

# Example-based Human Motion Denoising

Hui Lou and Jinxiang Chai

**Abstract**—With the proliferation of motion capture data, interest in removing noise and outliers from motion capture data has increased. In this paper, we introduce an efficient human motion denoising technique for the simultaneous removal of noise and outliers from input human motion data. The key idea of our approach is to learn a series of filter bases from precaptured motion data and use them along with robust statistics techniques to filter noisy motion data. Mathematically, we formulate the motion denoising process in a nonlinear optimization framework. The objective function measures the distance between the noisy input and the filtered motion in addition to how well the filtered motion preserves spatial-temporal patterns embedded in captured human motion data. Optimizing the objective function produces an optimal filtered motion that keeps spatial-temporal patterns in captured motion data. We also extend the algorithm to fill in the missing values in input motion data. We demonstrate the effectiveness of our system by experimenting with both real and simulated motion data. We also show the superior performance of our algorithm by comparing it with three baseline algorithms and to those in state-of-art motion capture data processing software such as Vicon Blade.

**Index Terms**—motion capture, data-driven character animation, motion denoising, statistical motion models, numerical optimization

## 1 INTRODUCTION

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 [29] 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 [25] 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.

This paper 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. 1). The key idea of our approach is to automatically learn a series of spatial-temporal filter bases from prerecorded 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

• H. Lou and J. Chai are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX, 77843-3112. E-mail: wslh@cse.tamu.edu; jchai@cse.tamu.edu

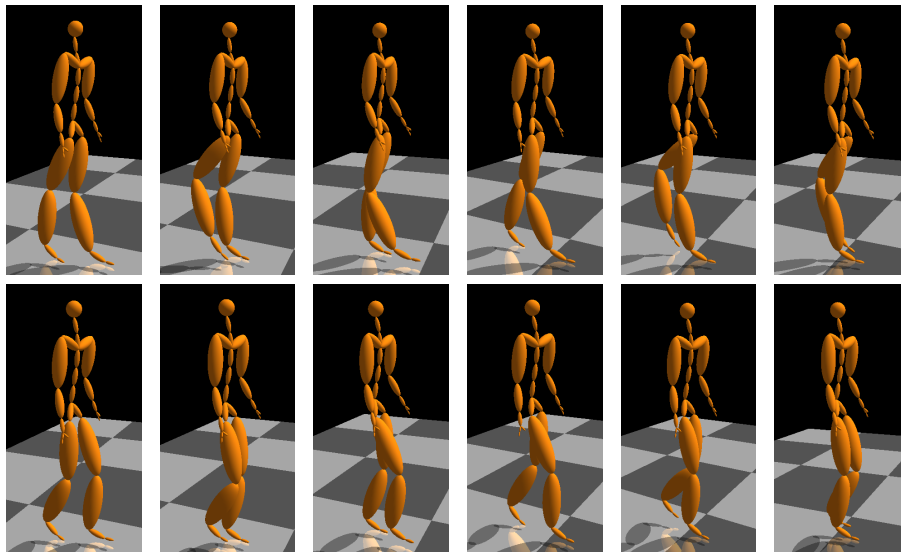


Fig. 1. Example-based human motion denoising: (top) input motion data corrupted by outliers (knee joint); (bottom) filtered motion data.

our algorithm in terms of filter bases learned from different databases and the size of the filtering window.

## 2 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 precaptured 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 [15], [6], [16], [17], [28]. For example, Lee and Shin [15] 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. [6] 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 [17] 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 [28].

An alternative solution is to use the Kalman filter framework [25] to sequentially filter noise present in human motion data [22], [24]. Shin and his colleagues [22] 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 [24] 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 [27], [21] or expressive animation [26]. For example, Yamane and Nakamura [27] introduced a dynamics filter to convert a physically infeasible source motion sequence into a feasible one. Recently, Wang and his colleagues [26] 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 precaptured motion data and applying them in such applications as motion synthesis [2], [18], [4], inverse kinematics [9], [3], motion compression [1], motion quantification [20], motion registration [5], and footskate detection [12]. Our work is different because we construct a series of spatial-temporal filters with multi-channel singular spectrum analysis (M-SSA) [23], [8] and use them for human motion denoising. M-SSA has been successfully applied to many practical problems in geophysics [8]. 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 [11], [10] 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 [13] and surface reconstruction [7].

### 3 HUMAN MOTION DENOISING

The goal of our paper 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 pre-recorded 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 precaptured motion data (Section 3.1), and then discuss how to deal with outliers from noisy motion data (Section 3.2). We formulate the motion denoising problem in a nonlinear optimization framework and define the object function for human motion denoising (Section 3.3). We develop an iterative algorithm for efficient optimization of the object function (Section 3.4). Finally, we discuss how to extend the framework to filling in missing values (Section 3.5).

#### 3.1 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) [23], [8], 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 precaptured 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  $\{x_l(t) : l = 1, \dots, L; t = 1, \dots, N\}$  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  $\mathbf{x}_t = [x_1(t), \dots, x_L(t)]^T$  represent a full-body character pose at frame  $t$ . Similarly, let  $\{y_l(t) : l = 1, \dots, L; t = 1, \dots, T\}$  be a sequence of noisy input motion data and let  $\mathbf{y}_t = [y_1(t), \dots, y_L(t)]^T$  denote a character pose at frame  $t$ .

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

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

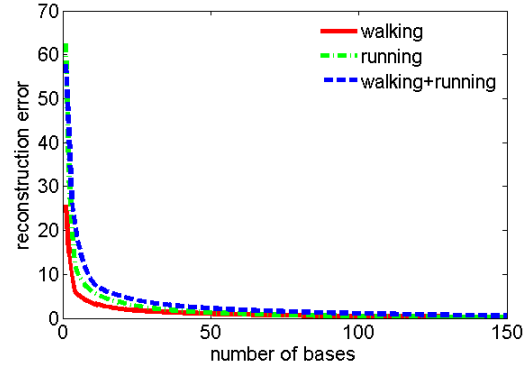


Fig. 3. Reconstruction errors vs. the number of filter bases used within a finite size (20) window. The reconstruction errors are evaluated via cross-validation techniques.

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.

We further form a fully augmented trajectory matrix:

$$D = (X_1 \ X_2 \ \dots \ X_L). \quad (2)$$

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:

$$C_D = \frac{D^T D}{S_N} = \begin{pmatrix} C_{1,1} & C_{1,2} & \dots & C_{1,L} \\ \vdots & C_{2,2} & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ C_{L,1} & C_{L,2} & \dots & C_{L,L} \end{pmatrix}, \quad (3)$$

where the blocks of  $C_D$  are given by

$$C_{l,l'} = \frac{X_l^T X_{l'}}{S_N}, \quad l, l' = 1, \dots, L, \quad (4)$$

with entries

$$(C_{l,l'})_{j,j'} = \frac{\sum_{n=1}^{S_N} x_l(n+j-1)x_{l'}(n+j'-1)}{S_N}, \quad j, j' = 1, \dots, M. \quad (5)$$

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 correlation 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  $M \times L$  eigenvectors  $\{\mathbf{e}^k \in R^{M \times L} | k = 1, \dots, M \times L\}$ . The  $M \times L$  eigenvectors provide a set of orthogonal filter bases, which

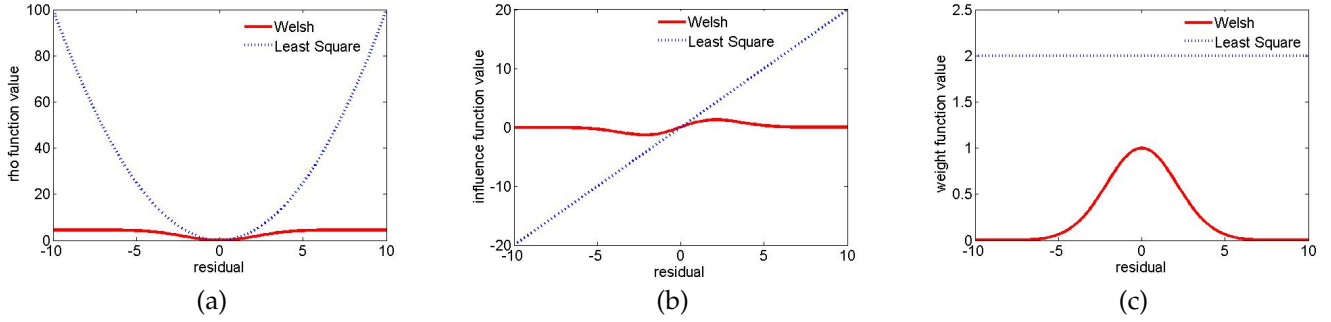


Fig. 2. The robust estimator (Welsch function) vs. the least-square estimator: (a) the function for measuring the residual distance ( $r$ ) between the reconstructed motion and noisy measurement ( $\rho(r)$ ); (b) the influence function for each data point ( $\phi(r) = \frac{\partial \rho(r)}{\partial r}$ ); (c) the weights for each data point ( $\frac{\phi(r)}{r}$ ). Note that the weights for outliers become zero for robust estimators.

can be used to reconstruct any segment of human motion data within a finite size window  $M$ :

$$\mathbf{y}_{1:M} = \sum_{k=1}^{M \times L} \langle \mathbf{e}^k, \mathbf{y}_{1:M} \rangle \mathbf{e}^k, \quad (6)$$

where the vector  $\mathbf{y}_{1:M} \in R^{M \times L}$  sequentially stacks all poses across the entire window and the operator  $\langle \rangle$  represents the dot product between two vectors. In practice, human movement is highly correlated. Therefore, a small number of patterns are often sufficient to represent natural human motion variation within a finite size window. Fig. 3 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:

$$\begin{aligned} \mathbf{z}_{1:M} &= \sum_{k=1}^K c_k \mathbf{e}^k \\ &= \sum_{k=1}^K \langle \mathbf{e}^k, \mathbf{y}_{1:M} \rangle \mathbf{e}^k, \end{aligned} \quad (7)$$

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  $c_k, k = 1, \dots, K$  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  $C_D$ . 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.

### 3.2 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 (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 [11], [10].

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

$$E_{data}(\mathbf{z}_{1:T}, \mathbf{y}_{1:T}) = \sum_t \sum_l \rho(z_l(t) - y_l(t)), \quad (8)$$

where  $E_{data}$  measure the distance between the noisy input and the filtered motion. The robust estimator  $\rho$  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). 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 [11], [10].

Mathematically, the Welsch robust function is defined as follows:

$$\rho(r) = \frac{p^2}{2} (1 - \exp(-\frac{r^2}{p^2})), \quad (9)$$

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 (8). We experimentally set the parameter of Welsch estimator ( $p$ ) to 3.

### 3.3 Objective Function

We now combine the data-driven filter bases  $\mathbf{e}_k \in R^{M \times L}, k = 1, \dots, K$  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  $\mathbf{e}_k, k = 1, \dots, K$  embedded in precaptured motion data.

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

$$\hat{\mathbf{c}}, \hat{\mathbf{z}}_{1:M} = \arg \min_{\mathbf{c}, \mathbf{z}_{1:M}} \sum_{t=1}^M \sum_{l=1}^L \rho(z_l(t) - y_l(t)) + \lambda \|\mathbf{z}_{1:M} - E\mathbf{c}\|^2, \quad (10)$$

where the matrix  $E = [\mathbf{e}_1 \dots \mathbf{e}_K]$  stacks the filter bases  $\mathbf{e}_k, k = 1, \dots, K$  constructed from training data. The vector  $\mathbf{c} = [c_1, \dots, c_K]^T$  stacks all the filter weights. The first term, which is the robust estimation term defined in Equation (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 (7), and measures how well the filtered motion matches the spatial-temporal patterns embedded in precaptured motion data. The weight  $\lambda$  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:

$$\{\hat{\mathbf{c}}_s, \hat{\mathbf{z}}_t\} = \arg \min_{\{\mathbf{c}_s\}, \{\mathbf{z}_t\}} \sum_{t=1}^T \sum_{l=1}^L \rho(z_l(t) - y_l(t)) + \lambda \sum_{s=1}^S \|\mathbf{z}_{s,1:M} - E\mathbf{c}_s\|^2, \quad (11)$$

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.

### 3.4 Iterative Optimization

The overall cost function (Equation 11) includes two groups of unknowns: the filtering coefficients  $\mathbf{c}_s, s = 1, \dots, T - M + 1$  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.

#### 3.4.1 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:

$$\hat{\mathbf{c}}_s = \arg \min_{\mathbf{c}_s} \|\hat{\mathbf{z}}_{s,1:M} - E\mathbf{c}_s\|^2, \quad s = 1, \dots, S. \quad (12)$$

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

$$\{\hat{\mathbf{c}}_s\} = E^T \hat{\mathbf{z}}_{s,1:M}, \quad s = 1, \dots, S. \quad (13)$$

#### 3.4.2 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:

$$\hat{\mathbf{z}}_t = \arg \min_{\mathbf{z}_t} \lambda \sum_{m=\max\{1, t-T+M\}}^{m=\min\{M, t\}} \|\mathbf{z}_t - E_m \hat{\mathbf{c}}_{t-m+1}\|^2 + \sum_l \rho(z_l(t) - y_l(t)), \quad t = 1, \dots, T, \quad (14)$$

where the vector  $\mathbf{z}_t$  represents the filtered pose at frame  $t, t = 1, \dots, T$ . The matrix  $E_m$  is an  $L \times K$  matrix that stacks rows  $M(l-1) + m, l = 1, \dots, L$  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 [10], [19]. After applying IRLS to the objective function in Equation (14), we can iteratively solve the following weighted least-square problem:

$$\hat{\mathbf{z}}_t = \arg \min_{\mathbf{z}_t} \lambda \sum_{m=\max\{1, t-T+M\}}^{m=\min\{M, t\}} \|\mathbf{z}_t - E_m \hat{\mathbf{c}}_{t-m+1}\|^2 + \sum_l w_l(t) (z_l(t) - y_l(t))^2, \quad t = 1, \dots, T, \quad (15)$$

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

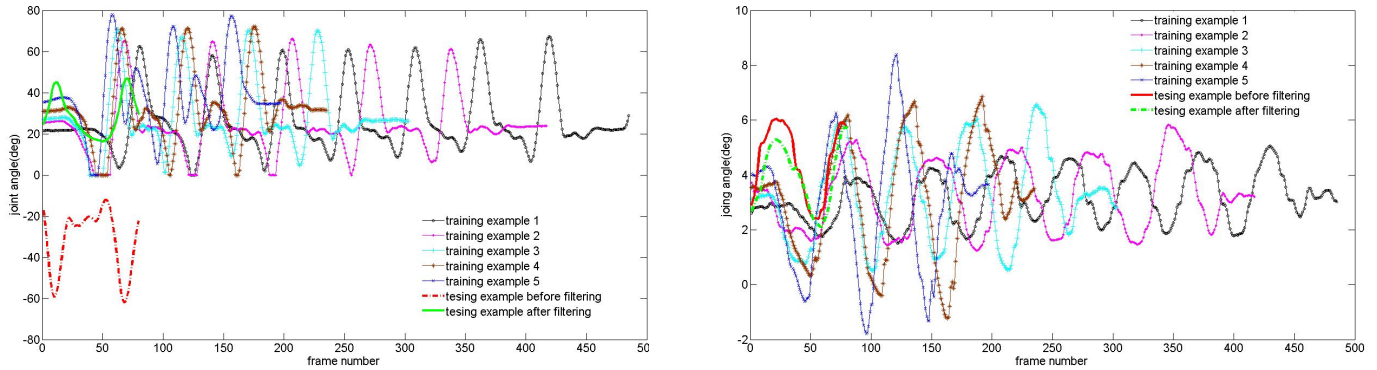


Fig. 4. Motion generalization: (left) joint angle values (knee) of training data set, noisy motion data and filtered motion data; (right) joint angle values (thorax) of training data set, noisy motion data and filtered motion data. Note that knee joint is corrupted by outliers across the entire motion while the thorax joint is not corrupted by any outliers.

### 3.4.3 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  $\hat{\mathbf{z}}_{1:T}^0 = \mathbf{y}_{1:T}$
2. Update the coefficients  $\hat{\mathbf{c}}_s$  and the motion  $\hat{\mathbf{z}}_{1:T}$ 
  - 2.1. Update  $\hat{\mathbf{c}}_s = E^T \hat{\mathbf{z}}_{s,1:M}$   $s = 1, \dots, S$
  - 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).

### 3.4.4 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 [14] to correct the footskate artifact.

### 3.5 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  $\alpha_{l,t} \in \{0, 1\}$ ,  $l = 1, \dots, L$ ,  $t = 1, \dots, T$  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:

$$\{\hat{\mathbf{c}}_s, \hat{\mathbf{z}}_t\} = \arg \min_{\{\mathbf{c}_s\}, \{\mathbf{z}_t\}} \sum_{t=1}^T \sum_{l=1}^L \alpha_{l,t} \rho(z_l(t) - y_l(t)) + \lambda \sum_{s=1}^S \|\mathbf{z}_{s,1:M} - E\mathbf{c}_s\|^2. \quad (16)$$

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

## 4 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. 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

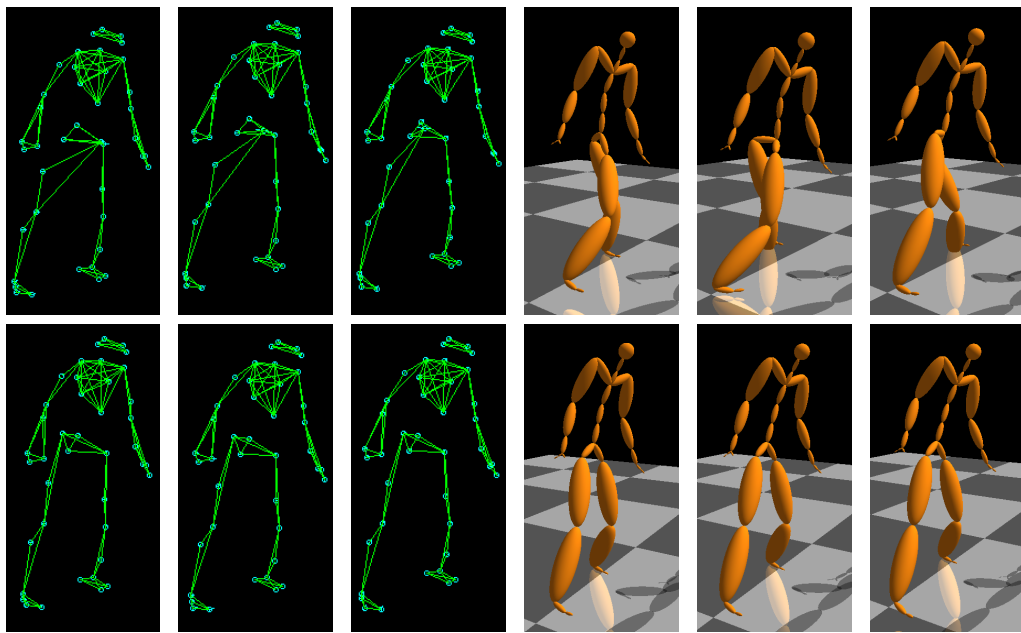


Fig. 5. Filtering real noisy motion data in 3D position space: (top left) the corrupted motion data in position space; (bottom left) the filtered motion data in position space; (top right) the corrupted motion data in joint-angle space; (bottom right) the filtered motion data in joint-angle space.

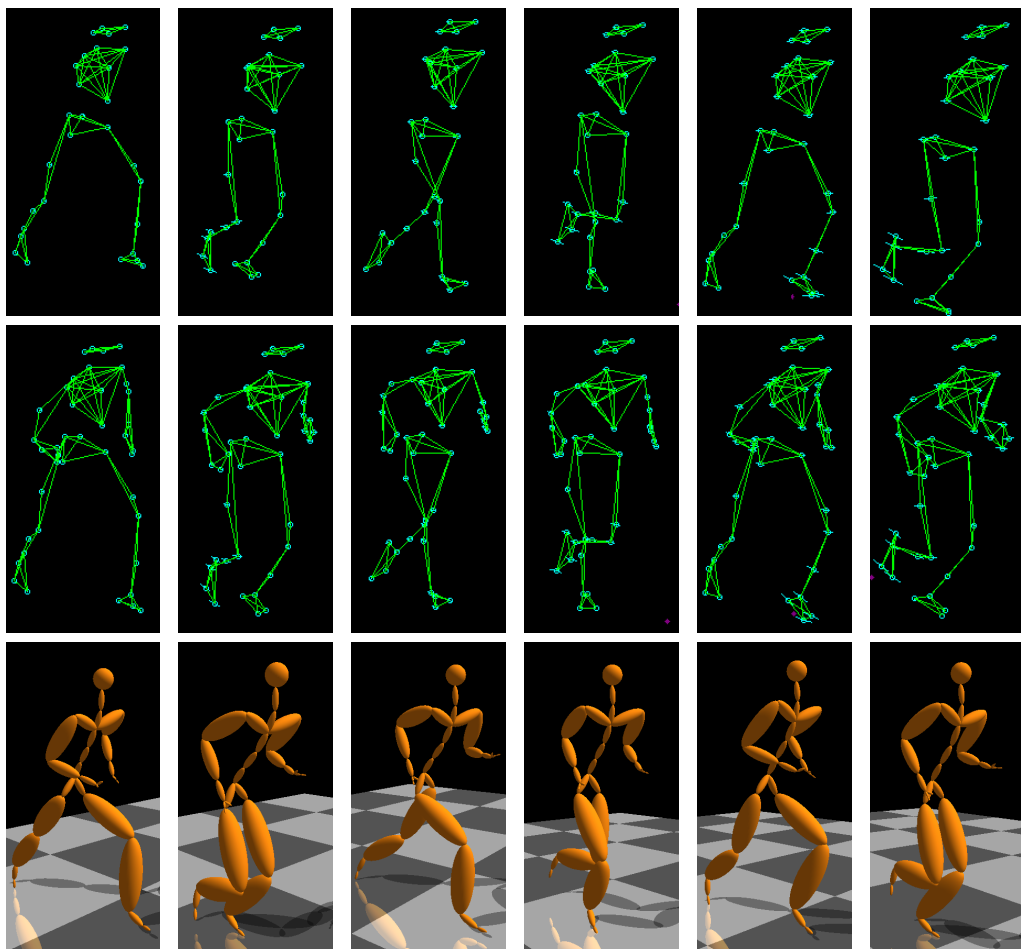


Fig. 6. Filling in missing data in the 3D position space: (top) missing motion data in 3D position space; (middle) completed motion data in 3D position space; (bottom) completed motion data in joint-angle space.

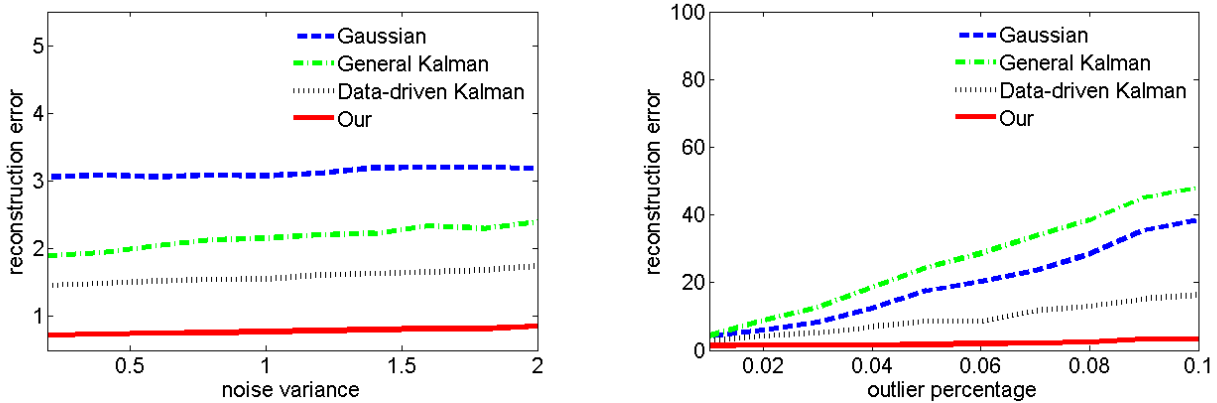


Fig. 7. Comparisons with alternative signal denoising techniques: (left) reconstruction errors vs. different noise levels; (right) reconstruction errors vs. different percentages of outliers.

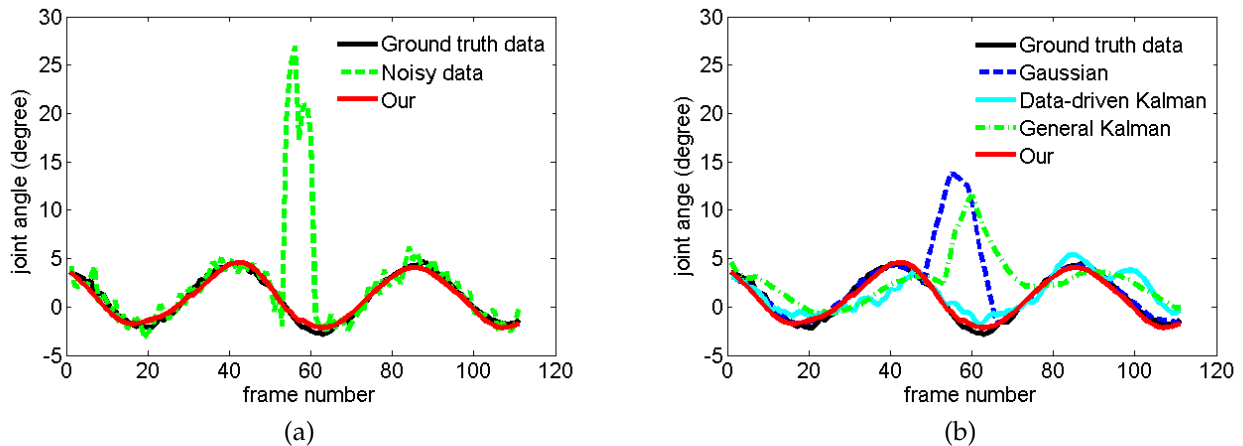


Fig. 8. Comparisons of joint angle values (thorax): (a) our result vs. ground truth data; (b) our result vs. results obtained by three baseline filtering algorithms.

the  $K$  largest eigen values is approximately 99% of the total energy.

#### 4.1 Testing on Real Data

Due to occlusions and marker mislabeling, real mocap 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. 1 and Fig. 5 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 mocap data reported here were downloaded from the CMU online mocap library<sup>1</sup>. 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. 4 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. 6 shows sample frames of our results, where all the markers on both arms are completely missing.

#### 4.2 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 [25], 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

1. <http://mocap.cs.cmu.edu>

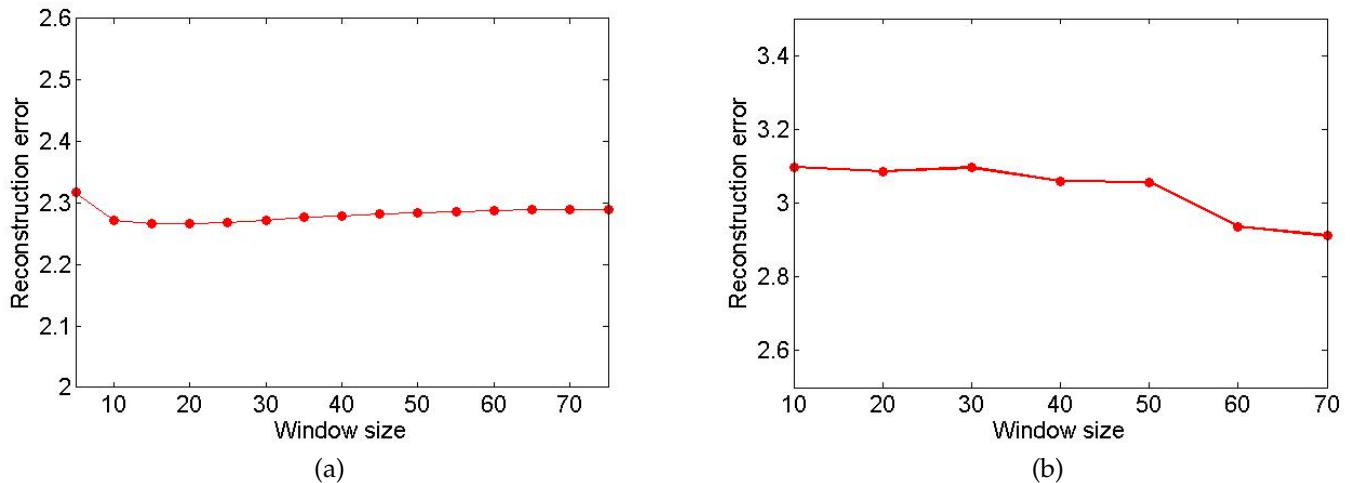


Fig. 9. Evaluation of the performance of the denoising algorithm with respect to different window sizes: (a) filtering a noisy walking sequence ( $\sigma = 3$ ); (b) filling in missing values in a jumping sequence. The errors are measured by degrees per joint angle per frame.

matrices to identity matrices by simply choosing the 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. 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. 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.

### 4.3 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 mocap 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 motion. The system can also produce a good result when we filter a noisy running sequence with filter bases learned from the mixed database.

**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. 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. 9.(b) shows the filtering errors for filling in missing values in a jumping sequence with respect to different window

sizes.

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. 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.

## 5 DISCUSSION AND CONCLUSION

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 precaptured 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. 4 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  $\mathbf{e}_k, k = 1, \dots, K$ , filtered motion  $\hat{\mathbf{z}}_{1:T}$ , and coefficients  $\hat{\mathbf{c}}_s, s = 1, \dots, S$  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.

## REFERENCES

- [1] O. Arikan, "Compression of motion capture databases," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 890-897, 2006.
- [2] M. Brand and A. Hertzmann, "Style machines," *ACM Siggraph*, pp. 183-192, 2000.
- [3] J. Chai and J. Hodgins, "Performance animation from low-dimensional control signals," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 686-696, 2005.
- [4] J. Chai and J. Hodgins, "Constraint-based motion optimization using a statistical dynamic model," *ACM Transactions on Graphics*, vol. 26, no. 3, article no. 8, 2007.
- [5] Y.-L. Chen, J. Min, and J. Chai, "Flexible registration of human motion data with parameterized motion models," *ACM Symposium on Interactive 3D Graphics and Games (i3D 2009)*, pp. 183-190, 2009.
- [6] Y. Fang, C. C. Hsieh, M. J. Kim, J. J. Chang, and T. C. Woo, "Real time motion fairing with unit quaternions," *Computer-Aided Design*, vol. 30, no. 3, pp. 191-198, 1998.
- [7] S. Fleishman, D. Cohen-or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 554-552, 2005.
- [8] M. Ghil, M. R. Allen, M. D. Dettinger, K. Ide, D. Kondrashov, M. E. Mann, A. W. Robertson, A. Saunders, Y. Tian, F. Varadi, and P. Piou, "Advanced spectral methods for climatic time series," *Reviews of Geophysics*, vol. 40, no. 1, pp. 3.1-3.41, 2002.
- [9] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 522-531, 2004.
- [10] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust statistics: the approach based on influence functions*, Wiley.
- [11] P. Huber, *Robust statistics*, Wiley.
- [12] L. Ikemoto, O. Arikan, and D. Forsyth, "Knowing when to put your foot down," *ACM Symposium on Interactive 3D Graphics and Games (i3D 2006)*, pp. 49-53, 2006.
- [13] T. R. Jones, F. Durand, and M. Desbrun, "Noniterative, feature-preserving mesh smoothing," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 943-949, 2003.
- [14] L. Kovar, J. Schreiner, and M. Gleicher, "Footskate cleanup for motion capture editing," *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 97-104, 2002.
- [15] J. Lee and S. Y. Shin, "Motion fairing," *Proceedings of Computer Animation 96*, pp. 136-143, 1996.
- [16] J. Lee and S. Y. Shin, "A coordinate-invariant approach to multiresolution motion analysis," *Graphical Models and Image Processing*, vol. 63, no. 2, pp. 87-105, 2001.
- [17] J. Lee and S. Y. Shin, "General construction of timedomain filters for orientation data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 2, pp. 119-128, 2002.
- [18] Y. Li, T. Wang, and H.-Y. Shum, "Motion texture: A two-level statistical model for character synthesis," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 465-472, 2002.
- [19] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical recipes in C: the art of scientific computing*, second edition, Cambridge University Press, New York, 1992.
- [20] L. Ren, A. Patrick, A. A. Efros, J. K. Hodgins, and J. M. Rehg, "A data-driven approach to quantifying natural human motion," *ACM Transactions on Graphics*, vol. 24, No. 3, pp. 1090-1097, 2005.
- [21] K. W. Sok, M. Kim, and J. Lee, "Simulating biped behaviors from human motion data," *ACM Transactions on Graphics*, vol. 26, no. 3, article no. 3, 2007.
- [22] H. J. Shin, J. Lee, M. Gleicher, and S. Y. Shin, "Computer puppetry: an importance-based approach," *ACM Transactions on Graphics*, vol. 20, no. 2, pp. 67-94, 2001.
- [23] H. V. Storch and F. W. Zwiers, *Statistical analysis in climate research*, Cambridge University Press, 1999.
- [24] S. Tak and H.-S. Ko, "A physically-based motion retargeting filter," *ACM Transactions on Graphics*, vol. 24, no. 1, pp. 98-117, 2005.

- [25] G. Welch and G. Bishop, "An introduction to the kalman filter," *ACM SIGGRAPH 2001 Course Notes*, 2001.
- [26] J. Wang, S. Drucker, M. Agrawala, and M. Cohen, "The cartoon animation filter," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1169-1173, 2006.
- [27] K. Yamane and Y. Nakamura, "Dynamics filter - concept and implementation of online motion generator for human figures," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 421-432, 2003.
- [28] Vicon blade, <http://www.vicon.com/products/blade.html>, 2009.
- [29] Vicon Systems, <http://www.vicon.com>, 2009.