as rigid body, since they don't have significant deformation.

Second, we can reduce the common formulation of 6 DOF (Degree of Freedom) rigid body affine transformations to 3 DOF, which leaves only the rotation parameters to be optimized. To do this, we utilize the heuristic that the translation vector between 2 rigid bodies is actually pointing from the COM (Center of Mass) of the first rigid body to the second one. Therefore, we can normalize each rigid body by their own COM. This leaves only the 3 rotation parameters (rotation along xy, xz, yz axes respectively) to be determined. Note that in this step, we cut down the searching space from 6D to 3D, which is a significant improvement. Rotations were defined using the following matrix:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Third, we note that the distance measure is critical to designing the objective function for PSO algorithm. We propose two different distance measures, **all pair-wise distance measure** and **greedy pair-wise distance measure**. A comparison of these 2 difference distance measures is shown in Section 4.1.

Fourth, in practice, we have to face the unfortunate problem of missing data in the motion capture device in some frames, due to all-angle occlusions. Occlusion results in all-zero 3D position data for the missing points. For simplicity, we replace the missing data with the previous non-zero 3D position.

The data set for our PSO are time series from several motion capture sessions. We used the NDI Optotrak Certus, which utilizes active markers, and there were 14 markers in the full body setup (see table 1 in the appendix). The process of estimating body pose is formulated as finding the pose which maximizes the overlap between the silhouettes of the model and the data.

4.1 The Model

The model for our experiment, which represents the reference to which the PSO optimizes, is a defined set of coordinates corresponding to a known position of markers, which themselves correspond to a standard pose of a human subject. Fourteen markers were

individually placed at the major joints. After initialization to the known model, the model used iteration (i-1) as the referent for the optimization algorithm.

4.2 Distance measure and objective function for PSO

We propose two distance measures as the objective function in PSO algorithm, **all pair-wise** and **greedy pair-wise**.

All pair-wise distance (D1)

Steps:

1. Compute pair wise distance between all points in model and data sets;
2. Sum up all these distance in step 2, to get an "All pair-wise distance measure".

Greedy pair-wise distance measure (D2)

Steps:

1. Compute pair wise distance between all points in model and data sets;
2. Pick the minimum distance between the i^{th} point in data and all points in the model; This gives the perfect match for i^{th} point in data;
3. Sum up all these distance in step 2, to get a "Greedy pair-wise distance measure".

Note that this distance sometimes is ambiguous because one model point may be matched to multiple data points. However, this ambiguous matching can be alleviated by high sampling rate in human motion capture. Our experiment shows that in 60Hz sampling rate, the human body rotates less than 5 degrees, which is not likely to have ambiguous matching.

5 Experiments

We ran two experimental tests to determine the limitations of our algorithm. We used the most challenging motion sequence from the data series we collected, which is of a human model turning around in a circle. For the first test, we extracted two samples from the time series which were spaced at a sufficient distance apart to allow for a 30 degree rotational difference between the two points. We repeated the procedure for the second test, except this time we allowed for a rotation of 70 degrees. Our experiments show that PSO can find the global optimal parameters even in large angles up to 70 degrees difference between the model and the data.

5.1 Parameters

Table 2. Parameters modified in our PSO algorithms

Name	Value	Meaning
options.niter	50	The maximum number of iterations
options.npart	60	The number of particles in the swarm
options.vmax	3	The absolute speed limit on velocity

options.cbi	0.03	The initial value of the individual best acceleration factor
options.cgi	0.01	The initial value of the global best acceleration factor

Note that we set the "vmax" parameter to 3, which is relatively small. This is because we do not want the particles to be traveling so fast as to overshoot the optimal solution. Also, the initial acceleration factors are set to be small to guarantee a fine search in each neighboring parameter space for each particle in the swarm. The absolute speed limit on velocity was also set to a relatively low value of 3, in order to prevent possible overshooting of the optimal point in the solution space. The default value of this parameter in the PSO toolbox is set to infinity.

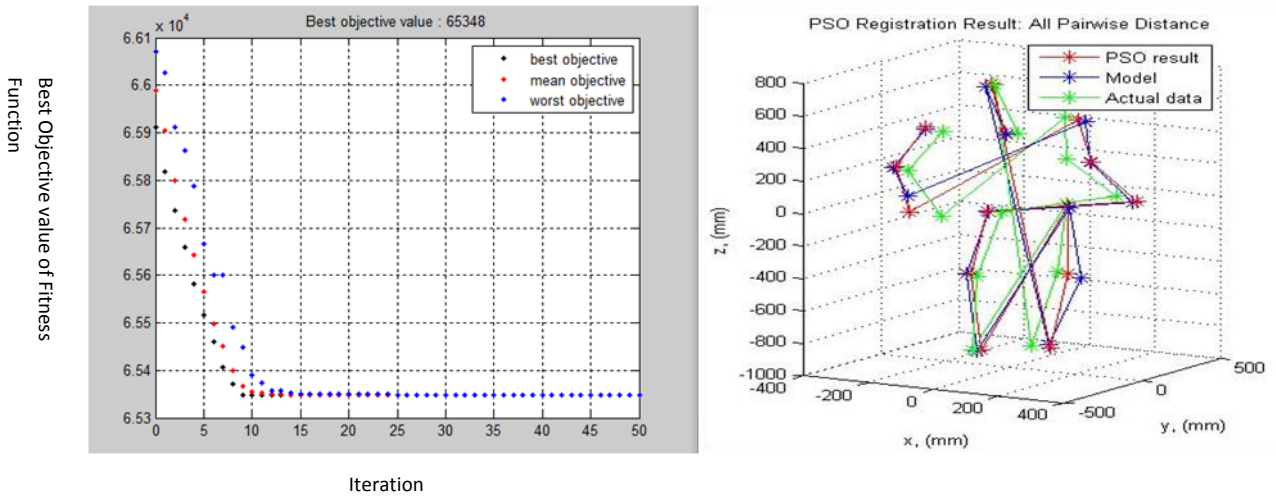


Figure 1: Experiment on 30 °rotation (Distance measure of D1 and D2) :

Note that the lines that connect the point clouds in the figures are just for better visualization

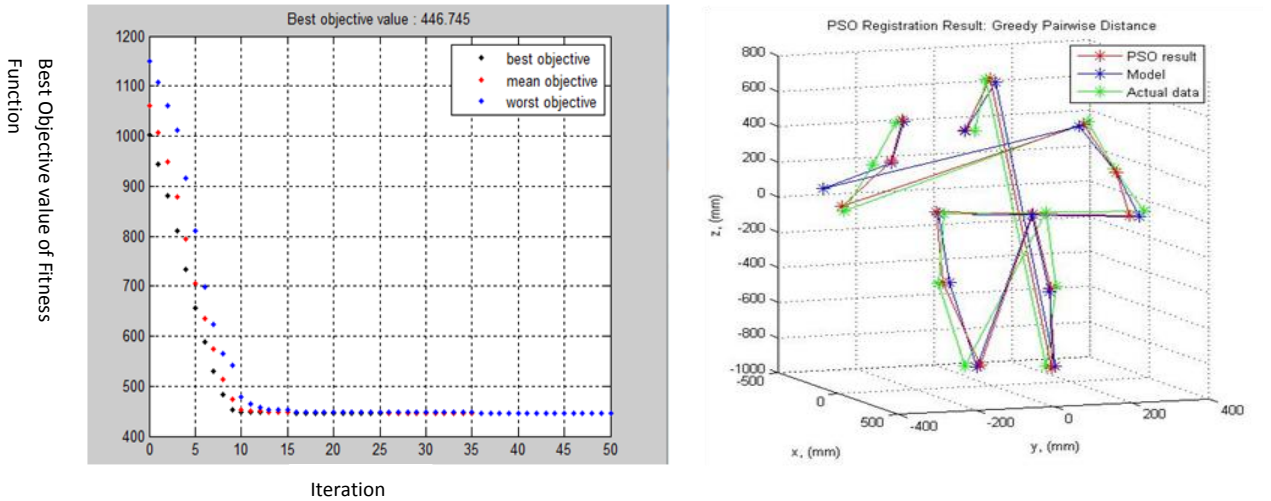


Figure 2: Experiment on 30 °rotation (Distance measure of D1 and D2: Greedy Pairwise Distance)

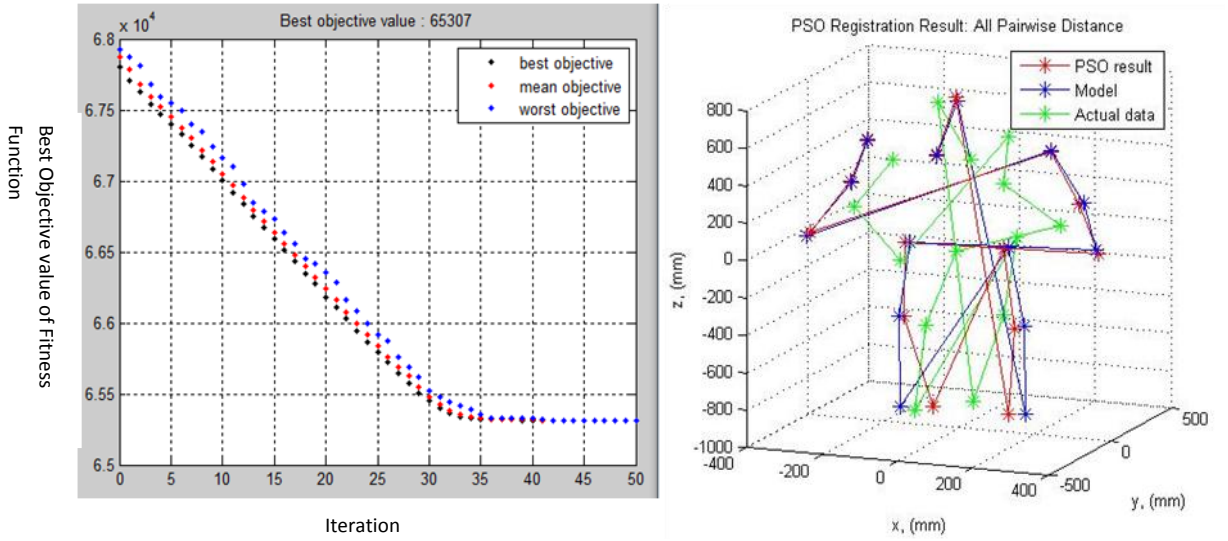


Figure 3: Experiment in 70 °rotation (Distance measure of D1 and D2: All Pairwise Distance)

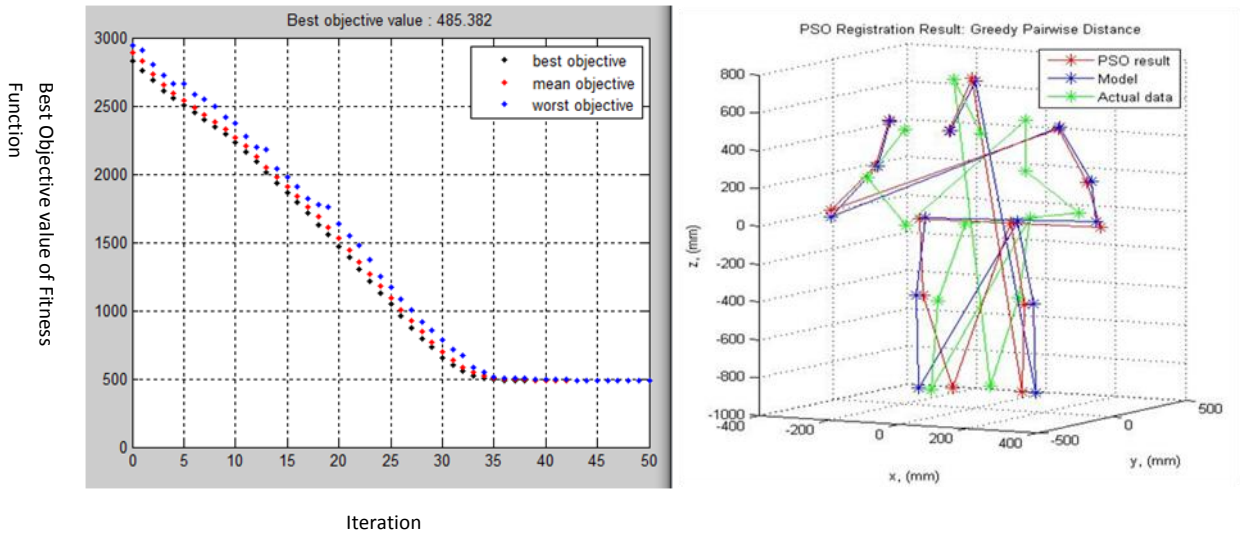


Figure 4: Experiment in 70 °rotation (Distance measure of D1 and D2: Greedy Pairwise Distance)

6. Discussion

In our worst case experiment, with a 70 degree rotation, the PSO algorithm converged onto a global minimum after approximately 35 iterations. In our more modest rotation of 30 degrees, the algorithm converged after only 15 iterations. This means that our algorithm could be used to initialize markers even when the subject's pose differs significantly from the standard pose of the evaluation model. Also, since both of these cases far exceed the typical 2 degree difference between contiguous frames, we believe it is well within the capability of this algorithm to keep pace with real time data. We also found that the value of the all pairwise distance measure is generally very large, and is not efficient compared to the greedy pairwise distance model.

While the online capability has yet to be tested, we have reason to believe that, with a tuning of the algorithm to abort iterations after global convergence, it will be feasible to run this program in real time, though the code will have to be converted to C++ to bypass the overhead incurred by an interpreted language. Doing this would allow the algorithm to be incorporated into tracking software, which in turn would allow a precise estimation of marker position in the likely situation of marker occlusion.

Contributions

Each member attended regular meetings to discuss the problem and brainstorm solutions to the algorithm. Victor Fang wrote the technical parts of the algorithm. Michael Tolston contributed to writing the report, and Yunzheng He worked on importing the model into Motion Builder.

References

[1] Biovision BVH [online]. Available: http://www.cs.wisc.edu/graphics/Courses/cs-838 1999/Jeff/BVH.html

[2] C3D User Guide [online]. Available: ftp://ftp.c3d.org/manuals/c3dformat_ug.pdf

[3] F. De la Torre, J Hodgins, J. Montano, S. Valcarcel and J. Macey, "Guide to Carnegie Mellon University Multimodal Activity Database" [online]. Available: http://www.cs.cmu.edu/~ftorre/cmu-mad.pdf

[4] C. Robertson and E. Trucco, "Human body posture via hierarchical evolutionary optimization," *British Machine Vision Conference 2006*, 2006, pp. 999–1008.

[5] S. Ivekovic and E. Trucco, "Human Body Pose Estimation with PSO," *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2006, pp. 1256-1263.

[6] C. Wu and H. Aghajan, "Model-based human posture estimation for gesture analysis in an opportunistic fusion smart camera network," *Proc. of the IEEE International Conf. on Advanced Video and Signal-based Surveillance (AVSS 2007), London, UK*, 2007.

[7] PSO Toolbox [online]. Available: http://psotoolbox.sourceforge.net/

Appendix:

Use of Motionbuilder(Autodesk) to achieve a motion capture visualization

Autodesk MotionBuilder real-time 3D character animation software is an ideal tool as an postprocessor for captured c3d data file as well as for real-time character simulations. **We adopted MotionBuilder because it is a good tool for "retargeting," where retargeting means adapting motion to a character often with a proportion that is different from the proportion of the capture subject.**

MotionBuilder has an intermediate "skeleton" called an Actor. It's a set of constraints that will provide the source motion for our character. First we have to match the marker points to corresponding areas of the actor. After the Actor's proportion has been matched to the proportion of the capture subject, a character has the proportion of the 3D character that we build. We make a link between an Actor and a Character using the following abbreviated procedure:

Step 1. Importing marker data

Step 2. Bringing in Actor

Step 3. Matching Actor to markers

Step 4. Creating a marker set

Step 5. Exporting the marker set

Step 6. Re-using the marker set

Step 7. Bringing in a Character

Step 8. Correlating the skeleton and Character

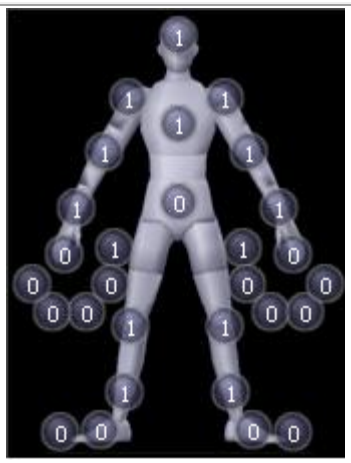
Step 9. Characterizing.

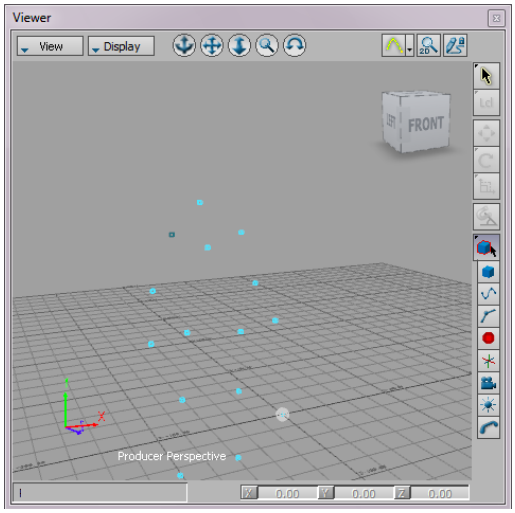
Step 10. Activating Actor and Character

Step 11. Editing motion

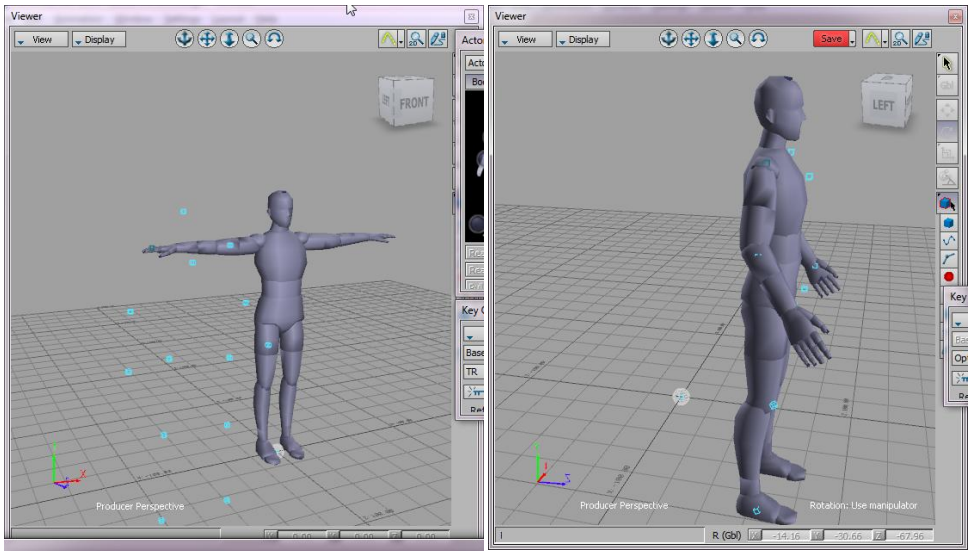
Step 12. Rendering video output

Table 1: Marker Mapping Index

	Marker set mapping	
Section	No.	Names
Head	1	CBHD
Shoulder	2	RSHO, LSHO
Elbow	2	RELB, LELB
Wrist	2	RWA,LWA
Upper torso	1	CLAD
Thight	2	RTHI,LTHI
Knee	2	RSHN,LSHN
Ankle	2	RANK, LANK
		



C3D Imported to Motionbuilder, first frame with a standing position



Before matching actor to markers **After matching actor to markers**