ACQUIRING 3D FULL-BODY MOTION FROM NOISY AND AMBIGUOUS INPUT

A Dissertation

by

HUI LOU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Computer Science

ACQUIRING 3D FULL-BODY MOTION FROM NOISY AND AMBIGUOUS INPUT

A Dissertation

by

HUI LOU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Computer Science

ABSTRACT

Acquiring 3D Full-body Motion from
Noisy and Ambiguous Input. (May 2012)

Hui Lou, B.S., University of Science and Technology of China
Chair of Advisory Committee: Dr. Jinxiang Chai

Natural human motion is highly demanded and widely used in a variety of applications such as video games and virtual realities. However, acquisition of full-body motion remains challenging because the system must be capable of accurately capturing a wide variety of human actions and does not require a considerable amount of time and skill to assemble. For instance, commercial optical motion capture systems such as Vicon can capture human motion with high accuracy and resolution while they often require post-processing by experts, which is time-consuming and costly. Microsoft Kinect, despite its high popularity and wide applications, does not provide accurate reconstruction of complex movements when significant occlusions occur. This dissertation explores two different approaches that accurately reconstruct full-body human motion from noisy and ambiguous input data captured by commercial motion capture devices.

The first approach automatically generates high-quality human motion from noisy data obtained from commercial optical motion capture systems, eliminating the need for post-processing. The second approach accurately captures a wide variety of human motion even under significant occlusions by using color/depth data captured by a single Kinect camera. The common theme that underlies two approaches is the use of prior knowledge embedded in pre-recorded motion capture database to reduce the reconstruction ambiguity caused by noisy and ambiguous input and constrain the solution to lie in the natural motion space. More specifically, the first approach constructs a series of spatial-temporal filter bases from pre-captured human motion data and employ them along with robust statistics techniques to filter noisy motion data corrupted by noise/outliers. The second approach formulates the problem in a Maximum a Posterior (MAP) framework and generates the most likely pose which explains the observations as well as consistent with the patterns embedded in the pre-recorded motion capture database. We demonstrate the effectiveness of our approaches through extensive numerical evaluations on synthetic data and comparisons against results created by commercial motion capture systems. The first approach can effectively denoise a wide variety of noisy motion data, including walking, running, jumping and swimming while the second approach is shown to be capable of accurately reconstructing a wider range of motions compared with Microsoft Kinect.

DEDICATION

To All Whom I Love

ACKNOWLEDGMENTS

Sometimes it is hard to believe that it has been five years and a half since I first came to study at Texas A&M University. This journey is coming to an end. A lot of things happened during this period and my feelings are very complicated at this special moment. It is time to make a conclusion and welcome a new future. All in all, I am very proud of myself for what I have decided and accomplished. Most importantly, I am very grateful to a lot of people who have supported me along the way.

First, I would like to thank my academic advisor, Professor Jinxiang Chai, for his constant support, guidance, without which this dissertation would have been impossible. Throughout the past years, I have gained profound understanding of a lot of things, including and far beyond research topics. I have become a much more mature, stronger, and more confident person and Dr. Chai is well deserving of the credits. I would also like to thank all my committee members, Professor Ergun Akleman, Professor John Keyser and Professor Dezhen Song, for their valuable time, insightful feedback and support all the time. Besides, the classes I have taken from Professor Keyser and Professor Song have greatly inspired my work.

I enjoy the experience of being a teaching assistant a lot, and I would love to thank all the Professors I have worked with, who have made it wonderful, enjoyable and rewarding. They are Dr. Jinxiang Chai, Dr. Walter Daugherity, Dr. Donald Friesen, Dr. Teresa Leyk, Dr. Scott Schaefer and Dr. Ronnie Ward. I would also like to take this opportunity to express my gratitude to the Department of Computer Science and Engineering at Texas A&M University especially for the generous financial support.

I am also very grateful to my lab mates Yenlin Chen, Jianyuan Min and Xiaolin Wei for their generous help, support and the laughter they have brought to me. They are very smart and diligent and I have learned a lot from them as a peer.

My Ph.D. life would be incomplete without mention of my dear friends who have helped me in various ways. They are like colors and have made my life colorful and interesting. In addition to my lab mates, they are Shu Du, Lei He, Ruoguan Huang, Huajun Liu, Xiaoyong Li, Tingting Ma, Yanan Tao, Saira Viqar, Lei Wang, Qing Xing, Wenjun Zhao etc.

Last, my deepest gratitude goes to my lovely family and my beloved fiancé Yang Qin. They are always by my side throughout all the tears and joys, believing in me and wishing me the best. I would never have gone this far without their encouragement, comfort and love. I will cherish them in my heart forever.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

Natural human motion data has been widely used in a lot of areas including education, sports, video games, virtual realities and animation. There have been great advances in human motion modeling, acquisition and synthesis in the past few decades. Available popular commercial motion capture systems include inertial systems (Fig. 1.1 (a)), magnetic systems (Fig. 1.1 (b)), optical systems (Fig. 1.1 (c)) and Microsoft Kinect (Fig. 1.1 (d)). While modern motion capturing technologies such as optical and magnetic capturing systems enable high-accuracy capturing with high frame rates, the capturing systems are usually expensive, intrusive, cumbersome and restrictive. Moreover, they often require hours of manual post-processing, which is not only time-consuming, but also error-prone. So, acquiring natural human motion data in inexpensive and non-obtrusive ways has been a long-standing challenging problem in the graphics and animation community. Furthermore, a practical system must be capable of capturing a wide variety of motions, which is also an important problem attracting numerous research efforts.

This dissertation explores motion reconstruction approaches from noisy and ambiguous input data captured by two commercial motion capture devices, Vicon and Kinect (Fig. 1.1 (c) and (d)). The first approach to be presented in Section 1 automatically reconstructs high-quality human motion from corrupted motion input (containing outliers, noise or missing data) obtained from Vicon optical capturing system [1] before post-processing, thereby saving significant amount of time and effort. The approach in Section 2 aims to accurately reconstruct a wide range of motion from monocular color/depth sequences obtained from a single Kinect camera [2],

This dissertation follows the style of *IEEE Transactions on Visualization and Computer Graphics*.

which is even more challenging due to camera noise and especially self-occlusions arising from monocular view.

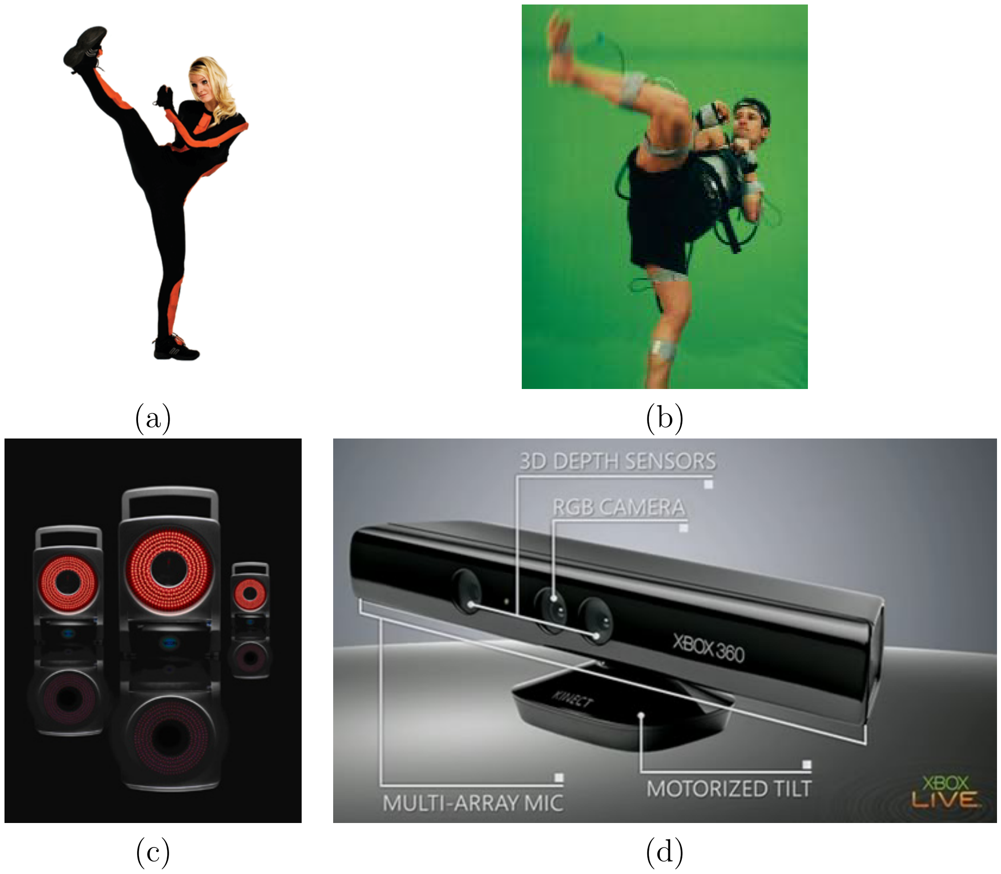


Fig. 1.1. Commercial motion capture systems: (a) Xsens inertial motion capture system [3]; (b) Ascension magnetic motion capture system [4]; (c) Vicon optical motion capture system [1]; (d) Microsoft Kinect camera [2].

A common challenge facing the two work explored in this dissertation is that the input signals from both systems are noisy and ambiguous. More specifically, motion capture data from optical motion capture systems like Vicon [1] frequently contain missing data due to occlusions and outliers caused by matching ambiguities between 2D markers. Non-optical commercial motion capture systems like inertial systems [3] or magnetic systems [4] do not have occlusion problems but the obtained data are often very noisy. Kinect sensors [2] are low-cost, however, the data from Kinect sensors are very noisy and often contain missing data due to self-occlusions. It is also ambiguous because 3D reconstruction requires associating each pixel with corresponding bone segment.

The fundamental reason why the problems are so challenging is that human movements are high dimensional and often cannot be accurately determined by noisy and ambiguous inputs from current motion capture systems. Human body is very complex, even a simplified human model usually contains at least 30 DOFs. However, the DOFs are not completely independent of each other. Actually quite the contrary, human motions are highly coordinated movements. For example, arms move in a certain way along with legs' movements in a walking motion; both legs move similarly in a jumping motion. Besides, consecutive frames exhibit temporal coherence. To model and utilize the patterns of natural human motion, we take advantage of pre-recorded motion capture databases by learning statistical models, which is the common theme underling both approaches to be presented in this dissertation. We use Vicon, a commercial optical motion capture system, to record high-quality motion databases, which contains various styles of motions, such as walking, running, jumping, dancing, boxing, bending, turning etc. We increase the variations of a specific motion style by recording the same motion multiple times, for example in different speeds, directions or moods as needed. Statistical models learned from the databases constrain the search space to natural human motions. We have demonstrated the effectiveness of data-driven priors in both systems of this dissertation.

The goals of the first work include filtering noise, removing outliers and filling in missing data. Human motion denoising is challenging because human motion involves highly coordinated movement, and the movements between different degrees of freedom are not independent. Standard signal denoising techniques such as Gaussian low-pass filter and Kalman filter [5] often process each degree of freedom independently. Therefore, motions filtered by standard filtering techniques often appear uncoordinated or unnatural because they cannot preserve the spatial-temporal characteristics embedded in natural human motion. The problem becomes even more complicated when captured motion data is corrupted by a certain percentage of outliers or missing values. In Section 1, we propose a novel data-driven technique for human motion denoising. The proposed system not only filters corrupted motion data but also keeps spatial-temporal patterns embedded in natural human motion data.

The second work is closely related to vision-based motion tracking and reconstruction [6–9], which has been an active research area for decades mainly because its potential in providing an inexpensive and non-obtrusive way to obtain human motion. Despite countless efforts and approaches from lots of researchers, it remains an open problem due to the significant challenges e.g. inherent depth ambiguity from a single view, occlusions, cloth/skin deformation, and illumination changes.

One popular approach for vision-based tracking and motion reconstruction is model-based approach, which approximates human body as a skeleton model parameterized in joint angle space. At each time step, a pose configuration is estimated which matches image measurements as well as being consistent with the previously estimated pose. Common image measurements include intensities, edges and silhouettes. Model-based sequential tracking requires pose initialization at the first frame and is subject to error accumulations. With the recent advances in sensing technologies, there has been more and more research on utilizing depth cameras due to their outstanding advantages, such as 3D information and low-cost. The ability of providing 3D data makes monocular tracking using depth cameras very practical. However, it remains a challenging problem due to strong noise in depth data and self-occlusions arising from monocular data. In Section 2, we propose an approach which effectively handles self-occlusions by taking advantage of prior knowledge embedded in pre-recorded motion capture database.

1.1 Contributions

This dissertation explores two data-driven approaches to reconstruct full-body human motion from noisy and ambiguous input data captured by commercial motion capture devices, Vicon and Kinect. The effectiveness of both approaches is demonstrated through extensive numerical evaluations on synthetic data and comparisons against results created by commercial motion capture systems. More specifically, the contributions are as follows:

Contributions of the first approach:

1. A data-driven filtering process for transforming corrupted motion capture data obtained into high-quality motion data.
2. A spatial-temporal statistical motion model for effective motion modeling and filtering.
3. A unified filtering framework that utilizes the constructed statistical motion models to filter noise, remove outliers, and fill in missing data.

4. An efficient optimization procedure for finding the optimal solution of a data-driven filtering problem.

Contributions of the second approach:

1. An online motion reconstruction process that sequentially reconstructs 3D skeletal poses using color/depth sequences obtained from a single Kinect camera.
2. A unified Maximum a Posterior (MAP) motion estimate framework that combines depth data, color information and 3D edges with human motion priors for 3D skeletal pose estimation.
3. An efficient optimization procedure for finding a MAP solution to our 3D skeletal pose reconstruction problem.

1.2 Organization

This dissertation is organized as follows. In Section 1, we describe the details of example-based human motion denoising on inputs from Vicon. Then, in Section 2, we present our work on reconstructing 3D full-body human movements from a combination of color and depth information captured by a single Kinect camera. In the end, we conclude the dissertation and discuss some future work.

2. EXAMPLE-BASED HUMAN MOTION DENOISING

One of the most popular and successful approaches for creating natural-looking human animation is to use motion capture data. A recent notable example of motion capture data is the movie *Beowulf*, where prerecorded motion data were used to animate all of the characters in the film. Meanwhile, in the animation community, a number of researchers have explored how to edit, transform, interpolate, retarget and recompose captured motion data to achieve new tasks.

All of these exciting applications and developments start with an accurate acquisition of high-quality human motion data. However, even with high-fidelity and expensive motion capture equipment, motion capture data may still contain noise and outliers that must be removed before further processing. For example, motion data recorded by commercial optical systems such as Vicon [1] often includes outliers and missing data, due to marker occlusions and mislabeling. Motion data captured by inertial or magnetic systems is often corrupted by sensor noise, output drifting or environment disturbances. However, the post-processing of noisy data often requires manual user editing, which is not only time-consuming, but also error-prone.

Human motion denoising is challenging because human motion involves highly coordinated movement, and the movements between different degrees of freedom are not independent. Standard signal denoising techniques such as Gaussian low-pass filter and Kalman filter [5] often process each degree of freedom independently. Therefore, motions filtered by standard filtering techniques often appear uncoordinated or unnatural because they cannot preserve the spatial-temporal characteristics embedded in natural human motion. The problem becomes even more complicated

©2010 IEEE. Reprinted, with permission, from H. Lou and J. Chai, "Example-Based Human Motion Denoising," *IEEE Transactions on Visualization and Computer Graphics*, pp. 870–879, 2010.

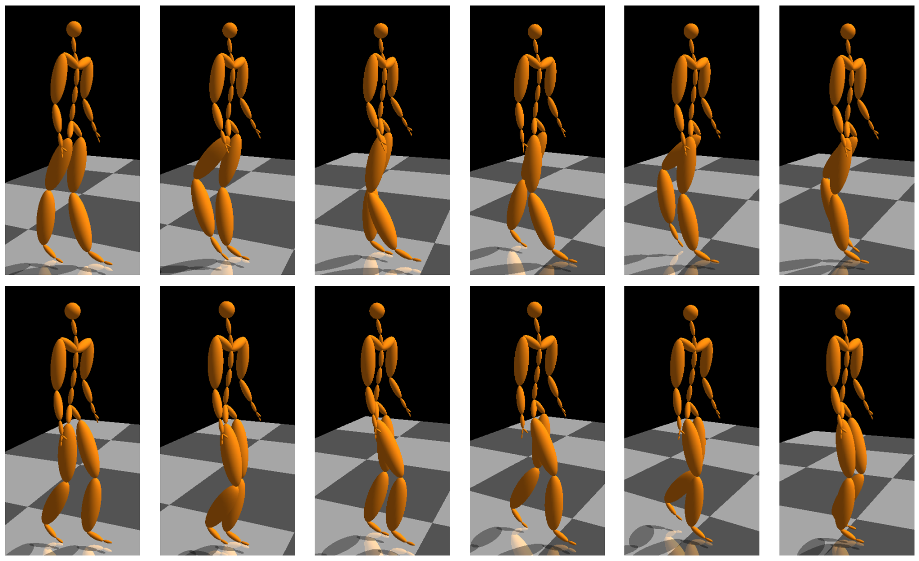


Fig. 2.1. Example-based human motion denoising: (top) input motion data corrupted by outliers (knee joint); (bottom) filtered motion data.

when captured motion data is corrupted by a certain percentage of outliers or missing values.

This work proposes a novel data-driven technique for human motion denoising. The proposed system not only filters corrupted motion data, including noise reduction, outlier removal and missing data completion, but also keeps spatial-temporal patterns embedded in human motion data (see Fig. 2.1). The key idea of our approach is to automatically learn a series of spatial-temporal filter bases from pre-recorded human motion data and use them to filter corrupted human motion data. Mathematically, we formulate the motion denoising problem in a nonlinear optimization framework. Our objective function measures the residual between the noisy input and the filtered motion in addition to how well the filtered motion preserves spatial-temporal patterns embedded in natural human motion. Optimizing the objective function generates high-quality human motion data which is "closest" to the input data.

We demonstrate the performance of our system by testing the algorithm on real and simulated motion data. We show how the system can effectively denoise a wide variety of noisy motion data, including walking, running, jumping and swimming. The quality of the filtered motions produced by our system highly depends on the percentage of outliers and the noise level in the noisy data. Therefore, we also evaluate how increasing or decreasing the percentage of outliers or the noise level influences the filtering results. We show the superior performance of our algorithm by comparing it with one of the most advanced commercial motion capture data processing software packages (Vicon Blade) and three baseline motion denoising techniques, including Gaussian filter, general Kalman filter and data-driven Kalman filter. Finally, we evaluate the performance of our algorithm in terms of filter bases learned from different databases and the size of the filtering window.

2.1 Background

This section briefly reviews related work in human motion denoising. Our algorithm is data driven–the system automatically learns a series of spatial-temporal filter bases from pre-captured motion data and uses them along with robust statistics techniques to filter noisy motion data. Consequently, we also discuss background in data-driven methods and robust statistics.

One popular approach for denoising human motion data is to apply linear time-invariant filters to noisy motion data [10–14]. For example, Lee and Shin [10] formulated rotation smoothing as a non-linear optimization problem and derived smoothing operators from a series of fairness functionals defined on orientation data. Fang et al. [11] applied a low-pass filter to the estimated angular velocity of an input signal to reconstruct a smooth angular motion by integrating the filter responses. Lee and Shin [13] presented a linear time-invariant filtering framework for filtering orientation data by transforming the orientation data into their analogues in a vector space, applying a filter mask on them, and then transforming the results back to the orientation space. Similar low-pass filters have also been implemented in commercial motion capture packages such as Vicon Blade [14].

An alternative solution is to use the Kalman filter framework [5] to sequentially filter noise present in human motion data [15, 16]. Shin and his colleagues [15] applied the Kalman filter to transform the movements of a performer recorded by an online optical motion capture system to an animated character in real-time. Tak and Ko [16] adopted an unscented Kalman filter framework and used it to convert a sequence of human motion data into a physically plausible motion sequence.

Unlike previous work, our denoising process is data-driven because the filter bases are automatically constructed from prerecorded human motion data. One key advantage of the data-driven denoising approach is the preservation of spatial-temporal patterns embedded in natural human motion data. It also enables us to detect and remove outliers and fill in missing values.

Several researchers have also explored techniques that transform human motion data into physically correct motion [17, 18] or expressive animation [19]. For example, Yamane and Nakamura [17] introduced a dynamics filter to convert a physically infeasible source motion sequence into a feasible one. Recently, Wang and his colleagues [19] presented a cartoon animation filter that takes an arbitrary input motion signal and modulates it in such a way that the output motion is more alive or exaggerated. However, none of the systems have attempted to process human motion data corrupted with outliers, noise and missing values.

Our work builds on the success of spatial-temporal human motion modeling for human motion synthesis and processing. Recent efforts have focused on constructing various motion models from pre-captured motion data and applying them in such applications as motion synthesis [20–22], inverse kinematics [23, 24], motion compression [25], motion quantification [26], motion registration [27], and footskate detection [28]. Our work is different because we construct a series of spatial-temporal filters with multi-channel singular spectrum analysis (M-SSA) [29, 30] and use them for human motion denoising. M-SSA has been successfully applied to many practical problems in geophysics [30]. In this work, we significantly extend the idea of M-SSA to human motion data modeling and filtering.

To filter noisy motion data corrupted by outliers, we apply robust estimators [31, 32] to measure the residual between filtered motion and noisy input. Robust statistics have also been applied to deal with outliers in graphics problems such as 3D mesh smoothing [33] and surface reconstruction [34].

2.2 Human Motion Denoising

The goal of this work is to develop a motion denoising algorithm for simultaneously filtering noise and outliers in input human motion data. To achieve this goal, we propose to construct a series of filter bases from prerecorded human motion data and use them along with robust statistics techniques for human motion denoising.

In this section, we first explain how to construct a series of filter bases from pre-captured motion data (Section 2.2.1), and then discuss how to deal with outliers from noisy motion data (Section 2.2.2). We formulate the motion denoising problem in a nonlinear optimization framework and define the object function for human motion denoising (Section 2.2.3). We develop an iterative algorithm for efficient optimization of the object function (Section 2.2.4). Finally, we discuss how to extend the framework to filling in missing values (Section 2.2.5).

2.2.1 Construction of Filter Bases

The key idea of our approach is to construct a series of filter bases that capture spatial-temporal patterns embedded in natural human motion. We model a series of filter bases using multi-channel singular spectrum analysis (M-SSA) [29, 30], which is based on the use of the Singular-value decomposition of the trajectory matrix obtained from training examples (time series data) by the method of delay. Applying the M-SSA to pre-captured human motion data enables us to identify a set of orthogonal spatial-temporal patterns embedded in natural human motion data. For simplicity's sake, we focus our description on constructing the M-SSA from one motion sequence.

Let $\{x_l(t) : l = 1, ..., L; t = 1..., N\}$ denote a sequence of prerecorded human motion data used for training, where L is the number of degrees of freedom to define a character pose and N is the number of the total frames. Let $\mathbf{x}_t = [x_1(t), ..., x_L(t)]^T$ represent a full-body character pose at frame t. Similarly, let $\{y_l(t) : l = 1, ..., L; t = 1..., T\}$ be a sequence of noisy input motion data and let $\mathbf{y}_t = [y_1(t), ..., y_L(t)]^T$ denote a character pose at frame t.

We first form a channel-specific trajectory matrix X_l by augmenting each channel $\{x_l(t) : t = 1, ..., N\}$ of the training data with S_N lagged copies of itself:

$$X_l \quad = \quad \begin{pmatrix} x_l(1) & x_l(2) & \dots & x_l(M) \\ x_l(2) & x_l(3) & \dots & x_l(M+1) \\ \cdot & \cdot & \dots & \cdot \\ x_l(S_N) & x_l(S_N+1) & \dots & x_l(N) \end{pmatrix}, \tag{2.1}$$

where M is the size of the lagged windows and $S_N = N - M + 1$ is the total number of the lagged windows across the entire training sequence. This is a standard procedure in time series analysis [35], which essentially transfers a one-dimensional time series $\{x_l(t) : t = 1, ..., N\}$ into a multi-dimensional time series $\{x_l^m(i) : i = 1, ..., S_N\}$ with vectors $x_l^m(i) = (x_l(i), ..., x_l(i + M - 1))' \in \mathcal{R}^M$. Each vector $x_l^m(i)$ characterizes the temporal structure of the original time series and M is chosen large enough to extract this temporal information.

We further form a fully augmented trajectory matrix:

$$D \quad = \quad \begin{pmatrix} X_1 & X_2 & \dots & X_L \end{pmatrix}. \tag{2.2}$$

To find the spatial-temporal patterns in degrees of freedom of human motion data, we compute a grand lag covariance matrix C_D as follows:

$$C_D \quad = \tfrac{D^T D}{S_N} \quad = \quad \begin{pmatrix} C_{1,1} & C_{1,2} & \dots & C_{1,L} \\ \cdot & C_{2,2} & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ C_{L,1} & C_{L,2} & \dots & C_{L,L} \end{pmatrix}, \tag{2.3}$$

where the blocks of C_D are given by

$$C_{l,l'} = \frac{X_l^T X_{l'}}{S_N}, \quad l, l' = 1, ..., L, \tag{2.4}$$

with entries

$$(C_{l,l'})_{j,j'} = \frac{\sum_{n=1}^{S_N} x_l(n+j-1)x_{l'}(n+j'-1)}{S_N}, \quad j, j' = 1, ..., M. \tag{2.5}$$

The covariance matrix $C_{l,l'}$ computes the covariance between trajectories of the l-th dof and the l'-th dof within a finite size (M) window. The grand lagged covariance matrix C_D is a symmetric $M \times L$ by $M \times L$ matrix, and it encodes spatial-temporal correlations of all degrees of freedom within a finite size window M. For example, the quantity $(C_{3,4'})_{1,2'}$ encodes the correlation between the 3-rd dof of the 1-st frame and the 4-th dof of the 2-nd frame within the window.

The key idea of the M-SSA is to apply the singular value decomposition (SVD) technique to the grand lag covariance matrix C_D and extract the spatial-temporal patterns embedded in the captured motion data. More specifically, we use the SVD technique to diagonalize the grand lagged covariance matrix and yield $M \times L$ eigen-vectors $\{\mathbf{e}^k \in R^{M\times L} | k = 1, ..., M \times L\}$. The $M \times L$ eigen-vectors provide a set of orthogonal filter bases, which can be used to reconstruct any segment of human motion data within a finite size window M:

$$\mathbf{x}_{1:M} = \sum_{k=1}^{M\times L} < \mathbf{e}^k, \mathbf{x}_{1:M} > \mathbf{e}^k, \tag{2.6}$$

where the vector $\mathbf{x}_{1:M} \in R^{M\times L}$ sequentially stacks all poses across the entire window and the operator $<>$ represents the dot product between two vectors. In practice, human movement is highly correlated. Therefore, a small number of patterns are

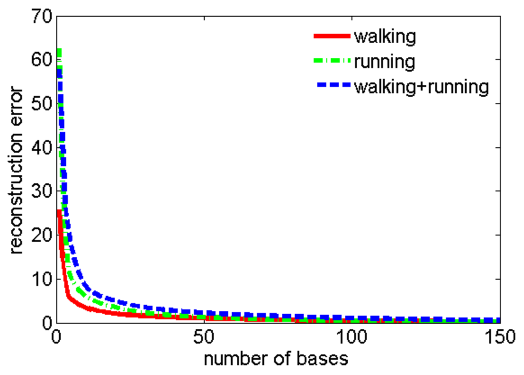


Fig. 2.2. Reconstruction errors vs. the number of filter bases used within a finite size (20) window. The reconstruction errors are evaluated via cross-validation techniques.

often sufficient to represent natural human motion variation within a finite size window. Fig. 2.2 shows that for human walking data, 50 filter bases might be sufficient to model spatial-temporal variation within a window of size 20.

The extracted orthogonal eigen-bases are conceptually similar to "sine" and "cosine" waves in Fourier analysis. As a result, we can design similar "low-pass" filters with orthogonal bases $\mathbf{e}^k$. This motivates us to design the following data-driven filter:

$$\begin{aligned} \mathbf{z}_{1:M} &= \textstyle\sum_{k=1}^{K} c_k \mathbf{e}^k \\ &= \textstyle\sum_{k=1}^{K} < \mathbf{e}^k, \mathbf{y}_{1:M} > \mathbf{e}^k, \end{aligned} \tag{2.7}$$

where the vector $\mathbf{z}_{1:M} \in R^{M \times L}$ sequentially stacks all poses of the filtered motion across an entire window. The coefficients $c_k, k = 1, ..., K$ are the filter weights. The cutoff threshold K is similar to cutoff frequency used in standard filters. In our experiment, we automatically determine the cutoff threshold by keeping 99% of the original motion variation.

As we've discussed earlier, the filter bases can be easily constructed by applying the singular-value decomposition technique to the grand lag covariance matrix C_D.

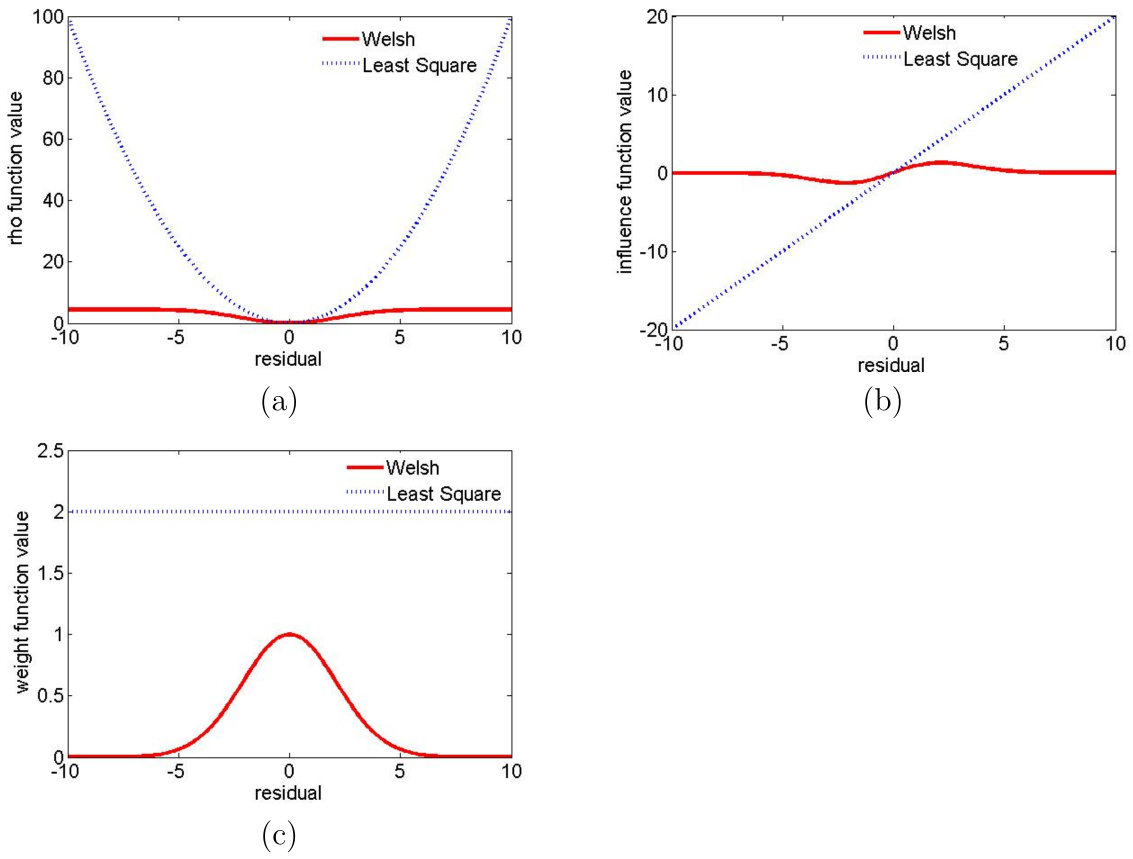


Fig. 2.3. The robust estimator (Welsch function) vs. the least-square estimator: (a) the function for measuring the residual distance (r) between the reconstructed motion and noisy measurement ($\rho(r)$); (b) the influence function for each data point ($\phi(r) = {}^{\partial\rho(r)}/_{\partial r}$); (c) the weights for each data point (${}^{\phi(r)}/_{r}$). Note that the weights for outliers become zero for robust estimators.

The whole learning process runs very fast. For example, it takes about two minutes to construct filter bases of a finite size window (20) from a training database of 2000 frames with our Matlab implementation.

2.2.2 Dealing with Outliers

Real motion data from optical motion capture systems often contains outliers due to marker occlusions and mislabeling. However, the data-driven filter described in Equation (2.7) will not work well for outliers because least-square solutions (*i.e.* the dot product between the input data and eigen patterns) give too much influence to outliers, and a single degree of freedom with a large error (an outlier) will deteriorate the solution dramatically. One effective way to deal with outliers is to use robust estimators to reduce the influence of outliers [31, 32].

We measure the distance between the filtered data $z_l(t)$ and the noisy data $y_l(t)$ with robust estimator ρ:

$$E_{data}(\mathbf{z}_{1:T}, \mathbf{y}_{1:T}) = \sum_t \sum_l \rho(z_l(t) - y_l(t)), \tag{2.8}$$

where E_{data} measure the distance between the noisy input and the filtered motion. The robust estimator ρ is a function of the residual $r_l(t) = z_l(t) - y_l(t)$ between the noisy data and reconstructed data. The derivative of this function characterizes the bias that a particular measurement has on the solution. In the least-square case, the influence of data points increases linearly and is unbounded.

To increase robustness we only consider estimators for which the influences of outliers tend to zero (see Fig. 2.3). We choose the Welsch estimator but the treatment here could be equally applied to a wide variety of other estimators. A discussion of various estimators can be found in [31, 32].

Mathematically, the Welsch robust function is defined as follows:

$$\rho(r) = \frac{p^2}{2}(1 - \exp(-\frac{r^2}{p^2})), \tag{2.9}$$

where the scalar p is a parameter for the robust estimator and r is the residual between the measured data and reconstructed data, which equals to $z_l(t) - y_l(t)$ from equation (2.8). We experimentally set the parameter of Welsch estimator (p) to 3.

2.2.3 Objective Function

We now combine the data-driven filter bases $\mathbf{e}_k \in R^{M\times L}, k = 1, ..., K$ with the robust estimator $\rho(r)$ for robust filtering of noisy motion data. Mathematically, we formulate the whole denoising process in an optimization framework. We define an objective function by measuring the distance between the input motion $\mathbf{y}_t$ and the filtered motion $\mathbf{z}_t$ as well as how well the filtered motion $\mathbf{z}_t$ preserves the spatial-temporal patterns $\mathbf{e}_k, k = 1, .., K$ embedded in pre-captured motion data.

We denoise human motion data within a finite size window by solving the following unconstrained optimization problem:

$$\hat{\mathbf{c}}, \hat{\mathbf{z}}_{1:M} = \arg\min_{\mathbf{c},\mathbf{z}_{1:M}} \sum_{t=1}^{M}\sum_{l=1}^{L} \rho(z_l(t) - y_l(t)) + \lambda\|\mathbf{z}_{1:M} - E\mathbf{c}\|^2, \tag{2.10}$$

where the matrix $E = [\mathbf{e}_1 ... \mathbf{e}_K]$ stacks the filter bases $\mathbf{e}_k, k = 1, ..., K$ constructed from training data. The vector $\mathbf{c} = [c_1, ..., c_K]^T$ stacks all the filter weights. The first term, which is the robust estimation term defined in Equation (2.9), makes sure the filtered motion stays as "close" as possible to the input motion (while discarding outliers at the mean time). The second term is reformulated from the data-driven filter described in Equation (2.7), and measures how well the filtered motion matches the spatial-temporal patterns embedded in pre-captured motion data. The weight λ controls the importance of two terms. We experimentally set the weight to 0.1.

To filter a motion sequence $\mathbf{y}_{1:T}$ of the length T $(T > M)$, we slide a window of the size M throughout the motion and simultaneously compute the filtered motion across the entire sequence. After summing over the optimization functions of all the sliding windows, we have the following optimization problem:

$$\begin{aligned}\{\hat{\mathbf{c}}_s, \hat{\mathbf{z}}_t\} = \arg\min_{\{\mathbf{c}_s\},\{\mathbf{z}_t\}} \textstyle\sum_{t=1}^{T} \sum_{l=1}^{L} \rho(z_l(t) - y_l(t)) + \\ \lambda \textstyle\sum_{s=1}^{S} \|\mathbf{z}_{s,1:M} - E\mathbf{c}_s\|^2,\end{aligned} \tag{2.11}$$

where the vectors $\mathbf{c}_s$ and $\mathbf{z}_{s,1:M}$ are the filtering coefficient and the filtered motion for the s-th sliding window respectively. The quantity $S = T - M + 1$ is the total number of sliding windows used for data filtering. Similarly, the first term evaluates the distance between the input motion data $\mathbf{y}_{1:T}$, and the filtered motion data $\mathbf{z}_{1:T}$ across the entire sequence. The second term makes sure that the filtered motion preserves spatial-temporal patterns embedded in human motion data.

2.2.4 Iterative Optimization

The overall cost function (Equation 2.11) includes two groups of unknowns: the filtering coefficients $\mathbf{c}_s, s = 1, ..., T - M + 1$ for each sliding window, and the filtered motion $\mathbf{z}_{1:T}$ across the entire sequence. The total number of the optimization parameters is $K * (T - M + 1) + T * L$.

We have found that direct optimization of the cost function is not efficient, particularly when T is large. The system often runs out of memory and becomes very slow; the optimization is also prone to fall into local minima. To address these issues, we introduce an iterative optimization algorithm to decompose the large optimization problem into a series of small optimization problems that can be solved efficiently.

In each iteration, we keep one group of the unknowns constant and search for the optimal update for the other group of unknowns. More precisely, we initialize the filtered motion $\hat{\mathbf{z}}_{1:T}$ with the noisy data $\mathbf{y}_{1:T}$. We then iteratively update the filtered motion and the filtering coefficients until the solution converges: (i). keep the filtered motion $\hat{\mathbf{z}}_{1:T}$ constant and seek the optimal filtering coefficients $\mathbf{c}_s$; (ii). keep the filtering coefficients $\mathbf{c}_s$ constant and update the filtered motion $\hat{\mathbf{z}}_{1:T}$. We discuss the two steps in more detail in the following section.

Weight Update

In this step, we keep the filtered motion $\hat{\mathbf{z}}_{1:T}$ constant and seek an optimal update of the filtering weights $\mathbf{c}_s$. Given the filtered motion $\hat{\mathbf{z}}_{1:T}$, we can estimate the optimal filtering coefficients $\mathbf{c}_s$ by decomposing the whole optimization function into $S = T - M + 1$ independent quadratic functions:

$$\hat{\mathbf{c}}_s = \arg\min_{\mathbf{c}_s} \|\hat{\mathbf{z}}_{s,1:M} - E\mathbf{c}_s\|^2, \quad s = 1, ..., S. \tag{2.12}$$

The quadratic objective functions have the following closed-form solution:

$$\{\hat{\mathbf{c}}_s\} \quad = \quad E^T \hat{\mathbf{z}}_{s,1:M}, \quad s = 1, ..., S. \tag{2.13}$$

Motion Update

After we update the filtering weights $\hat{\mathbf{c}}_s$, we keep them temporarily constant and use them to update the filtered motion $\hat{\mathbf{z}}_{1:T}$. This allows us to decompose the whole optimization function into the T small independent optimization functions:

$$\begin{aligned}\hat{\mathbf{z}}_t = \arg\min_{\mathbf{z}_t} \; & \lambda \textstyle\sum_{m=\max\{1,t-T+M\}}^{m=\min\{M,t\}} \|\mathbf{z}_t - E_m \hat{\mathbf{c}}_{t-m+1}\|^2 \\ & + \textstyle\sum_l \rho(z_l(t) - y_l(t)), \;\; t = 1, ..., T,\end{aligned} \tag{2.14}$$

where the vector $\mathbf{z}_t$ represents the filtered pose at frame $t, t = 1, ..., T$. The matrix E_m is an $L \times K$ matrix that stacks rows $M(l-1)+m, l = 1, ..., L$ of the matrix E. The min and max functions are used to deal with the first and the last $M-1$ frames, respectively.

The resulting optimization problem can be reformulated as an iteratively re-weighted least square (IRLS) problem [32, 36]. After applying IRLS to the objective function in Equation (2.14), we can iteratively solve the following weighted least-square problem:

$$\begin{aligned}\hat{\mathbf{z}}_t = \arg\min_{\mathbf{z}_t} \; & \lambda \textstyle\sum_{m=\max\{1,t-T+M\}}^{m=\min\{M,t\}} \|\mathbf{z}_t - E_m \hat{\mathbf{c}}_{t-m+1}\|^2 \\ & + \textstyle\sum_l w_l(t)(z_l(t) - y_l(t))^2, \;\; t = 1, ..., T,\end{aligned} \tag{2.15}$$

where the weights $w_l(t)$ can be computed by $w_l(t) = {}^{\varphi(z_l(t) - y_l(t))}/_{z_l(t) - y_l(t)}$ and $\varphi(.)$ is the derivative of the Welsch function $\rho(.)$

Iterative Optimization Procedure

Given an initial guess of the filtered motion, the motion denoising algorithm iteratively updates the filtered motion and the filtering weights across the entire sequence until the solution converges. The whole iterative procedure is outlined as follows:

1. Initialize $\hat{\mathbf{z}}^0_{1:T} = \mathbf{y}_{1:T}$
2. Update the coefficients $\hat{\mathbf{c}}_s$ and the motion $\hat{\mathbf{z}}_{1:T}$
    2.1. Update $\hat{\mathbf{c}}_s = E^T\hat{\mathbf{z}}_{s,1:M} \quad s = 1, ..., S$
    2.2. Update motion $\hat{\mathbf{z}}_{1:T}$ with IRLS techniques.
3. Repeat step 2 until the change of the cost function value is smaller than a user-defined threshold.

The decomposition of the large optimization problem into a series of small optimization problem significantly speeds up the optimization process. In our experiments, we have found that the algorithm typically converges after 10 to 20 iterations (less than five seconds).

Post Processing

Given an appropriate set of training data, the example-based motion denoising algorithm transforms noisy motion data into high-quality motion. However, the filtered motion might violate kinematic constraints imposed by the environment because the filter bases do not encode foot contact information in motion data. The most visible artifact is footskate. When this happens, the system uses the method in [37] to correct the footskate artifact.

2.2.5 Missing Data Fill-in

Motion capture data from optical motion capture systems often contains missing values due to marker occlusions. Our framework can easily be extended for filling in missing values in motion capture data. We assume that the index to missing data points is known in advance (this is often the case for optical motion capture systems). We use binary weights $\alpha_{l,t} \in \{0,1\}, l = 1,...,L; t = 1,...,T$ to indicate whether or not the observed data entries $y_l(t)$ contain missing data.

To reduce noise and remove outliers as well as fill in missing data, we solve the following optimization problem:

$$\begin{aligned} \{\hat{\mathbf{c}}_s, \hat{\mathbf{z}}_t\} = \arg\min_{\{\mathbf{c}_s\},\{\mathbf{z}_t\}} \textstyle\sum_{t=1}^{T} \sum_{l=1}^{L} \alpha_{l,t}\rho(z_l(t) - y_l(t)) \\ +\lambda \textstyle\sum_{s=1}^{S} \|\mathbf{z}_{s,1:M} - E\mathbf{c}_s\|^2. \end{aligned} \tag{2.16}$$

Similarly, we use the iterative procedure described in Section 3.4.3 to efficiently optimize the above objective function.

2.3 Results

The performance of our denoising algorithm has been evaluated with both real and simulated noisy data. We first evaluate the performance of our algorithm using real data captured by optical motion capture systems. Then we quantitatively assess the accuracy of our system with simulated noisy data and compare it with three baseline algorithms. The performance of our system highly depends on the filter bases and size of filtering windows. Therefore, we also evaluate how the database influences the resulting motion and how increasing or decreasing the window size influences the accuracy of the filtering algorithm. Our results are best viewed in the accompanying video although we show sample frames of a few results here.

We have tested our algorithm on a variety of human motion data, including walking, running, jumping, and swimming. Our algorithm works well for both joint angle data (.amc) and 3D marker position data (.c3d). In our experiments, all of the data were originally captured at 120 fps and then downsampled to 30 fps.

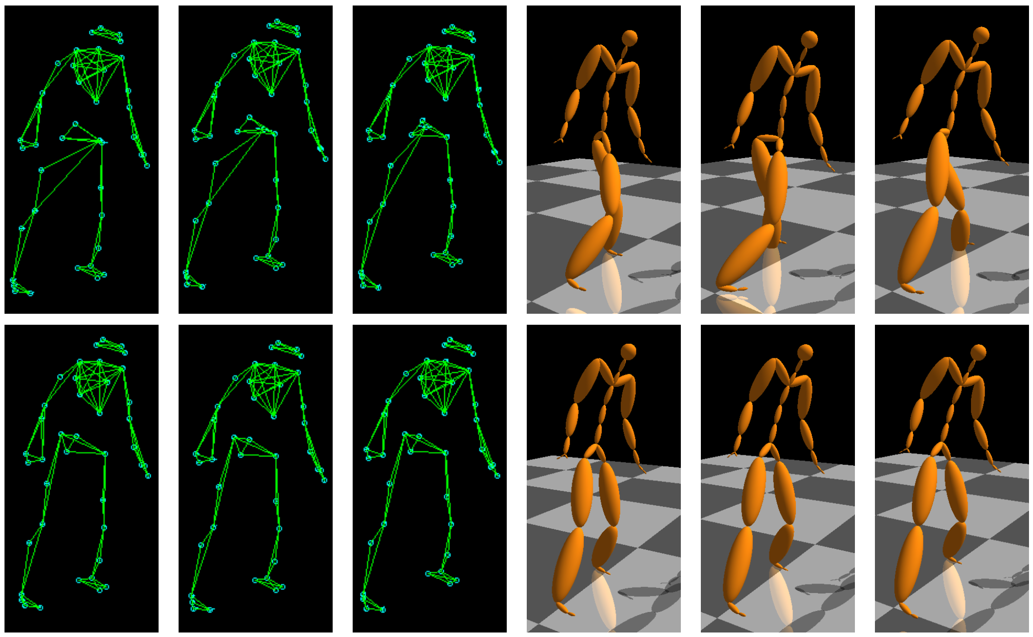


Fig. 2.4. Filtering real noisy motion data in 3D position space: (top left) the corrupted motion data in position space; (bottom left) the filtered motion data in position space; (top right) the corrupted motion data in joint-angle space; (bottom right) the filtered motion data in joint-angle space.

Our training database is behavior-specific and typically contains a small number of motion examples with different style variations. For example, the training data for walking contains five walking examples with different speeds and step sizes. We set the window size (M) to 20 frames. We automatically determine the number of bases (K) by keeping the energy to be at 99%, that is, the energy of the K largest eigen values is approximately 99% of the total energy.

2.3.1 Testing on Real Data

Due to occlusions and marker mislabeling, real motion capture data from Vicon systems often contain a certain percentage of missing values and outliers. Our algorithm can be used to filter noisy motion in both original marker position space (.c3d files) and joint-angle space (.amc). Fig. 2.1 and Fig. 2.4 show the filtering results in joint angle space and marker position space respectively.

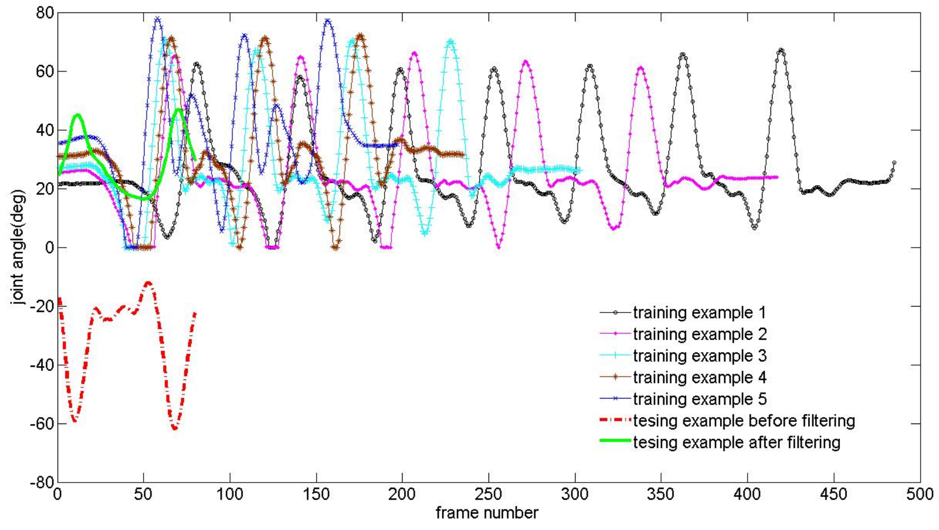
We have evaluated the performance of our algorithm on real motion data captured by optical motion capture systems (Vicon). Most corrupted motion capture data reported here were downloaded from the CMU online motion capture library[1]. For example, we constructed the filter bases from two online motion files "83-04.c3d" and "83-55.c3d" and then use them to denoise one corrupted motion sequence "83-68.c3d". All three files are from the same subject (No. 83). Our experiments show that our algorithm can filter motions that are not in the training database. Fig. 2.5 visualizes joint angle data (knee and thorax) of the training examples, the input noisy motion data, and the filtered motion data, respectively. Note that the filtered motion has a different phase and scale from the training data set.

We also tested our algorithm on completing missing data in both joint angle and marker position space. Fig. 2.6 shows sample frames of our results, where all the markers on both arms are completely missing.

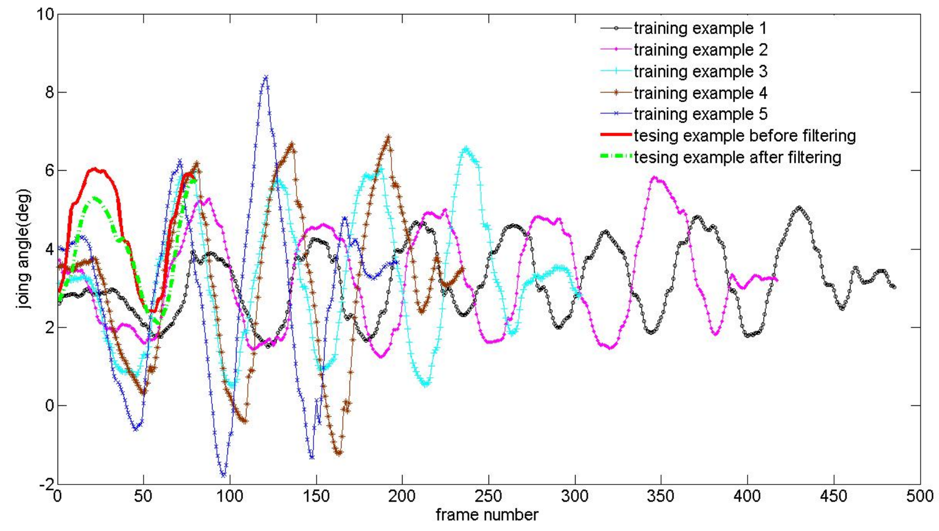
2.3.2 Testing on Simulated Data

We have quantitatively assessed the performance of our algorithm by comparing it with three baseline human motion denoising techniques: Gaussian filter, the simple general Kalman filter [5], and the data-driven Kalman filter. We applied a standard Gaussian filter to every degree of freedom independently. We experimentally set the window size to 11. We also implemented two types of Kalman filter. For the first one, we set its system matrices to identity matrices by simply choosing the

[1]http://mocap.cs.cmu.edu



(a)



(b)

Fig. 2.5. Motion generalization: (left) joint angle values (knee) of training data set, noisy motion data and filtered motion data; (right) joint angle values (thorax) of training data set, noisy motion data and filtered motion data. Note that knee joint is corrupted by outliers across the entire motion while the thorax joint is not corrupted by any outliers.

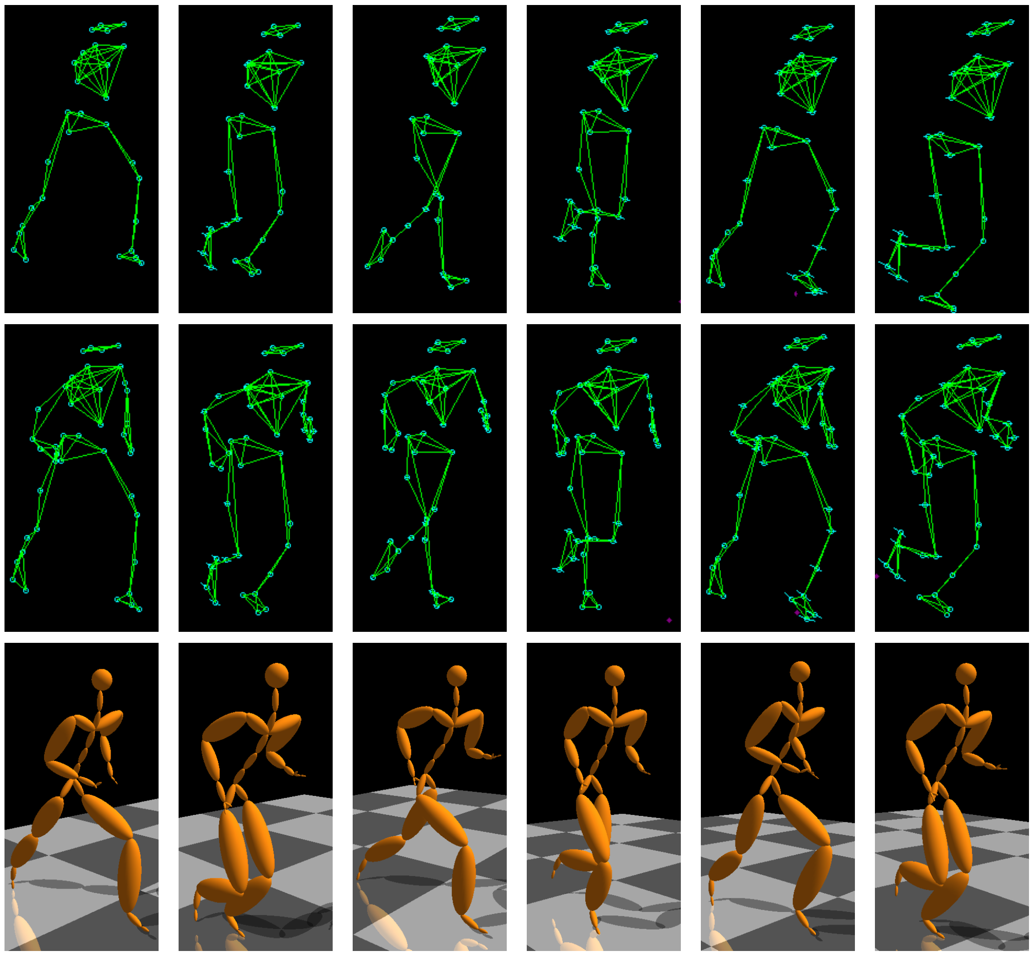
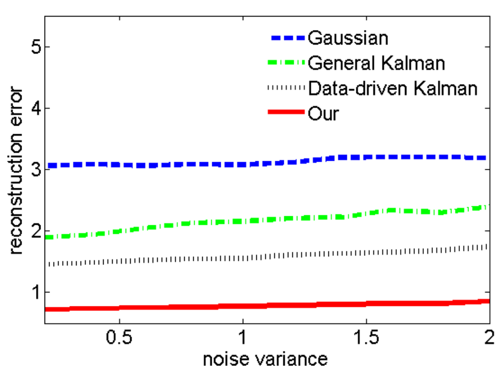


Fig. 2.6. Filling in missing data in the 3D position space: (top) missing motion data in 3D position space; (middle) completed motion data in 3D position space; (bottom) completed motion data in joint-angle space.



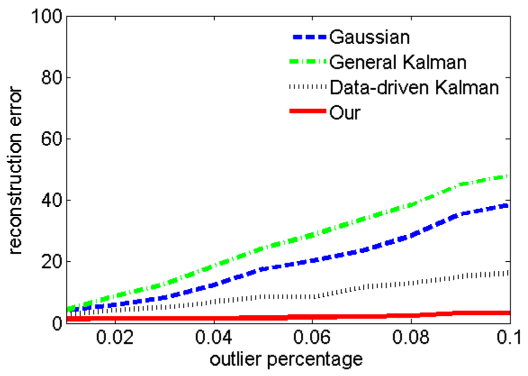


Fig. 2.7. Comparisons with alternative signal denoising techniques: (left) reconstruction errors vs. different noise levels; (right) reconstruction errors vs. different percentages of outliers.

prediction model as follows: $z_l(t+1) = z_l(t) + N(0, \sigma)$. The second Kalman filter is a simple data-driven algorithm, which learns the system matrices from the same set of training data as used in our algorithm.

Our evaluation is based on simulated noisy motion data corrupted by different percentages of outliers and different levels of Gaussian noise. We tested a number of trials on each setting and computed the average reconstruction errors measured by degrees per joint angle per frame. More specifically, we pulled testing motion sequences out of the training database and added simulated noise to the testing motion sequence. We then employed our algorithm as well as baseline algorithms to filter the simulated noisy motion data. The filtering error is computed by the average squared distance between the filtered motion data and ground truth data.

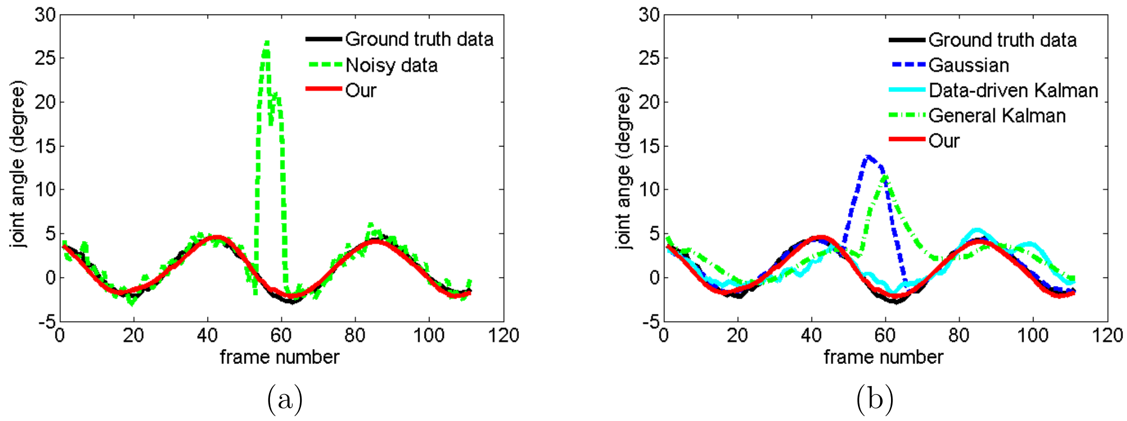


Fig. 2.8. Comparisons of joint angle values (thorax): (a) our result vs. ground truth data; (b) our result vs. results obtained by three baseline filtering algorithms.

Fig. 2.7 shows the comparison of four algorithms under various outlier percentages and noise levels. Our algorithm produces the best results for all testing scenarios. Fig. 2.8.(b) shows how both Gaussian filter and general Kalman filters fail to detect and remove outliers in the corrupted motion data. A simple data-driven Kalman filter cannot preserve spatial-temporal patterns in the input motion. Only our algorithm can filter noise and remove outliers while still keeping spatial-temporal patterns in the input motion.

2.3.3 Evaluation

The quality of the filtered motion depends on the filtering priors and the size of the filtering window. Therefore, we have designed several experiments to evaluate the performance of our algorithm.

Different filtering priors. We have evaluated the importance of filter bases by filtering the same set of noisy examples with priors learned from different motion databases. More specifically, we have tested on filtering noisy walking data with filter bases learned from a small walking data, a large walking database, and a mixed database which includes walking and running data. The small walking database includes 1624 frames of walking data recorded from the same motion capture subject as the testing data. The large walking database contains 60452 frames of walking data collected from 15 different subjects. The mixed database includes 45419 frames of walking and running data from 10 different subjects. The accompanying video shows the denoising results with different motion priors. We have observed that the system can produce good results for all three databases. However, the quality of the filtered motion becomes slightly worse when we use a large or mixed database to filter the noisy walking motion. This is due to the fact that spatial-temporal models are built on the entire database and larger databases tend to have more variations than necessary for the corrupted input motion, thereby lowering the quality of filtering. In addition to the good results on walking data, the system can produce a good result when we filter a noisy running sequence with filter bases learned from the mixed database.

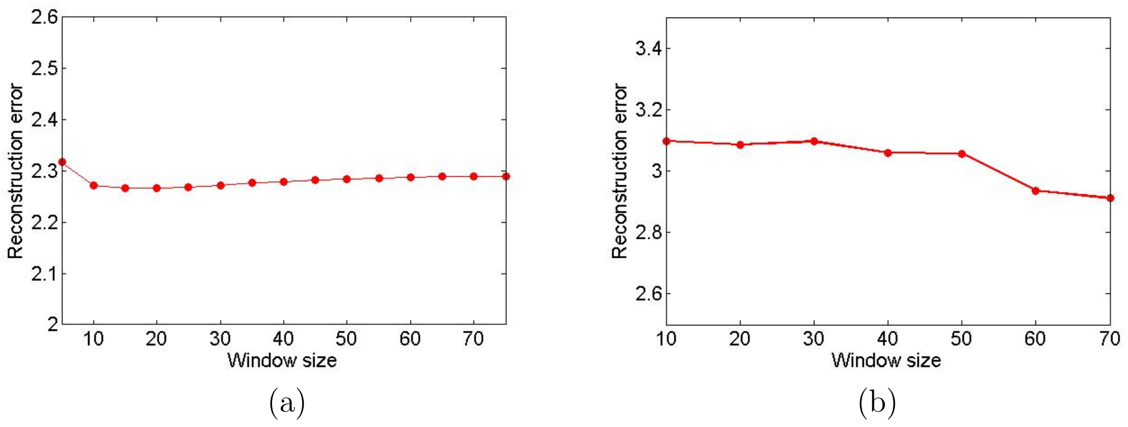


Fig. 2.9. Evaluation of the performance of the denoising algorithm with respect to different window sizes: (a) filtering a noisy walking sequence ($\sigma = 3$); (b) filling in missing values in a jumping sequence. The errors are measured by degrees per joint angle per frame.

Different window sizes. We quantitatively evaluated the effectiveness of our algorithm with respect to different window sizes (M). More specifically, we pulled a testing motion sequence out of the database, and added the simulated noise to the testing data. Next, we applied our filtering algorithm to transform the simulated noisy data into the filtered data, and compared the filtered data with ground truth data. In our experiments, we have observed that the influence of the window size on the performance of our algorithm is action-specific. For example, Fig. 2.9.(a) shows the filtering errors for a noisy walking sequence ($\sigma = 3$) with respect to different window sizes. The filtering errors are relatively large when the window size is smaller than 10. However, when the window size is larger than 10, the error becomes relatively insensitive to the window size. Fig. 2.9.(b) shows the filtering errors for filling in missing values in a jumping sequence with respect to different window sizes. Please note that there is a drop in reconstruction error for the jumping data around window size of 60, but not for the walking data. This can be explained by the fact that the jumping motion (180° jumping) has a longer cycle than normal walking motion, which therefore requires a larger window which needs to be large enough to better capture the temporal structure of the jumping motion, producing a smaller reconstruction error.

The accompanying video also shows a comparison with one of the most advanced motion capture softwares *Vicon Blade*. Our method produces better results than Vicon Blade. In particular when markers on both arms are missing throughout the motion, Vicon Blade fails to handle this case. In contrast, our algorithm successfully fills in the missing data on both arms across the entire sequence (see Fig. 2.6).

The system often fails to generate a good result if the training data does not contain any variations of the input motion. As shown in the accompanying video, the system produce poor results when we filter noisy walking data using the filter bases learned from a running database. Similarly, it is not appropriate to use a walking prior to fill in the missing values in a running sequence.

2.4 Discussion

We have presented an example-based algorithm for human motion data denoising. The key idea of our approach is to construct a series of filter bases from pre-captured human motion data and employ them along with robust statistics techniques to filter noisy motion data corrupted by outliers.

The constructed filter bases are similar in spirit to other filter bases used in signal processing community, such as “sine” waves, “cosine” waves, and “wavelet” bases. They are normalized and orthogonal to each other; a complete set of the filter bases can be used to uniquely reconstruct a segment of human motion data within a specific window size. One unique property of our filtering process is that it keeps important human motion patterns while discarding patterns that cannot be interpreted by training data. Therefore, the data-driven denoising filters can still keep high-frequency components of the input motion because frequent patterns may still contain high-frequency components. Another benefit of the data-driven filter bases is that it enables the system to detect outliers and fill in missing values.

Similar to other data-driven methods, one limitation of our approach is that an appropriate database must be available. We require the training data be "clean" (perceptually high-quality) and contain similar motion patterns of the input motion. Fig. 2.5 shows that our algorithm can filter an input motion sequence that is different from captured motion data. However, filtering the noisy input with completely different motion patterns (denoising walking data with running patterns, for instance) is not likely to yield reasonable results.

Our system has generated good results by filtering motion data containing different levels of noise and different percentages of outliers. We have observed that the denoising results deteriorate rapidly when the percentage of outliers is larger than 15%. However, we have not rigorously assessed when our system will break down.

In the future, we would like to explore how to construct spatial-temporal filter bases from the noisy data itself. It might be possible to extend our optimization framework to simultaneously estimate spatial-temporal filter bases $\mathbf{e}_k, k = 1, ..., K$, filtered motion $\hat{\mathbf{z}}_{1:T}$, and coefficients $\hat{\mathbf{c}}_s, s = 1, ..., S$ from noisy input data $\hat{\mathbf{y}}_{1:T}$. In addition, we plan to add joint angle limit constraints into the motion denoising framework in our future work.

3. ACQUIRING 3D FULL-BODY HUMAN MOTION USING A SINGLE KINECT DEVICE

Natural human motion is highly demanded and widely used in a variety of applications such as video games and virtual realities. However, acquisition of full-body motion at low-cost remains challenging because, meanwhile, the system must be capable of accurately capturing a wide variety of human actions. For instance, commercial optical motion capture systems such as Vicon, despite its high accuracy and frame rate, are very expensive, intrusive and require significant setup and cleanup time and efforts. Due to these limitations, a lot of work have been contributed to video-based motion capture [7–9], which does not require the use of tight suits or markers and also provides a cheap solution. Despite countless efforts from lots of researchers, it remains an open problem due to the significant challenges e.g. inherent depth ambiguity from a single view video, occlusions, cloth/skin deformation, and illumination changes. With the recent advances in sensor technologies, much further progress has been made by utilizing low-cost depth cameras to capture motion [38–41]. Microsoft Kinect [2] is a very successful example among these work. Despite its high popularity and wide applications, Microsoft Kinect does not provide accurate reconstruction of complex movements when significant occlusions occur. This paper explores an approach that accurately reconstructs a wide variety of 3D full-body human motion from monocular color/depth sequences captured by a single Kinect camera.

Different from the detection algorithm used in Microsoft Kinect, however, our approach is based on a simplified skeleton model and treats the problem as a model-based tracking problem. Human motion is advanced from time t to time t+1, given estimated initial pose at the first frame, and observations, including both color and depth data, at time t and $t+1$. At each time step, a pose configuration is estimated which matches color/depth measurements as well as being consistent with the prior knowledge. The problem is essentially formulated as a Maximum a Posterior (MAP) problem by combining observations and prior knowledge. Maximizing the posterior generates the most likely pose which explains the new observations and is also consistent with the patterns embedded in the pre-recorded motion capture database. Observed 3D points, containing both color and depth information, are utilized to measure the likelihood of a pose. Besides, 3D depth edges are introduced to enhance the quality of the measurement and proved to be effective in preventing error drifting particularly in the case of fast movements or large body parts partially occluded by small body parts (e.g. arms in front of the torso). However, there is another type of occlusion where body parts disappear mostly or completely for a short period time due to occlusions (e.g. one arm and one leg are occluded periodically during profile walking). To reduce ambiguities introduced by this type of occlusions as well as motion blurs, illumination changes and cloth deformations, we incorporate local motion priors into our framework to serve as additional constraints. We demonstrate the effectiveness of our approach by performing numerical evaluations on synthetic data and comparisons with Kinect results on real data. Our approach is shown to be capable of reconstructing a wider range of motions compared with Kinect (Fig. 3.1).

3.1 Background

Our work falls into the category of markerless human motion capture. So we will focus on the related work in this area in general. It would be incomplete, however, if we omit its counterpart, marker-based human motion capture, so we first briefly review the marker-based motion capture techniques available and the limitations.

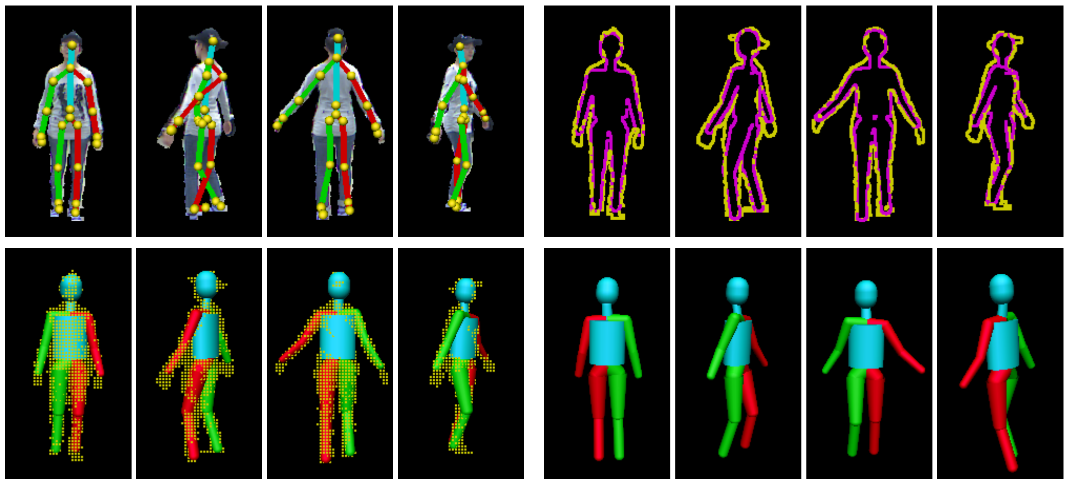


Fig. 3.1. Results for 360° walking motion: (top left) Kinect results superimposed on input color images; (bottom left) our results superimposed on input depth data; (top right) 3D depth edges from observed depth data in yellow and 3D depth edges from our results in pink; (bottom right) our results rendered from another viewpoint. We use different colors to distinguish left limbs from right limbs.

3.1.1 Marker-based Human Motion Capture

Marker-based motion capture techniques have been commercialized for quite a while and have influenced a lot of areas, including education, sports, and recently, computer animation for tvs and movies, video games, virtual realities etc. Vicon [1] is one of the most accurate optical commercial motion capture systems using this kind of technologies. However, it requires the performers wear skin-tight clothing and a set of retro-reflective markers, usually in the number of 40-50, which is not only intrusive, but also cumbersome. Besides, it needs a significant amount of setup time and often demands hours of post-processing by high-skilled experts. Furthermore, it is very expensive. Those limitations drive a lot of researchers to dedicate themselves to markerless motion capture.

3.1.2 Markerless Human Motion Capture

Markerless human motion capture have been studied for more than two decades and remains an active research area in computer vision and computer animation community. There is a vast literature on this topic and the reader is referred to [6–9] for a complete overview.

Studies can be divided into two main categories depending on whether human models are used or not, model-based [42–44], and model-free methods [45–49]. Model-based approaches typically define and use a humanoid kinematic model to find a sequence of pose configurations that best matches the observations by using an analysis-by-synthesis methodology. These approaches typically require an initial pose at the first frame to activate the tracking. Model-free methods, on the contrary, do not assume a priori skeleton model and estimate poses by probabilistically assembling detected and labeled body parts in 2D in a bottom-up approach [50–53], or by learning a mapping from 2D image features such as silhouettes to 3D poses directly [48, 54–56], or by looking up a database of exemplars together with their corresponding pose descriptors to find the most similar example [49, 57–60]

Approaches can also be classified as single hypothesis or multiple hypothesis. For example, single hypothesis approaches are mostly local-optimization methods [42,61,62], which often suffer from error accumulations. Multiple hypothesis methods use sampling-based approaches [44,63–67] to track non-linear motion and reduce the possibilities of getting stuck in local minimums. In contrary to sequential tracking, batch methods [54,68–70] optimize poses over a window of frames or an entire motion.

Methods also fall into two classes based on the number of cameras used, monocular and multi-view [43, 71]. Monocular work [56, 72–77] is very appealing since, in many applications, only a single camera or an image sequence from a single view is available. However, it is much more challenging compared with multi-view work due to the inherent depth ambiguities caused by 2D observations from one view. The techniques employed often add additional constraints such as kinematic constraints, or key frame constraints or movement constraints [42, 62, 78] etc.

Besides, there are a number of researchers dedicating themselves to markerless performance capture, which captures more detailed body deformations [71, 79–82]. For instance, the work by De Aguiar *et al.* [71] allow subjects to wear everyday apparel, such as a skirt in a dancing motion. It recovers poses by deforming mesh models of the subjects to match recorded images from 8 synchronized video cameras. There are also commercial systems available for markerless motion capture, including Organic Motion [83].

Our approach falls into the category of model-based single hypothesis sequential tracking.

With the recent availability of low-cost depth cameras, more research efforts have been put on markerless motion capture using depth data [38–41, 84, 85, 85–91]. Grest *et al.* [39] use Iterative Closest Point (ICP) technique to sequentially track an articulated mesh model from monocular depth data along with silhouette correspondences between the projected model and the image, mainly for upper body movements. ICP technique is also used in [40], which uses degenerated cylinders to model all body parts and combines 2D tracking results based on features of face and hands into the tracking framework. Besides tracking approaches, there are detection techniques, which usually do not assume a priori skeleton model and estimate poses by probabilistically assembling detected body parts in a bottom-up approach. Plagemann *et al.* [85] construct 3D meshes to detect geodesic extrema interest points and then apply learned patch classifiers to label them as head, hands or feet without distinguishing left from right. The core algorithm [41] used in Microsoft Kinect also takes a detection approach by formulating it as a per-pixel classification problem for each frame independently. It trains Randomized Decision Forests on a large synthetic depth image database rendered from motion capture data. Plagemann *et al.* [86] directly predict joint positions by regression based on Hough forests. Research efforts have also been put into combining detection and tracking techniques to benefit from both. Siddiqui and Medioni [87] combine bottom-up detections results and top-down likelihood in a data-driven MCMC framework for upper body pose estimation. Ganapathi and his colleagues [88] utilize body part detections obtained from [85] and further conduct local model-based search exploiting kinematic and temporal information. Ye *et al.* [89] utilize a data-driven pose detection technique to first identify a similar pose in the pre-captured motion database and then refine pose through non-rigid registration. Baak *et al.* [90] contribute in a similar manner by combining local optimization with global retrieval techniques.

Our work differs from any of the above work since we use a combination of color/depth and 3D edge information in our tracking and, most importantly, utilize effective local statistical priors to help reduce ambiguities, which makes the system more robust and reliable. To the best of our knowledge, none of the previous approaches have demonstrated the capability of reconstructing such a wide variety of complex movements with high accuracy.

3.1.3 Statistical Priors

There has been a long history of using statistical priors learned from prerecorded motion capture database to model, synthesize and track human motions due to their compact representations and generalizabilities. Approaches mainly fall into two categories, generative approaches [24, 72, 77, 92–101] and discriminative methods [45, 55, 56, 102, 103]. Howe *et al.* [72] build a mixture-of-Gaussians probability density model from training data and recover 3D motion from snippets of 2D tracking data by solving a Maximum A Posterior (MAP) problem. Chai *et al.* [24] learn a series of online Principle Component Analysis (PCA) models from pre-recorded motion capture database to reconstruct motion from low-dimensional signals from two views. The authors of [104] explore temporal motion models by performing PCA on a multi-activity database. Lou *et al.* [100] learn spatial-temporal patterns embedded in motion capture database to automatically clean up noisy motion capture data. Baak *et al.* [99] integrate motion priors dynamically retrieved from databases to stabilize sequential tracking. Li *et al.* [105] learn globally coordinated mixture of factor analyzers from motion capture data with each fact analyzer modeling a part of the manifold. While inspired by the work in [24], our second approach in this dissertation differs from [24] in two ways. First, temporal coherence is used implicitly in our work. Second, we combine online models with monocular depth data, color information and 3D edges rather than 3D marker positions obtained from synchronized cameras.

3.2 System Overview

The key idea of this approach is to identify important cues from the color/depth observations and combine them with prior knowledge to significantly reduce reconstruction ambiguities due to self-occlusions, lighting changes, clothes deformations etc.

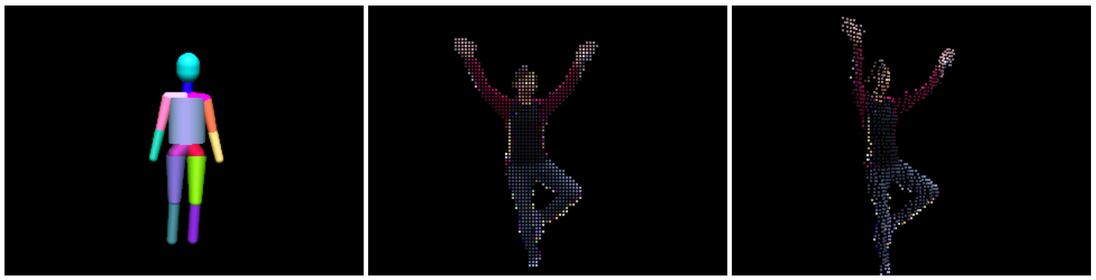

(a) (b) (c)

Fig. 3.2. Skeleton model and colored depth points: (a) the simplified skeleton model, with a different color for each bone; (b) 3D depth points with colors from images, camera view (points displayed are downsampled by 9); (c) 3D depth points from a second view with the same downsampling rate.

3.2.1 Problem Formulation

First, our work employs a simplified articulated skeleton model of 15 bones as shown in Fig. 3.2 (a), which has 34 DOFs totally including root rotation and translation. Bones are represented by elliptic cylinders and spheres, e.g. head is modeled as two spheres at ends and one cylinder in the middle, and torso is modeled as a single cylinder. Given this model, pose at each frame is parameterized in 34-dimensional space and tracking essentially becomes a parameter estimation problem for each frame.

Assume known pose configuration q_t at time t, then at time $t+1$, we formulate the parameter estimation problem in a Maximum a Posterior (MAP) framework. The goal is to infer the most likely pose q_{t+1}, given the new observation c_{t+1} at time $t+1$ along with the previous pose q_t and c_t. Based on Bayes' theorem, we have

$$\begin{aligned} \arg\max_q Pr(q|c) &= \arg\max_q \tfrac{Pr(c|q)Pr(q)}{Pr(c)} \\ &\propto \arg\max_q Pr(c|q)Pr(q) \end{aligned} \tag{3.1}$$

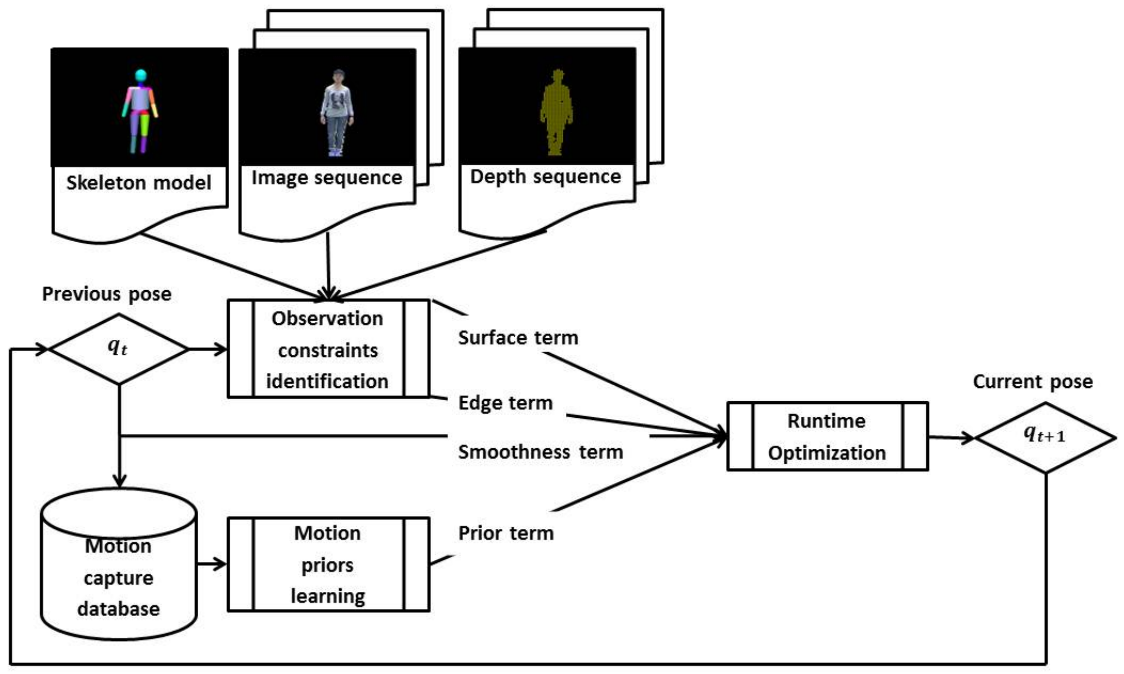


Fig. 3.3. System overview.

where $Pr(c)$ is a normalizing constant, $Pr(q)$ represents the prior probability and $Pr(c|q)$ is a likelihood probability.

Take the negative log of $Pr(q|c)$, the maximization problem described in Equation 3.1 becomes a minimization problem as follows:

$$\begin{aligned} \hat{q} &= \arg\min_q -lnPr(q|c) \\ &= \arg\min_q \underbrace{-lnPr(c|q)}_{E_{likelihood}} \underbrace{-lnPr(q)}_{E_{prior}} \end{aligned} \tag{3.2}$$

where the first term measures how well a pose matches the observations and the second term measures the a priori likelihood of the pose using the knowledge embedded in the pre-recorded motion data.

3.2.2 Major Components

The system consists of four major components (see Fig. 3.3):

Preprocessing. Before running our algorithm, we need to capture and process data using a Kinect device with the assistance of Kinect SDK [106]. Also, the pose for the first frame, q_0, needs to be estimated to initialize the tracking process. Please note that we assume skeleton models of the subjects are known in advance.

Observation Constraints Identification. Our system employs two main criteria to evaluate how well a pose matches the observations, 3D surface-based constraints and 3D edge-based constraints. The former measures the match between the rendered surface points and observed surface points, while the latter measures the consistency between the rendered edge points and observed edge points. This step takes *current pose* q_t, *Image sequence*, *Depth sequence* and *Skeleton model* (Fig. 3.3) as inputs and produces two terms for our overall objective function, *Surface term* and *Edge term* as shown in Fig. 3.3.

Motion Priors Learning. Our system automatically learns a local statistical model from selected motion capture data at each time $t + 1$ based on the previous pose q_t. The learned model adds additional knowledge or constraints to help reduce the ambiguities arising especially due to occlusions. This correspondes to *Prior term* in Fig. 3.3.

Runtime Optimization. At each time $t + 1$, our system solves a linear system iteratively to automatically find a pose q_{t+1} that best matches the constraints from the observations and is also consistent with the statistical patterns learned from the pre-captured data. We also use *Smoothness term* (Fig. 3.3) to help avoid sudden pose changes. This optimization is done by minimizing the overall objective function consisting of all these four terms. Reconstructed q_{t+1} will be used as a previous pose for the estimation of a new current pose q_{t+2}.

The preprocessing step is done off-line, while the rest are performed online sequentially. We describe these components in detail in the next four sections.

3.3 Preprocessing

In this section, we describe two steps prior to running our algorithm. The first step involves recording data from Kinect online, and processing recorded data off-line, which is relatively computational expensive and will decrease the frame rate if done in the online step. The second step prepares the initial pose for the tracking.

3.3.1 Data Acquisition

The recent release of Kinect SDK [106] has made both the data capturing and comparisons a lot easier. A single Kinect device, consisting of a depth camera and a video camera, is used as the source for all the real data used in our experiments. We simultaneously record video data with a resolution of 640×480 pixels, depth data of 320×240 pixels, and pose estimations while a subject wearing normal clothes performs in front of the Kinect cameras. Afterwards, video data with a resolution of 320×240 pixels is obtained by registering the coordinate system of the video camera to that of the depth camera. Meanwhile, 3D depth points are obtained for each frame of the record. All those steps are facilitated by the SDK. Furthermore, intrinsic parameters of the depth camera are available in the SDK, which eliminates the need for camera calibration. All the real data sequences used in our work are obtained as described above.

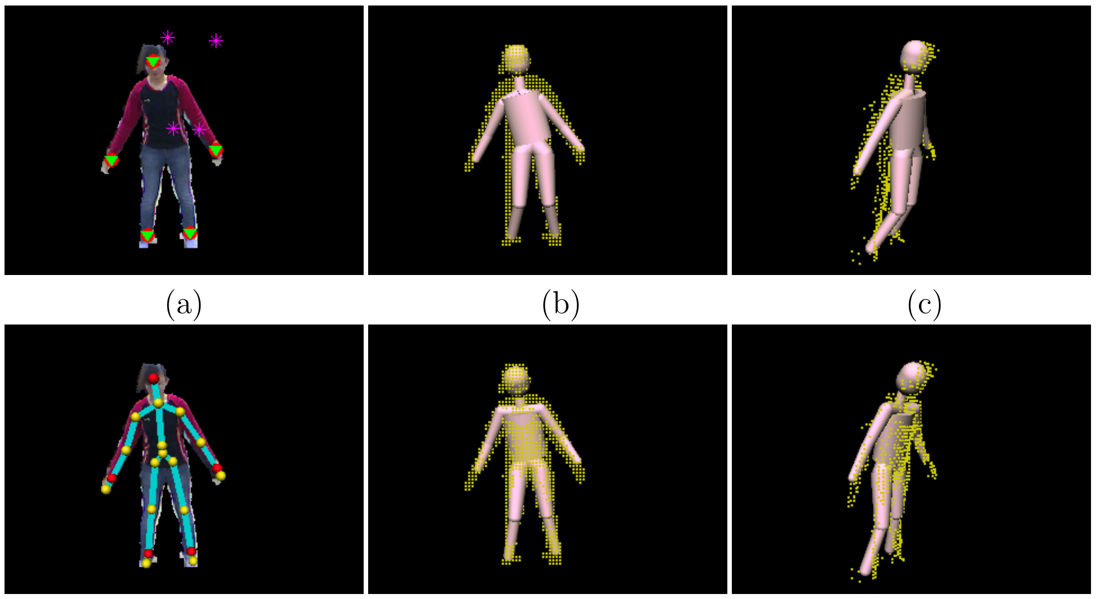
For synthetic data, we use ground truth motion data originally captured by Vicon system [1] to generate video and depth data, each with the resolution of 320×240 pixels, and 3D depth points. More specifically, color images are synthesized through rendering; 3D depth points are obtained by un-projecting from screen space to world space for all visible points; depth images are obtained by taking z values of 3D depth points for foreground and a user-specified large number (e.g. 100) for background. Camera parameters are also known in this case.

3.3.2 Tracking Initialization

Initialization of the first frame is a necessary step for our model-based tracking. In our work, for synthetic data, the first frame of the ground truth motion is used. For real data, there are two steps specifically to achieve this purpose with no user interaction needed.

First, we use 5 joint locations (head, left wrist, right wrist, left ankle, and right ankle) as targets to obtain an initial pose. This can be achieved typically by having user click 5 2d projections on the images. In our work, however, we simply take advantage of the 3D joint positions extracted from Kinect result. An initial pose is obtained by minimizing the differences between the extracted constraints and the positions of those 5 joints, which can be calculated via forward kinematics in terms of the joint angle pose. It is a typical inverse kinematics problem, and a normal standing pose is sufficient as an initial guess to this problem. Fig. 3.4 (a) to (d) show this step.

The result from the above step usually is not good enough for successful tracking since it is solely based on a few 3D targets. To improve it, the depth data and 3D depth edge of the first frame needs to be exploited, which is very similar to



(a) (b) (c) (d) (e) (f)

Fig. 3.4. First frame initialization: (a) joint projections superimposed on the corresponding color image; magenta asterisks represent projections of 5 joints of an initial given pose (head projection is out of image boundary); red circles represent targets extracted from Kinect result; green downward-pointing triangles represent the projections after optimization; (d) Kinect result superimposed on the corresponding color image with red spheres representing the 5 target joints used for optimization; (b) and (c) show optimized poses, corresponding to green projections in (a), from 2 different views along with depth points; (e) and (f) show optimized poses after applying our tracking procedure to the pose shown in (b) for one time step.

the tracking step detailed in the next section except that no color information is available and used in the process of building correspondences. Fig. 3.4 (b)/(c) and (e)/(f) present the results before and after this step.

3.4 Observation Constraints Identification

This section presents the details of observation constraints and how we obtain them.

At each time t, the observations consist of a color image C, a depth image D, and a set of 3D depth points denoted by P. C is 320×240 and each pixel contains an intensity value. D is also of size 320×240, with every pixel containing the z value of the corresponding 3D depth point.

The observations together actually is equivalent to a point cloud, where each point has both 3D position and color information (see Fig. 3.2 (b) and (c)). The constraints from the observations can be categorized into two groups: 3D surface-based constraints and 3D edge-based constraints. The 3D surface-based constraints measures the match between the rendered surface points and observed surface points, while the 3D edge-based constraints measures the consistency between the rendered edge points and observed edge points. These two groups of constraints are completely independent of each other due to the fact that they are obtained independently even for the same point. For instance, for an edge point on the rendered surface, its corresponding point in the group of 3D surface-based constraints usually is not an edge point, while its corresponding point in the group of 3D edge-based constraints must be an edge point. Therefore, the likelihood term in Equation 3.2 can be decomposed into two terms as follows:

$$\begin{aligned} E_{likelihood} &= -\ln Pr(c|q) \\ &= -\ln Pr(c_{surf}, c_{edge}|q) \\ &= \underbrace{-\ln Pr(c_{surf}|q)}_{E_{surf}} \underbrace{-\ln Pr(c_{edge}|q)}_{E_{edge}} \end{aligned} \tag{3.3}$$

3.4.1 3D Surface-based Constraints

Let q_t denote the pose configuration at frame t, $O_t(C, D, P)$ represents all the observations at frame t, with C as the color image, D as the depth image and P as the 3D depth points; similarly, $R_t(D, P, A)$ represents all the rendered data at frame t given the pose q_t and camera parameters. The goal is to solve q_{t+1}, which best explains $O_{t+1}(C, D, P)$. $R_t(P)$ is obtained by un-projecting from screen space to world space for all visible points at frame t. Besides, each rendered point has additional information represented in $R_t(A) = R_t(bid, loc)$, with $R_t(bid)$ denoting the bone IDs to which $R_t(P)$ belong and $R_t(loc)$ representing the local coordinates with respect to $R_t(bid)$. $R_t(bid)$ can be easily obtained by rendering different bones in different colors (see Fig. 3.2 (a)) while $R_t(loc)$ can be calculated based on q_t, $R_t(bid)$ and $R_t(P)$. $R_t(D)$ consists of the z values of all $R_t(P)$ for foreground pixels and a user-specified large number (we use 100) for background pixels.

At time $t+1$, c_{surf} is obtained by searching for the closest observed point $O_{t+1}(p^i)$ in $O_{t+1}(P)$ for each rendered point $R_{t+1}(p^i)$ in $R_{t+1}(P)$ based on q_{t+1}. More specifically, the search is conducted in a small neighborhood defined by a window of size 15×15 centered at the projected 2D pixel location of the rendered point, and weights are experimentally set to 1.0 and 0.0001 for depth difference and color difference respectively. Window size is also set experimentally. Generally speaking, larger window sizes produce better matching accuracy while demanding more computational power.

However, there is not much improvement in accuracy once the window size is larger than a threshold value, which is different for motions with different speeds. Through experiments, we find 15×15 is large enough to achieve the balance between efficiency and accuracy for a lot of motions and we use this value for all the testing examples.

We assume Gaussian noise with a standard deviation σ_{surf} for c_{surf}, then the likelihood term for c_{surf} can be defined as follows:

$$\begin{aligned} E_{surf} &= -\ln Pr(c_{surf}|q) \\ &= -ln \prod_i \frac{1}{\sqrt{2\pi}} \exp \frac{-\|O_{t+1}(p^i) - f(R_{t+1}(a^i), q_{t+1})\|^2}{\sigma_{surf}^2} \\ &= -\sum_i \ln \frac{1}{\sqrt{2\pi}} + \sum_i \frac{\|O_{t+1}(p^i) - f(R_{t+1}(a^i), q_{t+1})\|^2}{\sigma_{surf}^2} \end{aligned} \tag{3.4}$$

where $R_{t+1}(a^i) = R_{t+1}(bid^i, loc^i)$ is the bone ID and local coordinates corresponding to $R_{t+1}(p^i)$ and f is the forward kinematics function which computes the global point position given the pose configuration q_{t+1}, bone ID bid^i, and local coordinates loc^i of the point. In this minimization problem, q_{t+1} is initialized to be equal to q_t, and a Δq is solved in a least square method iteratively with the update $q_{t+1} = q_{t+1} + \Delta q$ at the end of each iteration. Here comes the derivation for Δq:

$$\begin{aligned} E_{surf} &= -\sum_i \ln \frac{1}{\sqrt{2\pi}} + \sum_i \frac{\|O_{t+1}(p^i) - f(R_{t+1}(a^i), q_{t+1})\|^2}{\sigma_{surf}^2} \\ &= -\sum_i \ln \frac{1}{\sqrt{2\pi}} + \sum_i \frac{\|O_{t+1}(p^i) - f(R_{t+1}(a^i), q_{t+1} + \Delta q)\|^2}{\sigma_{surf}^2} \\ &= -\sum_i \ln \frac{1}{\sqrt{2\pi}} + \sum_i \frac{\|O_{t+1}(p^i) - f(R_{t+1}(a^i), q_{t+1}) - \frac{\partial f_R^i}{\partial q_{t+1}} \Delta q)\|^2}{\sigma_{surf}^2} (Taylor\ expansion) \end{aligned} \tag{3.5}$$

Let the derivative of the above objective function with respect to Δq equal 0, we have

$$\sum_i \frac{2 \frac{\partial f_R^i}{\partial q_{t+1}}^T (\frac{\partial f_R^i}{\partial q_{t+1}} \Delta q + f(R_{t+1}(a^i), q_{t+1}) - O_{t+1}(p^i))}{\sigma_{surf}^2} = 0 \tag{3.6}$$

where $f_R^i = f(R_{t+1}(a^i), q_{t+1})$.

Reorganizing the equation leads to:

$$\underbrace{\sum_i (J_R^i)^T J_R^i \Delta q}_{A_{surf}} = \underbrace{\sum_i (J_R^i)^T b_R^i}_{b_{surf}} \tag{3.7}$$

where $J_R^i = {}^{\partial f_R^i}/_{\partial q_{t+1}}$ and $b_R^i = O_{t+1}(p^i) - f(R_{t+1}(a^i), q_{t+1})$. This procedure differs from conventional Iterative Closest Point (ICP) technique by integrating color information from the observations seamlessly into the measurements. This is allowed by the fact that each observed 3D point has a color associated with it, and so does any rendered point overlapping with the observed color image of the same frame. Color information helps increase accuracy of tracking as demonstrated in the results section.

3.4.2 3D Edge-based Constraints

The above optimization is prone to local minimum especially when pose changes fast, therefore, it is not good enough to measure the match between the estimated poses and the observations. Edges are very important features. However, if edges points are treated the same as any other point on the rendered surface, the closest points found are often not edge points of the observed surface when the truth is quite the opposite. We introduced 3D edge constraints into the tracking framework (Fig. 3.5), where for each point on the observed 3D edge, the closest point on the rendered 3D edge needs to be searched for in a small neighborhood defined by a window of size 21×21 (experimentally set) centered at its 2D pixel location. All found closest points will be used as new constraints to solve for a likely pose change similar to the procedure in the above section. To obtain a 3D edge, we first apply

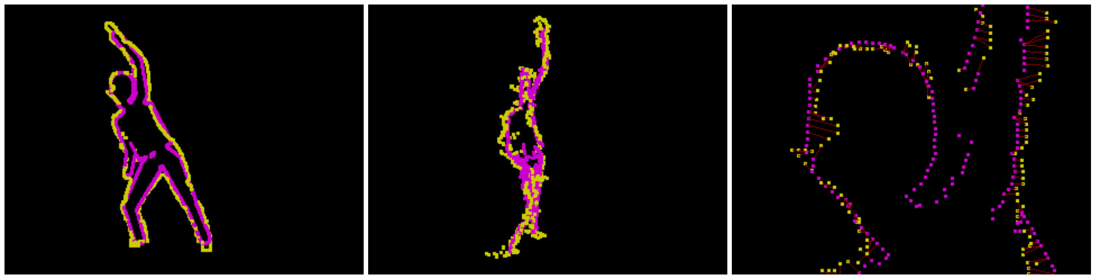


Fig. 3.5. 3D edges and closest correspondences. Rendered 3D edge is shown in pink while observed 3D edge is shown in yellow and red lines connect 3D points on the observed edge with their closest correspondences. The first two images are from camera view and a second view. The third image shows an enlarged view.

Canny edge detector to the depth image to get a 2D edge. Since each foreground pixel in the depth image corresponds to a 3D depth point, the 3D edge can be easily obtained by associating all 3D depth points with the pixels on the 2D edge. There are cases when 2D edge pixels of the detection results are actually background pixels, 3D depth points associated with their immediate foreground neighbor pixels are used instead if there is any. This component is similar to the 3D surface-based constraints, however it greatly increases the influence of key features.

Notations for 3D edges can be simply adapted from notations for 3D surfaces by replacing R and O with RE and OE respectively. To be precise, we use $RE(P, A)$ and $OE(P)$ to represent 3D edges for rendered depth data and observed depth data respectively. Again, $RE(A) = RE(bid, loc)$ is the additional information corresponding the $RE(P)$. Similarly, we assume Gaussian noise with a standard deviation σ_{edge} for c_{edge}, the likelihood term for c_{edge} is defined as follows and can be solved accordingly.

$$\begin{aligned}
E_{edge} &= -\ln Pr(c_{edge}|q) \\
&= -ln \textstyle\prod_i \frac{1}{\sqrt{2\pi}} \exp \frac{-\|OE_{t+1}(p^i)-f(RE_{t+1}(a^i),q_{t+1})\|^2}{\sigma^2_{edge}} \\
&= -\textstyle\sum_i \ln \frac{1}{\sqrt{2\pi}} + \sum_i \frac{\|OE_{t+1}(p^i)-f(RE_{t+1}(a^i),q_{t+1})\|^2}{\sigma^2_{edge}} \\
&= -\textstyle\sum_i \ln \frac{1}{\sqrt{2\pi}} + \sum_i \frac{\|OE_{t+1}(p^i)-f(RE_{t+1}(a^i),q_{t+1}+\Delta q)\|^2}{\sigma^2_{surf}} \\
&= -\textstyle\sum_i \ln \frac{1}{\sqrt{2\pi}} + \sum_i \frac{\|OE_{t+1}(p^i)-f(RE_{t+1}(a^i),q_{t+1})-\frac{\partial f^i_{RE}}{\partial q_{t+1}}\Delta q)\|^2}{\sigma^2_{surf}} (Taylor\ expansion)
\end{aligned} \tag{3.8}$$

$$\underbrace{\sum_i (J^i_{RE})^T J^i_{RE}}_{A_{edge}} \Delta q = \underbrace{\sum_i (J^i_{RE})^T b^i_{RE}}_{b_{edge}} \tag{3.9}$$

where $J^i_{RE} = {\partial f^i_{RE}}/{\partial q_t}$ and $b^i_{RE} = OE_{t+1}(p^i) - f(RE_{t+1}(a^i), q_t)$.

3.5 Motion Priors Learning

The pose update inferred in the previous section is purely based on the likelihood of matching the observations. However, human pose is high dimensional, still having 34 DOFs even though expressed with a simplified skeleton model. Only observed constraints from monocular sequences are far from sufficient to determine a meaningful pose update especially in the case of occlusions. To address this problem, we use a series of local linear models learned from pre-captured motion capture database to constrain the current search space. More specifically, we learn local Principal Component Analysis (PCA) models indirectly based on K nearest neighbors of the previous pose and evaluate how well an estimated pose can be expressed by learned PCA models. PCA is a dimensionality reduction technique, which has been widely used and proven to be effective in a variety of areas including data compression, image processing, visualization, pattern recognition and time series prediction. However, PCA model is a linear model and human motion exhibits highly nonlinear patterns, so a single PCA is not sufficient for an entire motion. Therefore, we use a series of Local PCA (LPCA) models, with each of them modeling an instant part of the motion. While the reader is referred to [107, 108] for detailed mathematical derivations, we briefly describe in this section how to build LPCA models and combine them with the previous formulation. Note that original motion capture data has 30 bones and 62 DOFs including root position and orientation. We preprocess them to suit the simplified skeleton model before model learning.

Given pose q_t at frame t, K closest poses are selected from the motion capture database efficiently by the method discussed in [24]. The following w poses of each of the K poses in their original motion sequences will be stacked together into a data matrix D of size $(m-6)$ by N, where m is the dimensionality of q_t and N is total number of frames used, $N <= K \cdot w$ considering frames close to the end of the motions do not have w subsequent frames, and we exclude root position and orientation in model learning. We experimentally set $w = 3, K = 50$. Apply Singular Value Decomposition(SVD) to the covariance data matrix $C = {}^{\tilde{D} \cdot \tilde{D}^T}/_{(N-1)}$, where $\tilde{D}$ is obtained by subtracting mean pose $\bar{q}$ from each column of D, now we have:

$$U \cdot S \cdot V^T = SVD(C) \tag{3.10}$$

Each row vector of U is an eigenvector and S is a diagonal matrix of eigenvalues. We use the following criteria to determine how many basis vectors to keep and *energy* is experimentally set to be 0.95.

$$e = \min\{d, \frac{\sum_{i=1}^{d} S(i)}{\sum_{i=1}^{m-6} S(i)} \geq energy\} \tag{3.11}$$

To evaluate how well a pose q_{t+1}, in high dimensional space, matches the learned model, we first transform the pose q_{t+1} to low space and then transform it back to high space as pose $\tilde{q}_{t+1}$. We measure the differences between q_{t+1} and $\tilde{q}_{t+1}$, which is known as reconstruction error. The smaller the reconstruction error is, the better pose q_{t+1} is consistent with the natural patterns embedded in the motion database. Similar to the formulation in the previous section, we have

$$\begin{aligned} E_{prior} &= -\ln Pr(q) \\ &= -\ln \frac{1}{\sqrt{2\pi}} + \frac{((U_e U_e^T - I)\|q_t + \Delta q - \bar{q})\|^2}{\sigma_q^2} \end{aligned} \tag{3.12}$$

where U_e is of size m by e, with first 6 rows all 0's and the rest from the first e column vectors of U. I is an identity matrix of size m by m except that the first 6 rows and the first 6 columns contain all 0's. σ_q is the assumed standard deviation for q. Again, let the derivative with regard to Δq equal 0, we get:

$$\underbrace{J_q^T J_q}_{A_{prior}} \Delta q = \underbrace{J_q^T b_q}_{b_{prior}}, \tag{3.13}$$

where $J_q = U_e U_e^T - I$, $b_q = J_q(\bar{q} - q_{t+1})$.

3.6 Runtime Optimization

Given initialized pose at the first frame, at each time t, the MAP problem actually solves for a pose change so that the new pose matches the corresponding observations as well as the statistical properties learned from captured data.

The overall objective function derived from the MAP is a weighted combination of two constraints terms and an online motion prior term. We also add a smoothness term to help avoid sudden pose changes in the reconstructed motion.

$$\arg\min_q E_{surf} + \lambda_1 E_{edge} + \lambda_2 E_{prior} + \lambda_3 E_{smoothness} \tag{3.14}$$

In our experiments, λ_1 is set to 10; λ_2 is set to between 0.001 and 0.01 (we used 0.001 for all the experiments except for the $360\,^\circ$ walking example when we used 0.01.) λ_3 is set to 0.00001.

It is easy to derive that the overall Jacobian term is a weighted combination of the individual Jacobian terms derived in the above sections (Note we ignore the differences among the standard deviations).

$$\underbrace{A_{surf} + \lambda_1 A_{edge} + \lambda_2 A_{prior} + \lambda_3 A_{smooth}}_{A_{total}} \Delta q = \underbrace{b_{surf} + \lambda_1 b_{edge} + \lambda_2 b_{prior} + \lambda_3 b_{smooth}}_{b_{total}} \tag{3.15}$$

where $A_{smooth} = I$ and $b_{smooth} = q_t - q_{t+1}$.

$$\Delta q = (A_{total})^{-1} b_{total} \tag{3.16}$$

q_{t+1} is initialized with q_t and solved iteratively. The algorithm consists of the following steps:

1. Obtain $R(D,P)$ through rendering based on the skeleton model, pose configuration at frame t, q_t, and known camera parameters. Initialize pose at frame $t+1$ with the previous pose, $q_{t+1} = q_t$.
2. Calculate $R(A)$ and build correspondences for visible points. Specially, for each point visible in the rendered image $R(p^i) \in R(P)$, determine which bone $R(bid^i)$ it belongs to, calculate the local coordinates $R(loc^i)$ with respect to $R(bid^i)$, and record the corresponding pixel's image intensity. This step is a must for the evaluation of partial derivatives $\partial f_R^i/\partial q$ and $\partial f_{RE}^i/\partial q$.

3. Calculate $RE(P, A)$ and $OE(P)$. $RE(P)$ and $OE(P)$ are acquired by first applying Canny edge detection to $R(D)$ and $O(D)$ respectively, followed by a mapping from 2D screen space to 3D world space. $RE(A)$ is easily obtained in a similar way to $R(A)$.
4. Build online local database by searching for the K-Nearest Neighbors (K-NN) based on pose q_t, and using the w subsequent frames of each of the K nearest neighbors. Learn a LPCA model as detailed in Section 3.5.
5. Evaluate A_{total} and b_{total} from Equation 3.15.
6. Compute Δq by Equation 3.16 and then update $q_{t+1} = q_{t+1} + \Delta q$. Go to step 1 if Δq is larger than a pre-defined threshold.

This algorithm typically converges in 10 iterations. For step 2, we ease the process by rendering each bone in a unique color. So, for each visible point's projection $s = (u, v)$ on the rendered color image, the bone ID can be easily determined by indexing a pre-stored color map. Besides, the local coordinates can be obtained by un-projecting from screen space (u, v, w) to world space and then transforming to bone's local space. We normally down sample the screen space by 4 in each dimension when computing $R_t(P)$, which means the number of visible points used for 3D surface-based constraints is roughly $1/4 \times 1/4$ of the total.

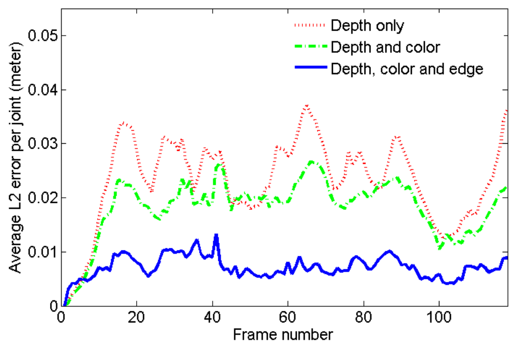
3.7 Results

In this section, we present the results. We conduct experiments on both synthetic data, rendered from ground-truth motions, and real data, recorded from a Kinect Device. For real data, we conducted 3 capture sessions and processed the data using the method as described in Data Acquisition section, with the latest available SDK releases back then, Kinect SDK beta 1 and beta 2 respectively. Data for motion 1, 2, 4 were captured using Kinect SDK beta 1 and data for motion 3, 5, 6, 7 were captured using Kinect SDK beta 2. Up to the writing of this dissertation, there has been another release, v1. We didn't recapture and reprocess the data due to time constraints, and most importantly, there are no visual differences between Kinect results of different versions on the motions we are testing on.

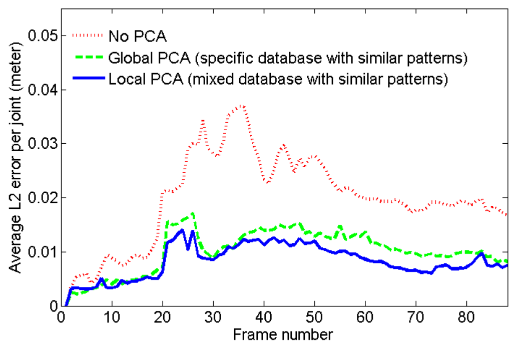
3.7.1 Experiments on Synthetic Data

Experiments are conducted on synthetic data to assess and justify different components of our approach. Two ground-truth motions, a dancing motion of 118 frames and a bending motion of 88 frames, are used to generate all the video images, depth images and 3D depth points. The dancing motion is rendered from a viewpoint without obvious occlusions while the bending motion is rendered with the left arm completely occluded for a duration of 7 frames. The first frame from the ground truth motion is used to initialize the tracking for each experiment. Three different experiments are performed progressively on the dancing motion to demonstrate the importance of color and edge while three other experiments on the bending motion are conducted to assess the influence of different motion priors. For dancing motion, the first experiment only considers depth information and color information is neglected in the step of looking for closest points. The second experiment distinguishes itself from the first one by taking color information into consideration when searching for closest points. The third experiment advances further by extending ICP to 3D edges. For bending motion, the first experiment utilizes no prior knowledge, which differentiates itself from the other two. The other two experiments use a specific database with only bending motions, a total of 4355 frames, a mixed database with 5 different motions including bending motions, a total of 13115 frames, respectively.

To quantitatively evaluate all these components, tracking results from all experiments are compared against ground truth motion. The error metric measuring the discrepancy between an estimated pose and a ground truth pose is used for every frame. Specifically, they are computed as follows. At each time step, for each joint (totally 16 including root), the L2 difference between its estimated position and ground truth position is calculated. The average difference per joint is therefore obtained. This is repeated for all frames of all estimated motions. Fig. 3.6 (a) shows the experimental results on the dancing motion and it is clearly seen the effectiveness of incorporating color and especially 3D edge information into the tracking framework. Fig. 3.6 (b) illustrates the importance of an appropriate database on the tracking results as well as the necessity of a local motion prior in order to use a general database.



(a)



(b)

Fig. 3.6. Comparisons on 2 sets of experiments: (a) comparisons on results obtained using different components on synthetic dancing motion; (b) comparisons on results reconstructed using different motion priors on synthetic bending motion.

Table 3.1
Details of testing examples.

No.	Motion Name	Database	Compare with Kinect
1	dancing_1	No	Comparable
2	dancing_2	No	Comparable
3	poker face dancing	No	Comparable
4	sitting	No	Better
5	yoga triangle	No	Better
6	yoga tree	database_1	Better
7	360° walking	database_1	Better
8	front kicking	database_2	Better
9	forward bending	database_2	Better
10	diagonal bending	database_2	Better
11	backward turning	database_2	Better

Table 3.2
Details of motion capture databases.

No.	Database Name	No. of different motions	No. of frames
1	database_1	2	6146
2	database_2	5	13115

3.7.2 Experiments on Real Data

We also perform experiments on real data collected from a single Kinect device. Kinect works pretty well when there are no occlusion or minor occlusions, however, in the presence of significant occlusions, it often fails. To demonstrate the effectiveness and advantages of our approach, we show results on a variety of motions, with a focus on the motions involving varying levels of occlusions where Kinect shows failures. Our results are best viewed in the accompanying video although we show sample frames of a few selected results here.

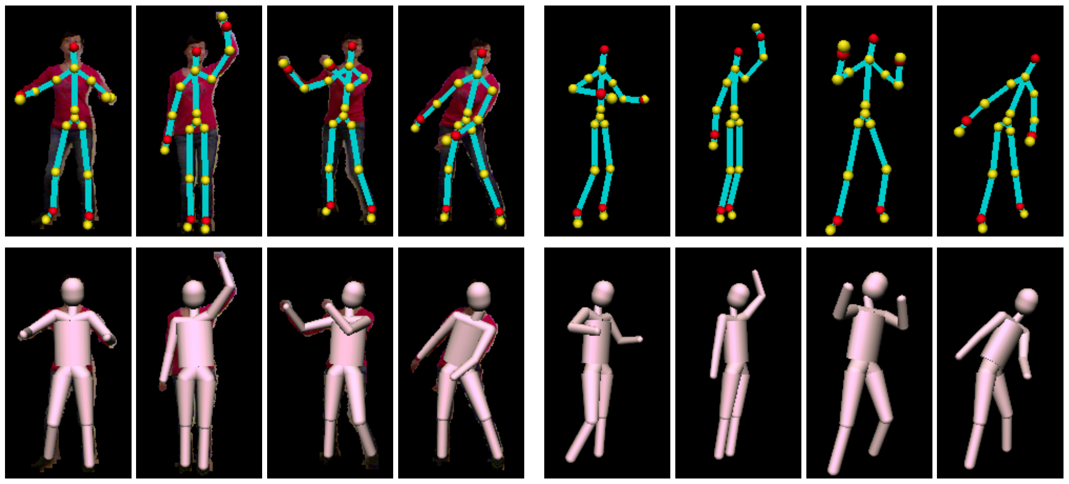


Fig. 3.7. Results for poker face motion: (top left) Kinect results superimposed on input color images; (bottom left) our results superimposed on input color images; (top right) Kinect results from another view; (bottom right) our results from another view.

Comparisons with Kinect

The experiments fall into 4 categories. The first category contains 3 different motions, on which our algorithm and Kinect work equivalently well. One of the examples, named poker face dancing (Table 3.1), is from the choreography of the poker face dancing in Dance Central game. Fig.3.7 shows the selected key frames

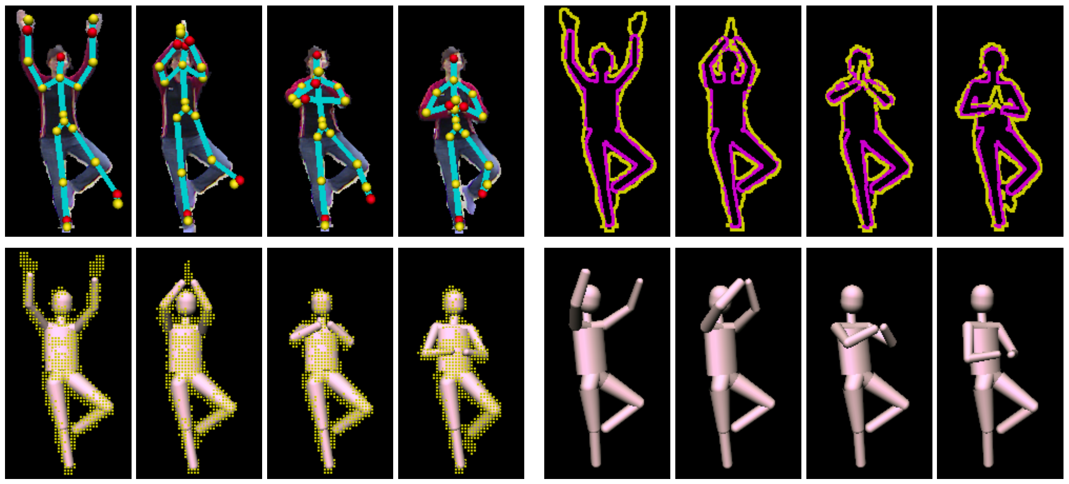


Fig. 3.8. Results for yoga tree motion: (top left) Kinect results superimposed on input color images; (bottom left) our results superimposed on input depth data; (top right) observed 3D depth edges in yellow and rendered 3D depth edges from our results; (bottom right) our results rendered from another viewpoint.

of Kinect results and our results. The other 3 categories consist of motions which demonstrate the superiority of our approach. Specifically, the second category has 2 motions, which has little occlusions and our algorithm exhibits advantages over Kinect without the support of prior knowledge. The third category has 2 motions, the yoga tree motion and 360° walking motion, with obvious occlusions and local motion priors learned from a small mixed database (Table 3.2) is used to reduce the ambiguities caused by the occlusions. There are 4 motions in the fourth category and they all share the same mixed database consisting of a total of 13115 frames of 5 different motions. Table 3.1 and Table 3.2 summarize the testing motions and databases used in all those experiments on real data. *database_1* contains yoga tree motion and 360° walking motion, both of which are different from the corresponding testing motions. *database_2* consists of diagonal bending motion, forward bending

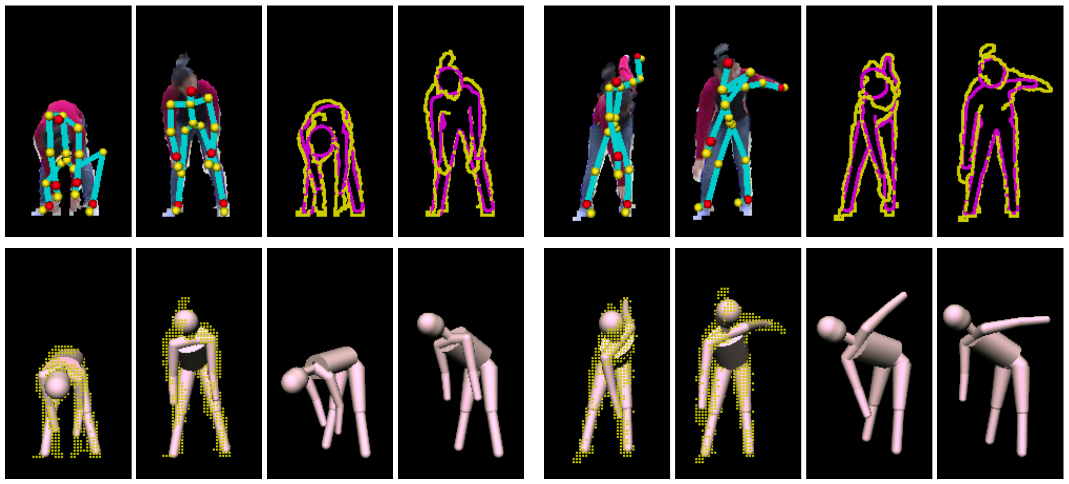


Fig. 3.9. Results for forward bending motion and diagonal bending motion: (top left) Kinect results and depth edges for forward bending motion; (top right) Kinect results and depth edges for diagonal bending motion; (bottom left) our results for forward bending motion (camera view and a second view); (bottom right) our results for diagonal bending motion (camera view and a second view).

motion, front kicking motion, 180° turning motion and a dancing motion similar to testing motion dancing_2. Fig. 3.8, Fig. 3.9, and Fig. 3.10 display the selected key frames of reconstructed motions as well as Kinect results. Our results are best viewed in the accompanying video although we show sample frames of a few results here.

Evaluations on 3D Edge and LPCA

We also evaluate the importance of 3D edge by running our algorithm with and without the component, respectively, and then compare the results visually by showing them side by side. Similar experiments are conducted to show the importance of LPCA. Fig. 3.11 and Fig. 3.12 show the selected key frames from those comparisons.

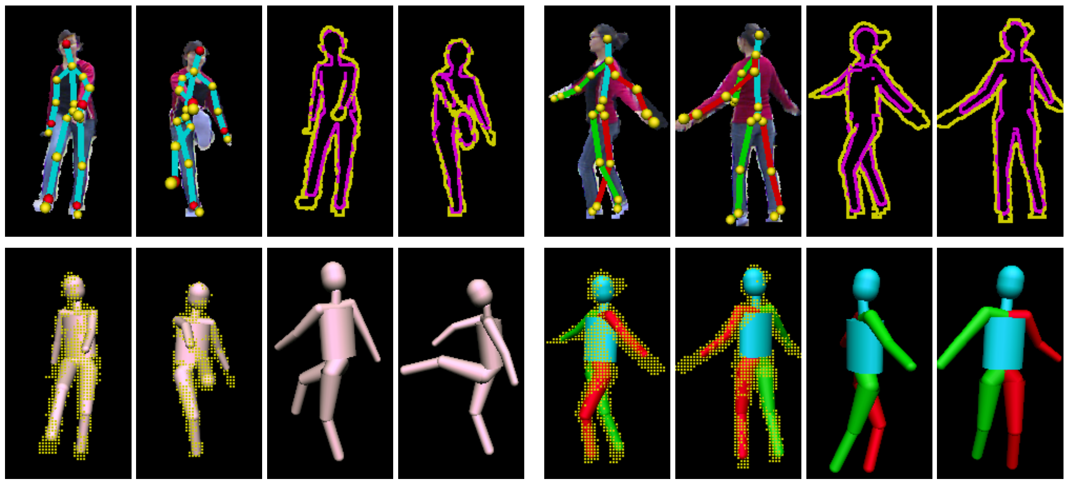


Fig. 3.10. Results for front kicking motion and backward turning motion: (top left) Kinect results and depth edges for front kicking motion; (top right) Kinect results and depth edges for backward turning motion; (bottom left) our results for front kicking motion (camera view and a second view); (bottom right) our results for backward turning motion (camera view and a second view).

3.8 Discussion

In this work, we proposed an approach, which effectively reconstructs 3D full-body human motion from a combination of color and depth streams obtained from a single Kinect device on subjects performing wearing normal clothes. We introduced 3D edge obtained from depth stream into our framework to increase the accuracy of tracking, proven to be effective especially when pose change is large or smaller body parts occluding larger ones. Moreover, motion priors learned from pre-recorded motion capture database are employed to handle obvious occlusions where body parts are mostly or completely occluded. It is worth noting that we do not consider situations when the subject is fully occluded by other people or objects in the environment. Therefore, from time t to time t+1, only some parts disappeared or re-appeared while

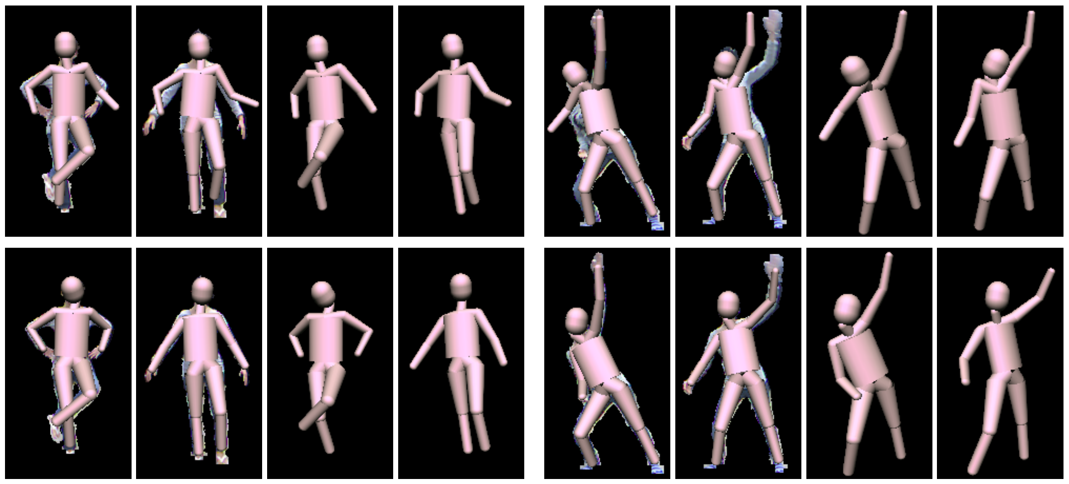


Fig. 3.11. 3D edge evaluation: (top left) results without edge for sitting motion (camera view and a second view), with results from camera view superimposed on input color images; (bottom left) results with edge for sitting motion (camera view and a second view), with results from camera view superimposed on input color images; (top right) results without edge for yoga triangle motion (camera view and a second view), with results from camera view superimposed on input color images; (bottom right) results with edge for yoga triangle motion (camera view and a second view), with results from camera view superimposed on input color images.

the rest's visibility status remains the same. LPCA priors tend to have a stronger influence over the pose solving when ambiguities rise due to body parts disappearing or re-appearing. This framework enhances the accuracy and robustness of tracking by taking advantage of all available information, which is demonstrated by experiments on both synthetic and real data.

Our approach is subject to common limitations shared by any other model-based tracking work, including the requirement of an initial pose and the need of a fitted skeleton model. Our work eliminates the need of heavy user interaction or manual process by taking advantage of Kinect's results as initial constraints. A normal

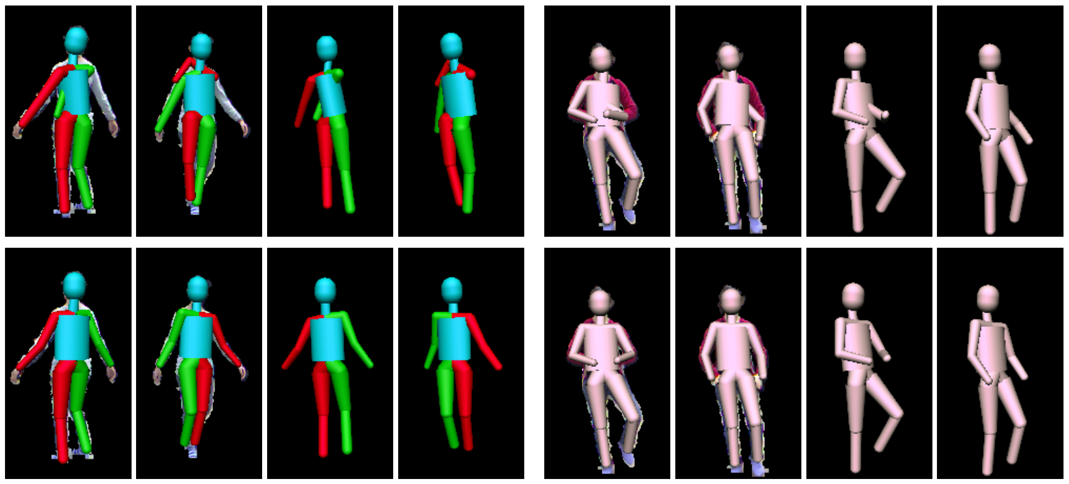


Fig. 3.12. LPCA evaluation: (top left) results without LPCA for 360 ° walking motion (camera view and a second view), with results from camera view superimposed on input color images; (bottom left) results with LPCA for 360 ° walking motion (camera view and a second view), with results from camera view superimposed on input color images; (top right) results without LPCA for yoga tree motion (camera view and a second view), with results from camera view superimposed on input color images; (bottom right) results with LPCA for yoga tree motion (camera view and a second view), with results from camera view superimposed on input color images.

standing pose is all that we need to feed to our algorithm, which makes the initialization a much easier job. For the skeleton model, currently cylinders and spheres are used to approximate the geometries of all bones. Although we have not rigorously evaluated how sensitive our algorithm is to the accuracy of the skeleton, we use the same skeleton for all the experiments, with the subject wearing three different outfits. However, we believe a more accurate model, e.g. skinned mesh models, will show more promise in the quality of the tracking results. Currently, the parameters for the skeleton model, e.g. the lengths and widths for the limbs, are manually set. It is not difficult, however, to apply a similar optimization procedure to solve for lengths and widths if provided with the ground truth poses. We show that our algorithm can deal with a wider range of motions than Kinect does with the assistance of motion priors. However, it requires the availability of an appropriate database, which contains the same motion patterns as that in the testing sequences. LPCA priors allow the use of a general database instead of a specific one, which decreases the need for user intervention.

Another limitation of the current system is that it does not allow the user to move freely in the space. This is caused by the limited space covered by a single Kinect camera of 43° vertical by 57° horizontal field of view because our algorithm itself is capable of reconstructing motions happening over a region, instead of being limited to motions with overall static root positions/orientations. We demonstrate this capability to some extent through a few examples. For instance, poker face dancing motion is reconstructed with obvious root translations in the direction parallel to the camera image plane, 180° turning motion and 360° walking motion have obvious changes in root orientation. Our system is suitable for a lot applications which expect users to face toward the camera or act in a restricted area.

4. CONCLUSIONS AND FUTURE WORK

Obtaining natural human motion from various inputs at a low cost has been an active research area for a few decades. It remains a challenging problem due to the high complexity of human model and motion. All the existing solutions have their own limitations, e.g. Vicon commercial motion capture system is very expensive, intrusive and requires significant amount of setup and cleanup time despite of its high accuracy, Microsoft Kinect gaming platform often fails to detect certain motions although it is low-cost and allow users to wear everyday apparel. The research work presented in this dissertation aims to address the problem of reconstructing 3D full-body human motion from noisy and ambiguous inputs.

In Section 2, we introduced an algorithm to make the data cleanup of Vicon motion capture an automatic process. The input to the system is either noisy 3D joint position data or noisy joint angle data, and the system automatically filters the noisy data to generate natural human motion data. In Section 3, we presented an approach to reconstruct a wide variety of human motion from a combination of video and depth streams from a single Kinect device. It is effective and low-cost. In general, both work take noisy and ambiguous signals as inputs and solve an under-constrained problem by adding additional constraints from pre-recorded motion capture database. Furthermore, we demonstrated the effectiveness of both work by doing comparisons on real data.

Both work learns statistical priors from the database; however, they differ from each other by the way priors are learned. The work in Section 2 learns a spatial-temporal prior which is based on windows of motion segments and encodes not only the spatial correlation, but also the temporal correlation embedded in the motion database. However, the work in Section 3 learns a sequence of spatial priors to reduce the reconstruction ambiguities at each time step. Temporal information is implicitly encoded by learning each spatial prior based on the previous estimated pose. One of the main reasons for choosing different priors is that the first work is more suitable for an off-line solution while the second one shows more promise in a lot of applications with an online solution.

As any other data-driven approach that uses database, our work bears a similar limitation, which is the need for an appropriate database. Otherwise, if the motion database does not contain similar motion patterns as desired, prior knowledge learned from the database will jeopardize the reconstruction results. We have evaluated how our algorithms respond to general databases, e.g. a walking database of five walking examples with different speeds and step sizes was used for walking in Section 2, and a mixed database of 5 different motion behaviors was used for reconstruction of 4 real sequences in Section 3. In general, the work in Section 2 demands behavior-specific databases while the work in Section 3 tolerates more general databases with different motion patterns. This difference is caused by the fact that the online local models used in Section 3 are capable of extracting a relevant portion from the entire motion capture database dynamically by doing K-NN searches at each time step while the off-line models employed in Section 2 are built on the entire motion capture database at all times.

The work presented in this dissertation along with other related work has demonstrated the effectiveness of data-driven approaches in generating natural motions, however, those statistical models do not consider physics behind motions, such as contact forces, so they lead to physically implausible motions easily, e.g. foot sliding, foot-ground penetrations and motion jerkiness. So combining statistical models with physics models is one of the directions I am interested to explore in future to generate motions that are not only natural, but also physically plausible.

Currently, our work in Section 3 needs pose initialization at the first frame prior to tracking. Although the initialization needs no user interaction, it still restricts the work from being used in a lot of interactive applications. Besides, sequential tracking is inherently subject to error accumulations although the 3D edge component introduced in our work in Section 3 is proved to be effective in reducing such error. This is particularly true in case of fast motions. In the meantime, there have been increasing interests and progress in motion reconstruction by detection. Kinect gaming platform is a great example using this technique. Sequential tracking and detection both have their strengths and are truly complimentary to each other. In future, I would like to study how to combine tracking and detection together to take advantages of both techniques.

The skeleton model used in Section 3 consists of cylinders and spheres only. We believe a more accurate model, e.g. a skinned mesh model, will help increase the tracking accuracy. This can also be studied in future. In addition, we manually set the parameters for the limbs, e.g. the widths and lengths. This limits the scalability and availability of the system considering there exists a wide range of body shapes. However, this problem can be easily solved in a similar manner. Briefly speaking, we can assume known poses by having users imitate several pre-defined poses and solve for skeleton model parameters, which best match the observations.

REFERENCES

[1] Vicon. (2012) Vicon. [Online]. Available: http://www.vicon.com

[2] Microsoft. (2012) Kinect. [Online]. Available: http://www.xbox.com/en-US/kinect

[3] Xsens. (2012) Xsens MVN. [Online]. Available: http://www.xsens.com/en/general/mvn

[4] Ascension. (2012) MotionStar. [Online]. Available: http://www.ascension-tech.com/realtime/RTMotionSTARTethered.php

[5] G. Bishop and G. Welch, "An introduction to the kalman filter," *in Proc. SIGGRAPH Course*, vol. 8, pp. 27 599–3175, 2001.

[6] D. Gavrila, "The Visual Analysis of Human Movement: A Survey," *Computer Vision and Image Understanding (CVIU 1999)*, vol. 73, no. 1, pp. 82–98, 1999.

[7] T. Moeslund, A. Hilton, and V. Kruger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding (CVIU 2006)*, vol. 104, no. 2-3, pp. 90–126, 2006.

[8] D. Forsyth, O. Arikan, L. Ikemoto, J. O'Brien, and D. Ramanan, "Computational studies of human motion: part 1, tracking and motion synthesis," *Foundations and Trends® in Computer Graphics and Vision*, vol. 1, no. 2-3, pp. 77–254, 2005.

[9] R. Poppe, "Vision-based human motion analysis: An overview," *Computer Vision and Image Understanding (CVIU 2007)*, vol. 108, no. 1-2, pp. 4–18, 2007.

[10] J. Lee and S. Shin, "Motion fairing," *Computer Animation (CA 1996)*, pp. 136–143, 1996.

[11] Y. Fangt, C. Hsieh, M. Kim, J. Chang, and T. Woo, "Real time motion fairing with unit quaternions," *Computer-Aided Design*, vol. 30, no. 3, pp. 191–198, 1998.

[12] J. Lee and S. Shin, "A coordinate-invariant approach to multiresolution motion analysis," *Graphical Models (GM 2001)*, vol. 63, no. 2, pp. 87–105, 2001.

[13] ——, "General construction of time-domain filters for orientation data," *IEEE Trans. on Visualization and Computer Graphics (TVCG 2002)*, vol. 8, no. 2, pp. 119–128, 2002.

[14] Vicon. (2012) Vicon Blade. [Online]. Available: http://www.vicon.com/products/blade.html

[15] H. Shin, J. Lee, S. Shin, and M. Gleicher, "Computer puppetry: An importance-based approach," *ACM Trans. on Graphics (TOG 2001)*, vol. 20, no. 2, pp. 67–94, 2001.

[16] S. Tak and H. Ko, "A physically-based motion retargeting filter," *ACM Trans. on Graphics (TOG 2005)*, vol. 24, no. 1, pp. 98–117, 2005.

[17] K. Yamane and Y. Nakamura, "Dynamics filter-concept and implementation of online motion generator for human figures," *IEEE Trans. on Robotics and Automation*, vol. 19, no. 3, pp. 421–432, 2003.

[18] K. Sok, M. Kim, and J. Lee, "Simulating biped behaviors from human motion data," *ACM Trans. on Graphics (TOG 2007)*, vol. 26, no. 3, pp. 107–es, 2007.

[19] J. Wang, S. Drucker, M. Agrawala, and M. Cohen, "The cartoon animation filter," *ACM Trans. on Graphics (TOG 2006)*, vol. 25, no. 3, pp. 1169–1173, 2006.

[20] M. Brand and A. Hertzmann, "Style machines," *in Proc. 27th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 183–192, 2000.

[21] Y. Li, T. Wang, and H. Shum, "Motion texture: a two-level statistical model for character motion synthesis," *in Proc. of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 465–472, 2002.

[22] J. Chai and J. Hodgins, "Constraint-based motion optimization using a statistical dynamic model," *ACM Trans. on Graphics (TOG 2007)*, vol. 26, no. 3, p. 8, 2007.

[23] K. Grochow, S. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," *ACM Trans. on Graphics (TOG 2004)*, vol. 23, no. 3, pp. 522–531, 2004.

[24] J. Chai and J. Hodgins, "Performance animation from low-dimensional control signals," *ACM Trans. on Graphics (TOG 2005)*, vol. 24, no. 3, pp. 686–696, 2005.

[25] O. Arikan, "Compression of motion capture databases," *ACM Trans. on Graphics (TOG 2006)*, vol. 25, no. 3, pp. 890–897, 2006.

[26] L. Ren, A. Patrick, A. Efros, J. Hodgins, and J. Rehg, "A data-driven approach to quantifying natural human motion," *ACM Trans. on Graphics (TOG 2005)*, vol. 24, no. 3, pp. 1090–1097, 2005.

[27] Y. Chen, J. Min, and J. Chai, "Flexible registration of human motion data with parameterized motion models," *in Proc. 2009 Symposium on Interactive 3D Graphics and Games (i3D 2009)*, pp. 183–190, 2009.

[28] L. Ikemoto, O. Arikan, and D. Forsyth, "Knowing when to put your foot down," *in Proc. 2006 Symposium on Interactive 3D Graphics and Games (i3D 2006)*, pp. 49–53, 2006.

[29] H. Von Storch and F. Zwiers, *Statistical analysis in climate research.* New York, NY: Cambridge Univ. Press, 2001.

[30] M. Ghil, M. Allen, M. Dettinger, K. Ide, D. Kondrashov, M. Mann, A. Robertson, A. Saunders, Y. Tian, F. Varadi *et al.*, "Advanced spectral methods for climatic time series," *Rev. Geophys*, vol. 40, no. 1, p. 1003, 2002.

[31] P. Huber, E. Ronchetti, and E. Corporation, *Robust statistics.* Hoboken, NJ: Wiley, 1981, vol. 1.

[32] F. Hampel, E. Ronchetti, P. Rousseeuw, and W. Stahel, *Robust statistics: the approach based on influence functions.* Hoboken, NJ: Wiley, 2011, vol. 114.

[33] T. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature-preserving mesh smoothing," *ACM Trans. on Graphics (TOG 2003)*, vol. 22, no. 3, pp. 943–949, 2003.

[34] S. Fleishman, D. Cohen-Or, and C. Silva, "Robust moving least-squares fitting with sharp features," *ACM Trans. on Graphics (TOG 2005)*, vol. 24, no. 3, pp. 544–552, 2005.

[35] H. Hassani and A. Zhigljavsky, "Singular spectrum analysis: methodology and application to economics data," *Journal of Systems Science and Complexity*, vol. 22, no. 3, pp. 372–394, 2009.

[36] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical recipes (3rd edition): the art of scientific computing.* New York, NY: Cambridge Univ. Press, 2007.

[37] L. Kovar, J. Schreiner, and M. Gleicher, "Footskate cleanup for motion capture editing," *in Proc. 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2002)*, pp. 97–104, 2002.

[38] D. Grest, J. Woetzel, and R. Koch, "Nonlinear body pose estimation from depth images," *Pattern Recognition (PR 2005)*, pp. 285–292, 2005.

[39] D. Grest, V. Krüger, and R. Koch, "Single view motion tracking by depth and silhouette information," *in Proc. 15th Scandinavian Conference on Image Analysis*, pp. 719–729, 2007.

[40] S. Knoop, S. Vacek, and R. Dillmann, "Sensor fusion for 3D human body tracking with an articulated 3D body model," *International Conference on Robotics and Automation (ICRA 2006)*, pp. 1686–1691, 2006.

[41] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," *Computer Vision and Pattern Recognition (CVPR 2011)*, vol. 2, p. 3, 2011.

[42] C. Bregler, J. Malik, and K. Pullen, "Twist based acquisition and tracking of animal and human kinematics," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 179–194, 2004.

[43] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H. Seidel, "Motion capture using joint skeleton tracking and surface estimation," *Computer Vision and Pattern Recognition (CVPR 2009)*, pp. 1746–1753, 2009.

[44] H. Kjellstrom, D. Kragic, and M. Black, "Tracking people interacting with objects," *Computer Vision and Pattern Recognition (CVPR 2010)*, pp. 747–754, 2010.

[45] R. Rosales and S. Sclaroff, "Inferring body pose without tracking body parts," *Computer Vision and Pattern Recognition (CVPR 2000)*, p. 2721, 2000.

[46] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI 2002)*, pp. 509–522, 2002.

[47] K. Grauman, G. Shakhnarovich, and T. Darrell, "Inferring 3D structure with a statistical image-based shape model," *International Conference on Computer Vision (ICCV 2003)*, pp. 641–647, 2003.

[48] N. Howe, "Silhouette lookup for automatic pose tracking," *Computer Vision and Pattern Recognition Workshop (CVPRW 2004)*, pp. 15–22, 2004.

[49] R. Poppe, "Evaluating example-based pose estimation: Experiments on the humaneva sets," *Computer Vision and Pattern Recognition Workshop (CVPRW 2007)*.

[50] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI 1997)*, vol. 19, no. 7, pp. 780–785, 1997.

[51] D. Forsyth and M. Fleck, "Body plans," *Computer Vision and Pattern Recognition (CVPR 1997)*, pp. 678–683, 1997.

[52] T. Roberts, S. McKenna, and I. Ricketts, "Human pose estimation using learnt probabilistic region similarities and partial configurations," *European Conference on Computer Vision (ECCV 2004)*, pp. 291–303, 2004.

[53] X. Ren, A. Berg, and J. Malik, "Recovering human body configurations using pairwise constraints between parts," *International Conference on Computer Vision (ICCV 2005)*, vol. 1, pp. 824–831, 2005.

[54] M. Brand, "Shadow puppetry," *International Conference on Computer Vision (ICCV 1999)*, vol. 2, pp. 1237–1244, 1999.

[55] A. Elgammal and C. Lee, "Inferring 3D body pose from silhouettes using activity manifold learning," *Computer Vision and Pattern Recognition (CVPR 2004)*, vol. 2, pp. II–681, 2004.

[56] A. Agarwal and B. Triggs, "Recovering 3D human pose from monocular images," *IEEE Trans. on Pattern Aanalysis and Machine Intelligence (PAMI 2006)*, pp. 44–58, 2006.

[57] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," *International Conference on Computer Vision (ICCV 2003)*, pp. 750–757, 2003.

[58] E. Ong, A. Micilotta, R. Bowden, and A. Hilton, "Viewpoint invariant exemplar-based 3D human tracking," *Computer Vision and Image Understanding (CVIU 2006)*, vol. 104, no. 2, pp. 178–189, 2006.

[59] N. Howe, "Silhouette lookup for monocular 3D pose tracking," *Image and Vision Computing (IVC 2007)*, vol. 25, no. 3, pp. 331–341, 2007.

[60] R. Wang, S. Paris, and J. Popović, "Practical color-based motion capture," *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (SCA 2011)*, 2011.

[61] I. Kakadiaris and D. Metaxas, "Three-dimensional human body model acquisition from multiple views," *International Journal of Computer Vision (IJCV 1998)*, vol. 30, no. 3, pp. 191–218, 1998.

[62] S. Wachter and H. Nagel, "Tracking persons in monocular image sequences," *Computer Vision and Image Understanding (CVIU 1999)*, vol. 74, no. 3, pp. 174–192, 1999.

[63] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993.

[64] M. Isard and A. Blake, "Condensation-conditional density propagation for visual tracking," *International Journal of Computer Vision (IJCV 1998)*, vol. 29, no. 1, pp. 5–28, 1998.

[65] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," *Computer Vision and Pattern Recognition (CVPR 2000)*, vol. 2, pp. 126–133, 2000.

[66] C. Sminchisescu and B. Triggs, "Covariance scaled sampling for monocular 3D body tracking," *Computer Vision and Pattern Recognition (CVPR 2001)*, vol. 1, pp. I–447, 2001.

[67] F. Caillette, A. Galata, and T. Howard, "Real-time 3-D human body tracking using learnt models of behaviour," *Computer Vision and Image Understanding (CVIU 2008)*, vol. 109, no. 2, pp. 112–125, 2008.

[68] R. Plänkers and P. Fua, "Tracking and modeling people in video sequences," *Computer Vision and Image Understanding (CVIU 2001)*, vol. 81, no. 3, pp. 285–302, 2001.

[69] D. Liebowitz and S. Carlsson, "Uncalibrated motion capture exploiting articulated structure constraints," *International Journal of Computer Vision (IJCV 2003)*, vol. 51, no. 3, pp. 171–187, 2003.

[70] R. Navaratnam, A. Thayananthan, P. Torr, and R. Cipolla, "Hierarchical part-based human body pose estimation," *British Machine Vision Conference*, 2005.

[71] E. De Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Trans. on Graphics (TOG 2008)*, vol. 27, no. 3, p. 98, 2008.

[72] N. Howe, M. Leventon, and W. Freeman, “Bayesian reconstruction of 3D human motion from single-camera video,” *Neural Information Processing Systems (NIPS 1999)*, vol. 1999, p. 1, 1999.

[73] H. Sidenbladh, M. Black, and D. Fleet, “Stochastic tracking of 3D human figures using 2D image motion,” *European Conference on Computer Vision (ECCV 2000)*, pp. 702–718, 2000.

[74] C. Sminchisescu and B. Triggs, “Kinematic jump processes for monocular 3D human tracking,” *Computer Vision and Pattern Recognition (CVPR 2003)*, vol. 1, pp. I–69, 2003.

[75] C. Barrón and I. Kakadiaris, “Monocular human motion tracking,” *Multimedia Systems*, vol. 10, no. 2, pp. 118–130, 2004.

[76] X. Wei and J. Chai, “Videomocap: modeling physically realistic human motion from monocular video sequences,” *ACM Trans. on Graphics (TOG 2010)*, vol. 29, no. 4, pp. 1–10, 2010.

[77] Y. Chen and J. Chai, “3D reconstruction of human motion and skeleton from uncalibrated monocular video,” *Asian Conference on Computer Vision (ACCV 2009)*, pp. 71–82, 2010.

[78] G. Loy, M. Eriksson, J. Sullivan, and S. Carlsson, “Monocular 3D reconstruction of human motion in long action sequences,” *European Conference on Computer Vision (ECCV 2004)*, pp. 442–455, 2004.

[79] B. Rosenhahn, U. Kersting, K. Powell, and H. Seidel, “Cloth x-ray: Mocap of people wearing textiles,” *Pattern Recognition (PR 2006)*, pp. 495–504, 2006.

[80] A. Balan, L. Sigal, M. Black, J. Davis, and H. Haussecker, “Detailed human shape and pose from images,” *Computer Vision and Pattern Recognition (CVPR 2007)*, pp. 1–8, 2007.

[81] J. Starck and A. Hilton, “Surface capture for performance-based animation,” *Computer Graphics and Applications (CGA 2007)*, pp. 21–31, 2007.

[82] H. Li, B. Adams, L. Guibas, and M. Pauly, “Robust single-view geometry and motion reconstruction,” *ACM Trans. on Graphics (TOG 2009)*, vol. 28, no. 5, p. 175, 2009.

[83] Organic Motion. (2012) Organic Motion. [Online]. Available: http://www.organicmotion.com/

[84] Y. Pekelny and C. Gotsman, “Articulated object reconstruction and markerless motion capture from depth video,” *Computer Graphics Forum*, vol. 27, no. 2, pp. 399–408, 2008.

[85] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun, “Real-time identification and localization of body parts from depth images,” *International Conference on Robotics and Automation (ICRA 2010)*, pp. 3108–3113, 2010.

[86] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," *International Conference on Computer Vision (ICCV 2011)*, pp. 731–738, 2011.

[87] M. Siddiqui and G. Medioni, "Human pose estimation from a single view point, real-time range sensor," *Computer Vision and Pattern Recognition Workshops (CVPRW 2010)*, pp. 1–8, 2010.

[88] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real time motion capture using a single time-of-flight camera," *Computer Vision and Pattern Recognition (CVPR 2010)*, pp. 755–762, 2010.

[89] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, "Accurate 3D pose estimation from a single depth image," *International Conference on Computer Vision (ICCV 2011)*, pp. 731–738, 2011.

[90] A. Baak, M. Muller, G. Bharaj, H. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," *International Conference on Computer Vision (ICCV 2011)*, pp. 1092–1099, 2011.

[91] Y. Zhu, B. Dariush, and K. Fujimura, "Kinematic self retargeting: A framework for human pose estimation," *Computer Vision and Image Understanding (CVIU 2010)*, vol. 114, no. 12, pp. 1362–1375, 2010.

[92] V. Pavlovic, J. Rehg, and J. MacCormick, "Learning switching linear models of human motion," *Advances in Neural Information Processing Systems (ANIPS 2001)*, pp. 981–987, 2001.

[93] H. Sidenbladh, M. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking," *European Conference on Computer Vision (ECCV 2002)*, pp. 784–800, 2002.

[94] C. Sminchisescu and A. Jepson, "Generative modeling for continuous non-linearly embedded visual inference," *in Proc. 21st International Conference on Machine Learning*, p. 96, 2004.

[95] A. Agarwal and B. Triggs, "Tracking articulated motion using a mixture of autoregressive models," *European Conference on Computer Vision (ECCV 2004)*, pp. 54–65, 2004.

[96] R. Urtasun, D. Fleet, A. Hertzmann, and P. Fua, "Priors for people tracking from small training sets," *International Conference on Computer Vision (ICCV 2005)*, vol. 1, pp. 403–410, 2005.

[97] R. Urtasun, D. Fleet, and P. Fua, "3d people tracking with gaussian process dynamical models," *Computer Vision and Pattern Recognition (CVPR 2006)*, vol. 1, pp. 238–245, 2006.

[98] R. Li, T. Tian, and S. Sclaroff, "Simultaneous learning of nonlinear manifold and dynamical models for high-dimensional time series," *International Conference on Computer Vision (ICCV 2007)*, pp. 1–8, 2007.

[99] A. Baak, B. Rosenhahn, M. Muller, and H. Seidel, “Stabilizing motion tracking using retrieved motion priors,” *International Conference on Computer Vision (ICCV 2009)*, pp. 1428–1435, 2009.

[100] H. Lou and J. Chai, “Example-Based Human Motion Denoising,” *IEEE Trans. on Visualization and Computer Graphics (TVCG 2010)*, pp. 870–879, 2010.

[101] J. Tautges, A. Zinke, B. Krüger, J. Baumann, A. Weber, T. Helten, M. Müller, H. Seidel, and B. Eberhardt, “Motion reconstruction using sparse accelerometer data,” *ACM Trans. on Graphics (TOG 2011)*, vol. 30, no. 3, p. 18, 2011.

[102] A. Thayananthan, R. Navaratnam, B. Stenger, P. Torr, and R. Cipolla, “Multivariate relevance vector machines for tracking,” *European Conference on Computer Vision (ECCV 2006)*, pp. 124–138, 2006.

[103] C. Sminchisescu, A. Kanaujia, and D. Metaxas, “BM3E: Discriminative Density Propagation for Visual Tracking,” *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI 2007)*, pp. 2030–2044, 2007.

[104] R. Urtasun and P. Fua, “3D human body tracking using deterministic temporal motion models,” *European Conference on Computer Vision (ECCV 2004)*, pp. 92–106, 2004.

[105] R. Li, T. Tian, S. Sclaroff, and M. Yang, “3D human motion tracking with a coordinated mixture of factor analyzers,” *International Journal of Computer Vision (IJCV 2010)*, vol. 87, no. 1-2, pp. 170–190, 2010.

[106] Microsoft. (2012) Kinect SDK. [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows

[107] R. Duda, P. Hart, and D. Stork, *Pattern classification*, 2nd ed. Hoboken, NJ: Wiley, 2000.

[108] H. Abdi and L. Williams, “Principal component analysis,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

VITA

Hui Lou received her B.S. in Computer Science from University of Science and Technology of China (USTC) in 2006. Hui Lou received her Ph.D. degree in Computer Science in May, 2012, under the supervision of Dr. Jinxiang Chai. Her research interests include data-driven animation, computer graphics and computer vision.

Hui Lou can be reached at Department of Computer Science, Texas A&M University, College Station, TX 77843-3112. Her email is: wslh85@gmail.com.

The typist for this dissertation was Hui Lou.