Identifying Motion Capture Tracking Markers with Self-Organizing Maps

Matthias Weber[*]
FGAN e.V., FKIE, Germany

Heni Ben Amor[†]
TU Bergakademie Freiberg, Institute for Informatics, Germany

Thomas Alexander[‡]
FGAN e.V., FKIE, Germany

Abstract

Motion Capture (MoCap) describes methods and technologies for the detection and measurement of human motion in all its intricacies. Most systems use markers to track points on a body. Especially with natural human motion data captured with passive systems (to not hinder the participant) deficiencies like low accuracy of tracked points or even occluded markers might occur. Additionally, such MoCap data is often unlabeled. In consequence, the system does not provide information about which body landmarks the points belong to. Self-organizing neural networks, especially self-organizing maps (SOMs), are capable of dealing with such problems. This work describes a method to model, initialize and train such SOMs to track and label potentially noisy motion capture data.

Index Terms: H.1.2 [Models and Principles]: User/Machine Systems—Human information processing; I.2.6 [Artificial Intelligence]: Learning—Connectionism and neural nets

1 Introduction

Motion Capture (MoCap) enables a user to capture human motion in all its intricacies. Usually, MoCap systems use markers attached to anatomical landmarks to track certain points on the body. Active markers, i.e. markers that emit or receive signals, are big or need cables and therefore hinder natural motion. As a consequence, passive markers are often used. They are little reflector spheres that do not disturb the participant. Unfortunately, MoCap data provided by such passive systems often has deficiencies. Occluded markers might occur leading to not seen points and therefore lost tracking in subsequent frames. Additionally, MoCap data is usually unlabeled, i.e. no information about the markers is given. Neural networks are able to deal with such problems as they try to generalize on the learned data. Those with self-organizing features, e.g. self-organizing maps (SOMs), can already represent a map of linked structures like a human skeleton. In this work, a method will be presented how to model, initialize and train such SOMs so that they adapt to certain poses extracted from potentially noisy motion capture data and identify tracked points.

2 Related Work

Research on Motion Capture is in particular focused on robust marker tracking and skeleton fitting. For the former problem, [1] proposed an extended Kalman filter for preprocessing in conjunction with a motion model based on exponential maps for later estimation of the human skeleton configuration from the tracked point cloud. For the problem of skeleton fitting, many approaches revolve around fitting a reference skeleton model into the data cloud by using least-squares methods [2]. In [3] global and local techniques for skeleton fitting are presented. Global techniques consider the whole skeleton at once, while local techniques perform adaptation on a limited number of bones.

[*]e-mail: m.weber@fgan.de
[†]e-mail: amor@informatik.tu-freiberg.de
[‡]e-mail: alexander@fgan.de

IEEE Virtual Reality 2008
8-12 March, Reno, Nevada, USA
978-1-4244-1971-5/08/$25.00 ©2008 IEEE

3 Algorithm Description

In order to identify tracked markers and to fit a corresponding skeletal model, the presented algorithm first employs a principle components analysis (PCA) on the captured point cloud. The PCA transforms the data by projecting it onto a set of orthogonal axes indicating the directions of greatest variance in the data. When applied to a captured standing posture with the arms reaching out, this leads to a properly aligned and flattened point cloud. It is oriented such that the arms reach out in the direction of the x axis. The feet and head are oriented along the y axis. After this, a user-supplied reference skeleton model is scaled to the extensions of the x and y axis. This leads to an accurately aligned model with respect to the projected point cloud in PCA space. The scaled skeleton model is then retransformed into the real-world 3-dimensional space and moved to the median point of the test person's point cloud. As a result, the model is now aligned to the test person's initial posture.

After this initial step, the SOM is adapted to each frame of the captured animation. For this, the skeleton structure is considered as a SOM with joints represented as neurons. To ensure constant segment lengths in this skeleton structure over time, first every joint's position is adapted so that the distance to its parent is kept nearly the same as the distance in the initial scaled skeleton model built via PCA. This way the skeleton structure stays the same, even though the SOM might try to converge several joints to one point of attraction.

After preserving the segment lengths, the position of every tracked marker is presented to the SOM in every training step, with $\vec{x}$ being the training vector. Distances to all prototype vectors $\vec{m}$ are then computed, using Euclidean distance measure: $\|\vec{x}-\vec{m}_b\| = \min_i\{\|\vec{x}-\vec{m}_i\|\}$. The neuron with its prototype closest to $\vec{x}$ is the winning neuron b.

The prototype vectors are updated according to the following update rule, where t is the current iteration, $\vec{m}$ again is the prototype vector and ε and σ are the learning rate and learning radius respectively. The e expression on the right side is a neighborhood kernel centered at the winning neuron vector $\vec{m}_b$.

$$\vec{m}_i(t+1) = \vec{m}_i(t) + \varepsilon\,(\vec{x}-\vec{m}_i(t))\,e^{\left(-\frac{\|\vec{m}_b(t)-\vec{m}_i(t)\|^2}{2\sigma^2}\right)} \tag{1}$$

This adaptation step moves the winner neuron towards the current training point with the learning parameters ε and σ. All other neurons are also adapted towards this point but with other, much lower learning parameters. Such lower learning parameters seem to achieve a good compromise between attraction to the current training point and not moving too far away from the point the neuron normally belongs to.

Above steps are computed several times to properly adapt to a posture. After adaptation has finished, the neurons share nearly the same positions as the corresponding markers. Therefore, the markers can be identified and labeled using a simple nearest neighbors computation. As soon as this step has finished, the process has been completed for the current data frame. The model is now adapted to the markers and the markers are labeled according to the neuron names.

4 Evaluation and Conclusion

For evaluating the algorithm we performed a set of experiments. A participant performed several motions: Moving to an initial pose, performing some arm movement and body movement, doing the initial pose on several different places with different orientations. Finally, movement while using a device was captured. This sums up to 8 datasets with 14736 data entries, overall. First evaluation revealed that the system captured a minimum of 4 and a maximum of 21 markers, in average 17.51 markers with a standard deviation of 2.02. This means that a good average number of markers, 17 to 18, was detected and that there were few outliers because of the low standard deviation. Nonetheless, there were some bigger outliers, with a minimum of only 4 markers.

A grid search was performed to investigate the effect of learning parameters on the neural network. The evaluated parameters were ε, σ and the parameter lpm which is used to inhibit the learning capabilities of all non-winner neurons. The grid search performs the learning algorithm for each set of parameters and records the performed error. For computing an error the following metric was used:

$$E = \sqrt{\frac{n}{max_n}^2 + \frac{1}{N}\sum_i^N \|\vec{x}_i - \vec{s}_i\|} \qquad (2)$$

with E being the error, $\vec{x}_i$ the neuron vector, $\vec{s}_i$ the corresponding sample tracking vector, n the number of iterations and max_n the maximum possible number of iterations. As can be seen, the distances between the neurons and the tracking markers compose one part of the error metric. The other part of the error is based on the number of iterations necessary to achieve the adaptation. This equation ensures that even if one of the error parts gets low but the other is still high (like low number of iterations but high distance error for the neural net), there is still a measurable error.

Figure 1 shows the results of the grid-search for an area that is most interesting in terms of low error. The lpm parameter describes how much to inhibit non-winner neurons from learning. It should be in a range from 0.2 to 1.0, from a nearly winner-takes-all network (WTA) to a normal network with all neurons learning the stimulus. For this scenario, an optimum of 0.4 was chosen, i.e. non-winner neurons do not adapt too much to markers they do not belong to. Therefore, this value promises a good adaptation on consecutive frames in an animation.

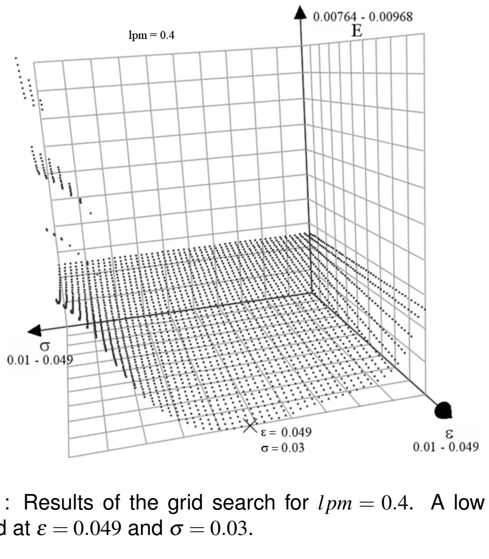

Figure 1: Results of the grid search for $lpm = 0.4$. A low error is achieved at $\varepsilon = 0.049$ and $\sigma = 0.03$.

The participant was also orientated in arbitrary directions to check the method's capabilities to cope with such situations. Especially the PCA algorithm, as it is used here to calculate position and orientation of the participant, was evaluated. In all cases the algorithm performed very well to find position and orientation. Even the SOM was trained correctly for every situation.

After that, a part of the data set, that contained only few marker occlusions (one or two markers occluded for only a few frames) was used to train and adapt the algorithm. This data was additionally degraded with one marker being occluded for a longer time. For this scenario, all markers were identified correctly. The addition for preserving the segment lengths proved very well in this situation. Figure 2 shows a sequence of images for the initialization and training steps and for some animation steps.

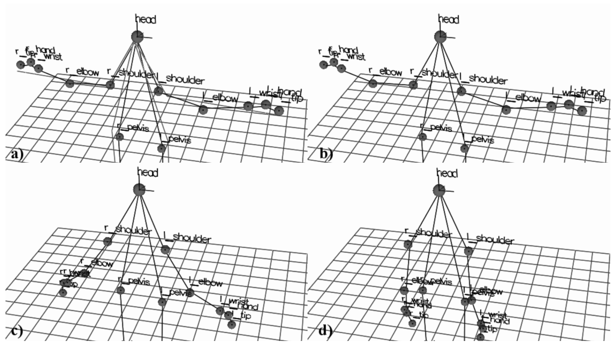

Figure 2: a) The initially set up model and initially trained SOM, b) the initial model set to the trained SOM, c) and d) the model after some animation steps.

Finally, the remaining data with a lot of markers being occluded (every once a while up to only four markers were seen) was used for adapting the SOM. This data even contained situations where the whole left arm data was missing for approximately 2 seconds. This was a very tough condition for the algorithm. Markers just disappeared and popped up on completely different positions. Unfortunately, the SOM is not very well fitted to track a lot of missing markers over a longer period of time, and, for human motion, even 2 seconds are a long period of time. During this time, the SOM tried to adapt to the other markers and completely lost track on its corresponding, but missing marker. It can even happen that other neurons adapt strongly to a reappearing marker, because in the mean-while it moved to their position and they also lost track to their corresponding markers.

Concluding, our approach is very well suited for situations where only a few markers (one or two markers on consecutive joints in a chain) are occluded or if occlusion happens for only a few frames. It can easily adopt to the posture and identify the markers 100% correctly. However, for capture data with a lot of occluded markers (e.g. a complete arm) our approach seems to be overstrained. The SOM seems to be inappropriate for such situations. Still, we believe that this approach can help to identify markers for slightly noisy human motion data.

References

[1] K. Dorfmüller-Ulhaas. Robust Optical User Motion Tracking Using a Kalman Filter. Technical Report 2003-6, University of Augsburg, Institut für Informatik, Universitätsstr. 2, D-86159 Augsburg, May 2003.

[2] J. F. O'Brien, R. Bodenheimer, G. Brostow, and J. K. Hodgins. Automatic Joint Parameter Estimation from Magnetic Motion Capture Data. In *Graphics Interface*, pages 53 – 60, 2000.

[3] M.-C. Silaghi, R. Plänkers, R. Boulic, P. Fua, and D. Thalmann. Local and Global Skeleton Fitting Techniques for Optical Motion Capture. In *CAPTECH '98: Proc. International Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, pages 26–40, 1998.