Real-Time Posture Reconstruction for Microsoft Kinect

Hubert P. H. Shum, Edmond S. L. Ho, Yang Jiang, and Shu Takagi

***Abstract*—The recent advancement of motion recognition using Microsoft Kinect stimulates many new ideas in motion capture and virtual reality applications. Utilizing a pattern recognition algorithm, Kinect can determine the positions of different body parts from the user. However, due to the use of a single depth camera, recognition accuracy drops significantly when the parts are occluded. This hugely limits the usability of applications that involves interaction with external objects, such as sport training or exercising systems. The problem becomes more critical when Kinect incorrectly perceives the body parts. This is because applications have limited information about the recognition correctness, and using those parts to synthesize a body postures would result in serious visual artifacts. In this paper, we propose a new method to reconstruct valid movement from incomplete and noisy postures captured by Kinect. We first design a set of measurements that objectively evaluates the degree of reliability on each tracked body part. By incorporating the reliability estimation into a motion database query during run-time, we obtain a set of similar postures that are kinematically valid. These postures are used to construct a latent space, which is known as the *natural posture space* in our system, with local Principle Component Analysis (PCA). We finally apply frame-based optimization in the space to synthesize a new posture that closely resembles the true user posture while satisfying kinematic constraints. Experimental results show that our method can significantly improve the quality of the recognized posture under severely occluded environments, such as a person exercising with a basketball or moving in a small room.**

***Index Terms*—Kinect, posture reconstruction, local principal component analysis, human computer interaction.**

I. Introduction

USing the newly introduced Microsoft Kinect, it becomes possible to recognize human movements with easy hardware setup. This fuses a large number of new research ideas on motion based systems, including entertainment applications such as motion gaming, as well as serious applications like sport training.

The major challenge of motion recognition with Kinect is the noisiness and incompleteness of the tracked postures. Kinect is a vision-based motion capture system using an infrared sensor to obtain depth information. Therefore, it suffers from similar problems with traditional optical motion capturer, such as occlusions and mixing up of tracked body parts. In our primary work [1], we report that a small amount of noise can be corrected with pre-captured motions and a physical simulation engine.

The posture recognition problem becomes more challenging when the user interacts with external objects. This is because Kinect has a single inferred sensor and every tracked part requires a direct line of sight. Consequently, Kinect applications usually require the user to face the sensor all the time without holding any objects. This hugely limits the usability of the applications, especially if we use Kinect to implement systems that require handling of equipment such as sport training. Imagine a user exercising with a basketball or a dumbbell weight when some parts of the body are occluded from the camera. Kinect will fail to accurately recognize the posture, resulting in unreliable or missing body parts. Using such a posture to construct a virtual character, or to interact with the application, would result in serious artifacts.

While previous research can reconstruct broken postures into more plausible ones, they cannot be directly applied in the Kinect problem. First, the proportion of body parts that are lost can be large and can be different from frame to frame. Second, Kinect returns a pose consisting of both correctly and incorrectly tracked parts, while the applications have limited knowledge about the recognition correctness. As a result, previous algorithms on low dimensional control that assume a relatively stable control signal [2] would not work, as the signal from Kinect is not consistent. On the other hand, simply using all tracked parts [3] results in poor accuracy, because some of them are incorrect.

In this paper, we propose a new method to measure the reliability of the tracked body parts. This is particular important in Kinect because the parts include a relative high number of false positive. An advantage of our designed measurement is that it relies on general behavioral information about the tracked parts, making it applicable to different tracking frameworks with different types of source data. Our measurement returns a degree of reliability instead of a Boolean value, such that we can make use of the parts that are neither completely correct nor incorrect. This is essential to achieve good recognition under extreme situations when many parts are lost.

Using the reliability measurement, we propose a new method to correct broken postures with a motion database. We first incorporate the calculated reliability value as weights into a motion database query to extract a set of similar postures that are kinematically valid. Then, using these postures, we construct a local latent space called the *natural posture space* with local Principle Component Analysis (PCA). Because of the high similarity of the source data, the constructed natural

Hubert P. H. Shum is with the Faculty of Engineering and Environment, Northumbria University, United Kingdom, e-mail: hubert.shum@northumbria.ac.uk

Edmond S. L. Ho is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, e-mail: edmond@comp.hkbu.edu.hk

Yang Jiang is with the Faculty of Engineering and Environment, Northumbria University, United Kingdom, e-mail: yang.jiang@northumbria.ac.uk

Shu Takagi is with the Department of Mechanical Engineering, The University of Tokyo, Japan, email: takagi@mech.t.u-tokyo.ac.jp

posture space is in low dimensional and smooth. We optimize in the space for a posture that fits well with the tracked parts while satisfying kinematic constrains. Finally, the result is passed to a physically simulated character for further enforcing kinematics features such as segment length and movement stability.

With our proposed framework, it is possible to utilize Kinect for applications such as real-time sport training, in which the users usually need to interact with large external objects. Experimental results show that our method performs effectively under extreme situation, such as when a large portion of the body is being occluded. We show practical examples when the user interacts with a box, plays with a basketball, and moves in a small room. We also give detailed analysis to show the high accuracy on the synthesized movements. While we implement our system and perform experiments with Kinect in this paper, the framework is general and is applicable to other tracking systems, such as the optical motion capturer.

The rest of the paper is organized as follow. We review related research in Section II. We then give an overview of our system and point out our major contributions in Section III. The core of our framework involves four major parts. First, we design a reliability measurement to evaluate the reliability of the tracked body parts by Kinect (Section IV). Second, we construct a natural posture space by querying a motion database with the Kinect posture and the reliability value (Section V). Third, we synthesize a new posture that fit well with the Kinect posture while satisfying kinematics constraints (Section VI). Forth, we apply a physical simulated character as a kinematics filter to generate natural movements (Section VII). Experimental results and system analysis are detailed in Section VIII. Finally, we discuss our method and future directions in Section IX.

II. Related Work

In this section, we first review previous work on Kinect based motion tracking. We then focus on the problem of reconstructing postures from an incomplete or corrupted data stream. We finally review the weighted principle component analysis framework that we adopt in this research.

A. Motion Tracking with Kinect

Here, we review different Kinect-related research, focusing on those related to motion analysis and synthesis. We point out the tracking inaccuracy problem of Kinect and discuss possible solutions.

Kinect uses an infrared sensor to capture a projected pattern and construct a depth image, which describes the distance between the sensor and each pixel. It fuses a wide variety of new research, including gesture recognition and control [4], natural user interface [5] and 3D surface reconstructions [6], [7]. New algorithms are proposed to estimate human postures from the single depth image [8], [9], [10], [11]. They provide economical ways for 3D motion acquisition. A comprehensive review of Kinect research can be found in [12]. We obtained user postures using the Kinect SDK [13], which involves training a classifier with a large amount of synthesized data to identify different body parts [14]. These postures are considered as source data in our system.

Postures recognized by Kinect are noisy. Raptis et al. propose a dance motion classification framework for real-time applications [15]. While the system is unaffected by the noise of the input data, dance-specific assumptions have to be made to achieve good robustness. Bailey et al. compare the perceived quality of the animations generated by the motions captured from Kinect and those from commercial optical motion capture system [16]. They apply a Butterworth filter to smooth the noisy motion. A similar analysis has been carried out by Fern’ndez-Baena et al. for rehabilitation applications [17]. While the study suggests that Kinect tracking is reliable, it assumes the motions to be performed in open area with no occlusion, which is not possible for applications involving highly dynamic movements and object manipulations.

A major reason of the tracking inconsistency is that body parts are tracked separately. The posture recognized does not necessary satisfy kinematic constraints such as segment length. Ye et al. match the depth image with postures from a motion database to identify different body parts [18]. However, the method requires the full body posture to be clearly visible from the depth image. Shum and Ho use a motion database to fill in the missing degree of freedom in the Kinect tracked posture, and apply a physical simulation engine to enforce kinematic features [1]. Because of the lack of reliability measurement on the Kinect tracked parts, the quality of the synthesized posture is heavily affected whenever Kinect percepts incorrectly. Shen et al. proposed an exemplar-based approach to correct 3D poses estimated from depth images captured by Kinect [19]. Although the method improves wrongly recognized posture from Kinect, only golf swinging motions are considered. It is unclear if the algorithm works when the database contains multiple classes of motion, and if the performance is consistent across different types of target movements. Our research targets in reconstructing postures from different motions using a general purpose motion database.

B. Posture Reconstruction

Here, we discuss how human posture from low-dimensional or corrupted source data can be reconstructed, and highlight the need of a reliability measurement for the source data.

The full body postures of the user can be approximated by a small number of sensors, such as the positions of a few reflective markers [3], the readings of some inertial sensors attached to the upper body [20], and the readings of 3D accelerometers attached only to the limbs [2], [21]. Such a posture reconstruction problem is challenging since the rest of body has to be estimated. Due to the high-dimensionality of human movements, it leads to an under-determined system that contains many solutions.

To obtain the right solution that produces natural-looking results, researchers have worked on creating a natural human posture space, and using pre-recorded human motion to constraint the solution space. Chai and Hodgins [3] propose a method based on the lazy learning algorithm [22] to reconstruct a full body posture from low-dimensional marker positions. Specifically, motion samples that are similar to the input signals are selected to construct a locally linear space using Principal Component Analysis (PCA) [23], which is then used as priors to constrain the reconstructed pose. Liu et al. propose to create the local models during run-time [20]. They reconstruct the new posture with the maximum a posteriori framework (MAP), which takes into account the spatial-temporal correlation patterns extracted from the sample motions. The major challenge of these methods is to construct the local models in real-time for every frame, which is computational costly especially for larger motion database.

To tackle this performance issue, Wei and Chai [24] propose to learn a probabilistic model as the pose priors from the sample motions using the mixture of factor analyzers (MFAs) [25], [26]. MFA partitions the entire configuration space into multiple local regions and represents each region with a small number of latent variables. Since the poses priors are pre-computed by MFA, the run-time performance is significantly improved. Grochow et al. [27] compute a probabilistic model using the scaled Gaussian process latent variable model (SG-PLVM) [28] from motion data, and find the most-likely pose according to the low-dimensional constraints given by the user. However, due to the high complexity of the method, the training time increases nearly cubically with the size of the training data, which limits the scale of the data set. Wu et al. improve the performance by selecting representative poses and train the pose priors with Gaussian Processes (GPs) [29]. They further propose to use fully independent training conditional (FITC) [30], [31] approximation to speed up the training process of GPs. As a result, it becomes feasible to learn the model from a large data set and cover a wider range of movements.

While it is possible to reconstruct natural-looking postures from low-dimensional input signals, these methods assume a reliable low-dimensional input signals, whereas motions from Kinect are noisy with heavy tracking error. A method to identify incorrectly tracked parts is needed. Lou and Chai [32] propose a data-driven method to correct noisy motion data caused by outliers and missing markers. The system learns a series of filter bases from sample motions, and spatial-temporally optimized the corrupted motion data to remove the noise. However, space-time optimization is computationally costly, and can only be applied when the whole motion is available, making it unsuitable for motions that are captured on-the-fly. We propose a set of reliability measurement to assess the body parts recognized by Kinect, and reconstruct the posture using the more reliable parts.

C. *Weighted Principal Component Analysis*

Here, we review the work in weighted PCA (wPCA), which takes the importance of the data into account when training the low dimensional models. We adopt the algorithm to weight body parts based on their respective reliability during the motion reconstruction process.

Yue and Tomoyasu propose a weighting framework to traditional PCA to improve its performance on fault detection and correction problems [33]. They introduce a two-levels weighting approach to maintain the accuracy and validity of the PCA model over time. The training data are weighted according to the importance in the sample-wise level and the variable-wise level.

The presence of outliers can significantly degrade the constructed PCA model. Kriegel et al. evaluate the impact of outliers and assign lower weights to the potential outliers to reduce their influences [34]. Pinto da Costa et al. also express similar concerns about PCA being sensitive towards noises when being used to analyze gene expression data in bioinformatics [35].

Forbes and Fiume propose using weighted PCA to improve the motion searching accuracy [36]. Based on a query motion, the system assigns heavier weight to more important body parts, such as the hands in a carrying motion. It then projects the sample motions and the query motion into the wPCA space with the assigned weight. Notice that once the weight has changed, the wPCA space has to be reconstructed. We propose to compose a local PCA space during run-time to minimize the space construction cost, and apply the concept of weight to minimize the influence of mis-recognized body parts.

III. Overview

Figure 1 shows the overview of our system. We first evaluate the reliability of different body parts in the Kinect recognized posture, in which some parts are occluded. Based on the reliability values, we conduct a database query to extract the K nearest neighbour postures. The postures are used to construct a natural posture space, with which we conduct a posture optimization process to synthesize a correct posture. Finally, we use the reconstructed posture to control a physically simulated character that produces natural movements.

A. *Contributions*

In this paper, we have three major contributions:

- We propose a measurement for evaluating the reliability of tracked body parts, which is represent in a continuous scale. The measurement considers abnormal behaviours, kinematics features and sensor status. It identifies false positive tracked parts in Kinect for more accurate posture reconstruction.
- We propose an algorithm to construct a natural posture space with local PCA considering the reliability values. It is a low dimensional space composed with pre-captured posture samples that are kinematically similar to the Kinect posture. It facilitates efficient and reliable posture synthesis for real-time applications.
- We propose a posture reconstruction framework using the natural posture space to reconstruct a valid posture from incomplete and noisy Kinect data in real-time. We optimize for a posture that corrects the non-reliable body parts from Kinect, while satisfying the positions of the reliable ones and obeying kinematics constraints. It allows us to reconstruct the posture of a user even if it is occluded.

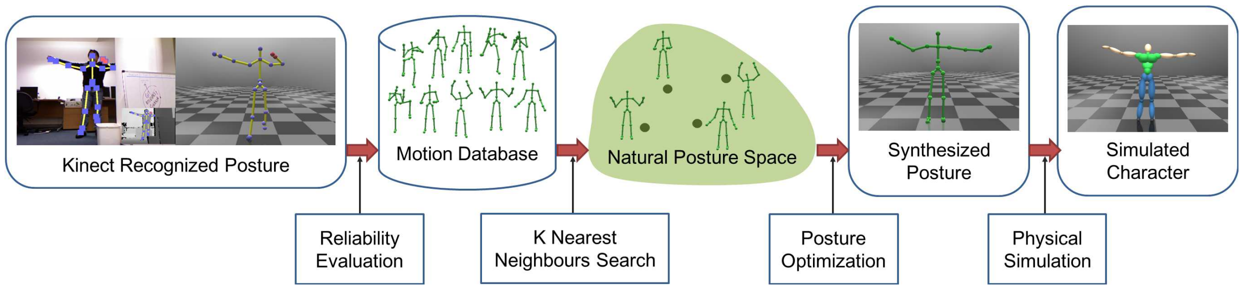


Fig. 1. The overview of the proposed framework to reconstruct noisy and incomplete Kinect posture.

IV. Reliability Measurement

An incorrectly tracked body part (i.e. false positive) in a motion recognition system is even more damaging than a missed part, because it would incorrectly guide the system to infer the posture. In this section, we explain our proposed reliability measurement to evaluate the reliability of a tracked body parts, such that we can identify false positives.

Although the actual implementation of the reliability terms may vary in different motion capturing systems, they can usually be classified into three groups: *the behaviours term*, *the kinematics term*, and *the tracking states term*. In the following, we explain our design of different terms that evaluates the posture returned by Kinect. We also discuss how the terms could be adjusted to fit into other popular motion capture systems.

A. Behaviours Reliability Term

The behaviours term refers to abnormal behaviour of a tracked part. In Kinect, this abnormality can be represented by the high frequency vibration of the body part positions, which happens when Kinect cannot track the part accurately. Here, we explain how we calculate the reliability value by evaluating the vibration.

Because Kinect acquires the position of a body part based on the depth pixels that are classified to it [14], when a part is partly/fully occluded, the position cannot be estimated accurately. This results in high frequency vibration of the tracked position. Similarly, when Kinect wrongly recognized an object as a body part, due to the lack of expected features on the object, the detected position becomes very unstable.

Assuming $p_i(f)$, $p_i(f+1)$ and $p_i(f+2)$ to be the 3D position of a tracked body part i in three successive frames, we can calculate the displacement vectors as:

$$d_i(f) = p_i(f+1) - p_i(f) \tag{1}$$
$$d_i(f+1) = p_i(f+2) - p_i(f+1) \tag{2}$$

The acute angle between the two vectors can be calculated by their dot product:

$$\theta_i(j) = \begin{cases} \frac{d_i(f)\cdot d_i(f+1)}{|d_i(f)||d_i(f+1)|} & \text{if } |d_i(f)| > d_{min} \\ & \text{and } |d_i(f+1)| > d_{min} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where d_{min} is the minimum length of an acceptable displacement vector, and is set to $3cm$ in our experiment. It is used to avoid getting a large angle change when the body part is almost steady.

The behaviour term, which is represented as the amount of vibration, is defined as:

$$Rb_i(f) = 1 - \frac{\max(\min(\frac{\sum_{f=0}^{f_b}\theta_i(f)}{f_b}, \theta_{roof}) - \theta_{floor}, 0)}{\theta_{roof} - \theta_{floor}} \tag{4}$$

where $Rb_i(f) \in [0.0, 1.0]$, f_b is the total number of frames we consider to detect vibration, θ_{floor} is an acceptable amount of rotation for each frame, θ_{roof} is the amount of rotation we consider to be the most unacceptable. Empirically, we found that setting $f_b = 3$, $\theta_{floor} = 90°$ and $\theta_{roof} = 135°$ gives a good result.

We observe that tradition marker based optical motion capture system, such as the *MotionAnalysis Eagle* system, also exhibits similar symptoms when a marker is mis-tracked. For example, the system sometimes mixes up close-by markers and results in sudden switches of marker positions. Similarly, an electro-magnetic motion capturer generates unstable marker positions when there is any external interference, such as the magnetic field generated by nearby electronic devices. Both situations can be detected by Equation 4.

B. Kinematics Reliability Term

The kinematics term refers to the kinematic correctness of the recognized posture. In Kinect, body parts are recognized independently without considering the kinematic relationship between parts. As a result, the segment length between two neighbouring parts is not consistent across frames, especially in the presence of incorrectly tracked parts. Here, we explain how we analyze the segment length to identify those parts.

We obtain a posture to gather the reference segment lengths. Kinect does not maintain pre-defined segment length for different body segments. Thus, the captured postures from users of different body sizes have different dimensions. The reference segment lengths is obtained/updated whenever (1) the user performed posture is similar to the predefined reference postures, and (2) all body parts are visible and tracked by Kinect. In our system, we defined two reference postures, which are a T-Pose (Figure 2 left) and a standard standing pose (Figure 2 right). These two postures are chosen because Kinect usually does not start tracking until either of them is performed. Thus, using such postures as references does not require any extra system initialization from the user.

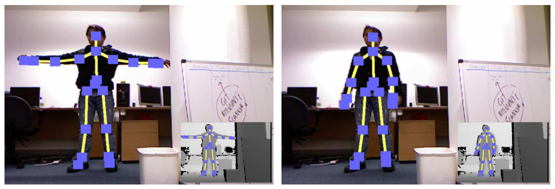


Fig. 2. The two reference postures for initializing segment lengths.

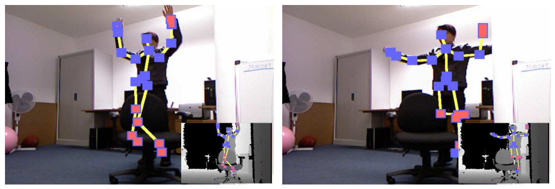


Fig. 3. Examples of the detected unreliable parts indicated as red square.

A body part can connect to multiple segments depending on the skeleton structure, such as the hips connecting to three segments. Assuming the body part i is connected to s_{part_total} body segments, for each connecting segment s, the segment difference ratio at frame f is calculated as:

$$d_s(f) = \min(\frac{|l_s(f) - l_{s\_ref}|}{l_{s\_ref}}, 1) \tag{5}$$

where l_{s_ref} is the reference segment length and $l_s(f)$ is the current segment length for segment s at frame f.

The kinematics reliability value of a body part is defined as the mean segment different ratio for all connecting segments:

$$Rk_i(f) = 1 - \frac{\sum_{s=0}^{s_{part\_total}} d_s(f)}{s_{part\_total}} \tag{6}$$

where $Rk_i(f) \in [0.0, 1.0]$.

Another possible implementation of the kinematics reliability term is to consider the segment orientation limit. Human joints have a limited rotational movement. With such limits defined for each segment, one can calculate the kinematics reliability value by evaluating how much the Kinect posture violates the segment orientation limit. However, we found that this implementation performs sub-optimally. This is because (1) it requires complex initialization to obtain the orientation limit of the user, and (2) the limit is usually very large, thus becoming ineffective to tell if a part is mis-tracked.

Our kinematics term can also be applied in traditional optical motion capture system. A simple implementation could be monitoring the distances between neighbouring markers in the same body segment, and observing any abnormal changes. For systems that conserve segment length such as the mechanical systems, the segment orientation limit is the best choice.

C. *Tracking State Reliability Term*

For most motion capture system, the hardware provides some indications on how well the posture is tracked. The tracking state term refers to the hardware feedback of whether a part is well tracked or not. Here, we explain how to apply tracking feedback from Kinect to evaluate the reliability value.

Kinect provides basic information to tell if a body part is tracked, inferred based on neighbouring parts, or not tracked when it is completely not visible. The tracking state term of part i at frame f is defined as:

$$Rt_i(f) = \begin{cases} 1.0 & \text{if tracked} \\ 0.0 & \text{if inferred} \\ 0.0 & \text{if not tracked} \end{cases} \tag{7}$$

where $Rt_i(f) \in [0.0, 1.0]$.

While it is possible to set a higher value for the inferred parts, we find that those parts usually are not very accurate. Comparing to a false positive, a false negative (i.e. wrongly classifying a correctly tracked joint as incorrect) has very little impact on the reconstruction system. Thus, we set the value to zero for the inferred parts.

The implementation of the tracking state term depends heavily on the hardware and the feedback it provides. For a traditional optical motion capture system, the tracking state can be defined as the number of cameras that can track a particular marker. For accelerometer based system such as Wiimote, once the applied force exceeds the sensor limit, the reading is cut off. The tracking state can be defined as how far the applied force is from the limit.

D. *Reliability Rate*

Here, we explain how we combine different terms to calculate the reliability rate of each body part.

The reliability rate of a body part i is defined as:

$$R_i = \sum_{f=0}^{f_{total}} w(f) \times \min(Rb_i(f), Rk_i(f), Rt_i(f)) \tag{8}$$

where $R_i \in [0.0, 1.0]$, f_{total} is the total number of frames to consider, $w(f)$ is a Gaussian weight for frame f with the largest value for the most recent frame, $Rb_i(f)$, $Rk_i(f)$, $Rt_i(f)$ are the reliability terms explained above. The use of reliability values in previous frames with a Gaussian weight provides a smoothing out effect. That is, when a body part is mis-tracked, it takes a few frames for the reliability rate to recover, and subsequently enhances the system stability. In our system running at $30Hz$, f_{total} is set to be 6, which means we consider a window of 0.2 second.

Figure 3 shows an example of applying Equation 8 on the body parts tracked by Kinect. To highlight the incorrectly tracked part, we render a red square on each part with the size proportional to $1 - R_i$. Notice that our system picks up most mis-tracked parts accurately, such as the left hand that is not accurately tracked in Figure 3 (Left) and wrongly tracked in Figure 3 (Right).

V. Natural Posture Space

In this section, we explain how we construct a natural posture space that is used to synthesize natural posture. This involves an offline process to prepare a motion database, as well as two online processes to extract posture samples and apply local Principle Component Analysis (PCA).

A. Motion Database

Here, we explain the offline process to prepare the motion database.

We create the motion database with motions captured from a traditional optical motion capture system. The captured motions are retargeted to the Kinect skeleton structure using commercial software, and the dimensions of body parts are designed according to [37]. We remove the global rotation along the vertical axis and the global 3D translation for normalization. Each posture is represented with a set of body parts positions.

The database is filtered to remove similar postures by thresholding the sum of squared differences of body parts positions. Apart from enhancing run-time efficiency, the major purpose of the process is to control the data sample density in posture space. If the density is unreasonably high, a database query may retrieve overly similar postures, and thus lack of the neccessary variation to construct a meaningful natural posture space as explained in Section V-B. We will analyze the effect of the filtering threshold on the quality of the synthesized motion in Section VIII-D

The motions that should be contained in the database depend on the target application. The idea is that we compose a database with the movements that the users are expected to perform. Our database includes motions of different classes such as boxing and walking. The unfiltered database contains roughly 21590 postures, which is then filtered into 2574. The database is relatively compact due to the scope of the target motion. However, if the application requires the user to perform a wide variety of motion, such as dancing in different style, a larger database will be needed. The implementation details of our database can be found in Section VIII.

Notice that while our framework can support body posture represented by orientations, we opt to use 3D positions. This is because Kinect tracking is based on position. The orientation estimation provided by Kinect is far less reliable than the position one. For parts with short segment length, orientation can easily flip over $180°$ with a small amount of positional error. We therefore synthesize posture using 3D positions, and apply [1] to reconstruct the orientation, which will be detailed in Section VII.

B. Local Principle Component Analysis

During run-time, we extract postures that are similar to the user performed one from the database and construct the natural posture space. This involves applying the reliability rate to obtain similar postures and reducing their dimensions with PCA.

Given a posture captured by Kinect, we evaluate the reliability rate with Equation 8 for each body part, and obtain the K nearest neighbours from the database. To compare the Kinect posture with those in the database, we first normalize it by removing the translation and vertical axis rotation of the root, such that they become view point invariant. We then retarget the Kinect posture to the standard body size used in the motion database [1].

Different from traditional K nearest neighbour search, we apply the reliability rate of a joint as a weight for finding similar postures. The posture difference function is defined as:

$$D(p^d, p^k) = \sum_{i=0}^{i_{total}} R_i (p_i^d - p_i^k)^2 \tag{9}$$

where p^d and p^k are the database posture and the Kinect posture respectively, i_{total} is the total number of body parts, p_i^d and p_i^k are the 3D body part position from the database posture and Kinect posture respectively. With the use of the reliability rate, we rely more on the correctly tracked parts, while reducing the influence of the unreliable ones.

Since the extracted K postures are similar to each other, we can represent them in a low dimensional space. We use PCA to construct the reduced space, and name it the natural posture space as it is constructed with real postures. The major advantage of creating a posture space instead of using individual posture directly is to allow synthesizing postures that are not available in the motion database. The space implicitly allows blending of postures, and thus can synthesize in a much wider range. In Section VIII-D, we will analyze the effect on changing the number of dimension used to represent the reduced space.

We applied brute force search for the K nearest neighbours because of the small size of the motion database. However, in case a large database is used, brute force search will be computationally costly. A possible solution is to quantize the reliability rate into a number of discrete values, and precompute a neighbour map for different combinations of reliability rates in the body parts. This will enhance the run-time efficiency, with the cost of extra memory usage for storing the pre-computed neighbour map.

VI. Posture Synthesis

In this section, we explain how we synthesize a posture that follows the tracked body parts while reconstructing missing or unreliable parts. The synthesizing framework is constructed as an optimization process that is driven by multiple energy terms. We will first detail different terms we designed, and then explain how we conduct the optimization process.

A. Control Term

Here, we explain the control term, which evaluates how well the synthesized posture fit with the Kinect posture.

Each sample point on the natural posture space, q^x, corresponds to a body posture in the full dimensional space, p^x. p^x can be calculated by back projecting q^x using the projection matrix calculated by PCA when constructing the natural posture space. To evaluate how well q^x fits with the Kinect posture, we consider p^x and apply the distance function mentioned in Equation 9:

$$Ec = D(p^x, p^k) \tag{10}$$

where p^k is the Kinect posture.

Notice that since the reliability rate is considered in Equation 9, we consider well tracked parts heavier than the unreliable ones. This prevents the synthesized posture being affected by wrongly tracked body parts.

B. Style Term

Here, we explain the style term, which evaluates how well the synthesized posture represents the posture style in the natural posture space. This term is particular important when the Kinect posture is noisy and incomplete, as it can help defining the missing parts.

The natural posture space encodes the style of posture intrinsically, defined by the K nearest neighbours that are used to construct the space. To ensure the synthesized posture p^x follows the style, the style term is defined as the distance between p^x and its closest neighbour:

$$Es = \min_n \frac{\sum_{i=0}^{i_{total}} (p_i^x - p_i^n)^2}{i_{total}} \tag{11}$$

where p^n represents one of the k neighbours used to construct the natural posture space, p_i^x and p_i^n represent body part i in the respective posture, and i_{total} is the total number of body parts.

While previous works such as [3] use probability density functions to represent the likelihood of the movement of a part in a covariance matrix, we argue that it may not be a good solution for our problem. This is because the samples used to construct the space do not come from a continuous motion. They are discrete neighbours that are close to the Kinect posture, in which some parts may not be available. Thus, the neighbours may not lay in a consistent probability distribution. Our method discretely considers each neighbour and minimizes the distance towards the closest neighbour. This ensures that the synthesized posture exhibits similar style.

C. Kinematics Term

Here, we explain the kinematics term, which maintains the kinematics features of the synthesized posture.

Similar to Section IV-B, we represent the kinematics requirements as the segment length. The only difference is that because the synthesized posture is from the natural posture space, its segment sizes should be similar to the standard body size used in the database. The kinematics term is thus defined as:

$$Ek = \frac{\sum_{s=0}^{s_{total}} (l_s^x - l_s^{database})^2}{s_{total}} \tag{12}$$

where l_s^x is the length of segment s in the synthesized pose, $l_s^{database}$ is the reference segment length of the standard body size used in the database [37], s_{total} is the total number of segments of the character.

It is also possible to define a segment orientation term to make sure the synthesized posture obeys the orientation limits. However, we found it unnecessary, because the use of the style term defined in Section VI-B achieves a similar effect to prevent violation of orientation limits. The segment length term we implemented, however, is a far stricter requirement and has to be enforced specifically.

D. Movement Continuity Term

Here, we explain the movement continuity term, which takes into account the synthesized postures in the previous frames to maintain smooth movement.

The movement continuity term minimizes the change of displacement vector of every body part in the synthesized postures. It is defined as:

$$\begin{aligned} Em &= \frac{\sum_{i=0}^{i_{total}} ((p_i^x - p_i^{x-1}) - (p_i^{x-1} - p_i^{x-2}))^2}{i_{total}} && (13) \\ &= \frac{\sum_{i=0}^{i_{total}} (p_i^x - 2p_i^{x-1} + p_i^{x-2})^2}{i_{total}} && (14) \end{aligned}$$

where p_i^x, p_i^{x-1}, p_i^{x-2} represent the position of body part i for the current, one frame before current, and two frames before current synthesized postures respectively, i_{total} is the total number of body parts.

E. Optimization

Here, we explain how we synthesize the final posture by optimization based on the terms explained above.

We applied a customized version of local stochastic search algorithm [38], which is a variation of the random sampling method [39], to optimize for the target posture. Given an initial sample on the natural posture space, we randomly sample a number of potential postures in the space for each iteration. The optimization score of each sample is evaluated as a weighted sum of the energy terms:

$$E = w_c Ec + w_s Es + w_k Ek + w_m Em \tag{15}$$

where w_c, w_s, w_k, and w_m are the weights. In our system, they are set as 1.0, 0.5, 1.5 and 0.25 respectively. The potential posture that minimizes the evaluation function will be considered as the initial posture sample of the next iteration. The optimization process continues until an optimal solution is found, or the number of iterations reaches a predefined limit. This allows us to control the trade-off between synthesis quality and computation time. We will provide detailed analysis on how these values affect the system performance in Section VIII-D.

There are some general principles for tuning the weight in Equation 15. First, the kinematics terms should not be violated as most character control systems in games and animation applications conserve the segment lengths of the characters. Thus, its weight is the highest. Second, the primary purpose of the system is to reconstruct the Kinect posture, with a secondary purpose to maintain a realistic style. This explains why the weight of the former is larger than that of the latter. Third, the movement continuity term is a trade-off between temporal smoothness and system responsiveness. For interactive applications, we should use the smallest possible value to minimize the lag introduced to the tracking system, as the term minimizes velocity change across frames.

Since our posture synthesis algorithm is a frame-based approach as oppose to the spacetime optimization method [40], it may suffer from posture inconsistency across frames. While it is possible to dramatically increase the weight of the movement continuity term w_{Em} to enhance the smoothness of the synthesized motion, it could over-constrain the optimization process and reduce system responsiveness. To remedy this, we use the previously synthesized posture as the initial sample posture to start the optimization. As a result, the optimization process always has to opportunity to explore postures that are similar to the previous one. This approach, however, cannot completely eliminate the high frequency vibration of the synthesized movement. We utilize a physical simulation system and a PD controller to track the reconstructed postures, which ensures a natural and kinematic valid final result. More details are given in Section VII.

While previous research suggest that the Covariance Matrix Adaptation (CMA) method works well in optimizing control parameters for character movement in the full dimensional space [41], [42], it is unnecessary in our problem. This is because PCA has already minimized the intrinsic redundancy within the parameters in the latent space, and hence CMA cannot produce significant improvements. On the other hand, as discussed in previous research [43], simple gradient descent usually does not work well with human motion optimization. This is because of the non-linear features of the motion evaluation process, as well as the presence of local optimal. A discrete sampling approach is more reliable.

VII. Physical Simulation

Based on the synthesized posture in the previous section, we adopt the physical modelling method in [1] to simulate natural and kinematically valid motions. In this section, we review the method and highlight the changes we made in this paper.

The physically simulated character is used as a kinematic filter to create natural movements. As in [1], we apply external forces and torques to drive the character to the synthesized posture. It has two major advantages comparing to simply displaying the synthesized posture. First, because of the use of a PD controller, the movement of the body parts obeys Newton physics. Second, because we construct the character in a physical world, kinematics features such as segment lengths are accurately maintained.

A. Physical World Modelling

Here, we explain how we model the physical world.

The physical environment is modelled with the Open Dynamics Engine [44]. We create an infinity large plane in the ODE world as the floor plane, which provides supporting force to the simulated characters. Gravity is implemented such that when no control force is applied, the characters fall onto the ground naturally.

Each character is represented by 19 body segments and 20 joints according to the Kinect skeleton definition. The size and the mass of each segment are set according to [37]. Segments are modelled with capsules for efficient collision detection, and the joints are modelled with ball joints, which indicate that each segment has 3 degrees of freedom in rotation.

B. Target Posture

Here, we explain the difference in constructing the target posture between our method and [1].

In this paper, we utilize the body part positions synthesized in the natural postures space as the target posture. The orientation of the joint is obtained by calculating the weighted sum of orientation from the K nearest neighbour used to construct the natural posture space. The weight is defined as the inverse of different between the synthesized posture and the neighbour using Equation 9. Comparing to [1], which defines the target posture as the posture that is most similar to the Kinect one in the motion database, our method create much better results, especially when the Kinect posture is being occluded.

C. PD Controller

Using the synthesized posture as the target posture, we calculate the control force and torque for each body part, and drive the character to fit into the target. Similar to [1], we control the movement with 3 dimensional forces and the 1 dimensional torque along the body segment direction. This facilitates easier control parameter tuning and more efficient simulation.

In each time step, the control force for a part i is calculated by a PD controller:

$$F_i = K_e(p_i^{target} - p_i^{current}) + K_d(p_i^{target'} - p_i^{current'}) \quad (16)$$

where p_i^{target} is the target position of the part, $p_i^{current}$ is the current position of the part, $p_i^{target'}$ and $p_i^{current'}$ are the respective derivative, K_e is the elasticity gain and K_d is the damping gain. A high K_e can improve the responsiveness of the character, while a high K_d produce more stable movements. We manually tune the smallest possible K_e and K_d, as a system with high control forces is usually unstable, and use same values for all joints. Furthermore, the magnitude of the resultant force F is bounded by a predefined value to avoid unexpected high control force while the target positions are very different from the current ones.

The control torque along the axis of body part i is calculated similarly:

$$T_i = K_i^{\epsilon}(\theta_i^{target} - \theta_i^{current}) + K_i^{\delta}(\theta_i^{target'} - \theta_i^{current'}) \quad (17)$$

where θ_i^{target} is the target rotation of the joint along the joint axis, $\theta_i^{current}$ is the current rotation, $\theta_i^{target'}$ and $\Theta_i^{current'}$ are the respective derivative, K_i^{ϵ} and K_i^{δ} are the hand tuned elasticity gain and damping gain. Similar to the force calculation, the torque T is bounded by a predefined value.

During simulation, the physical simulation engine ODE maintains the segment length and segment connectivity while applying the calculated control forces and torques. The resultant posture is the equilibrium state of the character, representing the posture that can satisfy the target posture the most.

VIII. Experimental Results

In this section, we show experimental results obtained from our system. We first show results of postures reconstruction

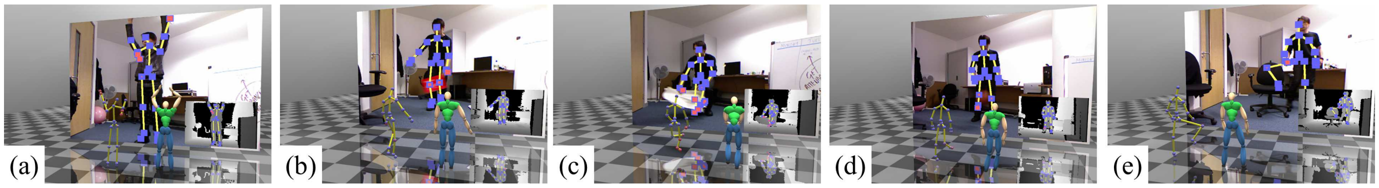


Fig. 4. Posture reconstruction using our system when the user (a) performs different postures, (b) interacts with a box, (c) interacts with a piece of paper, (d) plays a basketball, and (3) sits on a chair.

Motions	Examples	Frames	Upper SD	Lower SD
Locomotion	Run/Walk	3573	8.1	15.27
Object Handling	Pick/Carry	1521	8.9	11.80
Upper Body Movement	Point/Scratch	6607	17.8	6.6
Boxing	Punch/Kick	7769	11.8	13.7
Sword Fighting	Swing/Dodge	2120	16.3	13.4

TABLE I
DETAILS OF THE MOTION DATABASE IN OUR SYSTEM.

Kinect Dataset	Frames	Upper SD	Lower SD
Basic Tracking (Figure 4a)	948	13.4	9.24
Box (Figure 4b)	897	12.26	6.39
Flip Chart (Figure 4c)	1098	13.24	7.38
Basketball (Figure 4d)	831	11.81	4.18
Chair (Figure 4e)	748	7.87	11.86
Office (Figure 7)	685	13.23	9.51

TABLE II
DETAILS OF THE KINECT TESTING DATA IN OUR SYSTEM.

when the user interacts with external objects. Next, we compare our method with previously proposed algorithms. Finally, we analyze the accuracy and performance of our system. The readers are referred to the supplementary video for further details.

The experiments were conducted with a laptop computer with an Intel Core i7-2630 CPU (2.00GHz), 8GB RAM and a GeForce GTX 560M graphic card. The system achieved real-time performance and was capable of handling over 30 frames per second (fps), although the actual frame rate was limited by the Kinect hardware to 30 fps.

We use motions captured from traditional mocap to construct our motion database, and those captured from Kinect as our testing data. Our database consists of motions from different classes. Table I shows the details including example motions, number of frames, standard deviation of positional movement in centimetre for the upper and lower body. The overall average standard deviation in the whole database is 13.25cm for the upper body, and 11.63cm for the lower body.

The testing data from Kinect includes postures when the user interacts with different objects. The details are shown in Table II. The overall average standard deviation is 12.04cm for the upper body and 7.98cm for the lower body. These values indicate that the upper body movement of the Kinect captures is comparable to that of the database, while the lower body movement is slightly constrained due to the relatively small capturing area.

A. Posture Reconstruction

Here, we discuss the experiments for reconstructing the user postures in different situations.

As shown in Figure 4, we rendered the Kinect skeleton with purple body parts and yellow body segments. We used red color to indicate body parts with low reliability value. The reconstructed posture was rendered with the character in green clothes. The two postures were placed side by side for easier comparison.

We first tested how well our system reconstruct basic user postures such as raising arms and legs (Figure 4a). Then, we asked the user to interact with different objects, including a box (Figure 4b), a flip chart (Figure 4b), a basketball (Figure 4c) and a chair (Figure 4e). During the interactions, Kinect incorrectly considered the objects as part of the body. Our reliability measurement successfully detected those unreliable parts and reconstructs the posture accordingly.

B. Perceptual Comparison

Here, we compare our proposed algorithm with other motion reconstruction approaches, and assess the perceptual accuracy of each method using a survey-based evaluation [45].

We define method A as the approach proposed in [1]. Method B is an implementation of our proposed method with the reliability evaluation functions disabled, which is comparable to previous works that optimize the posture with all tracked parts without any reliability assessment such as [2] and [3]. Our proposed algorithm is named as method C.

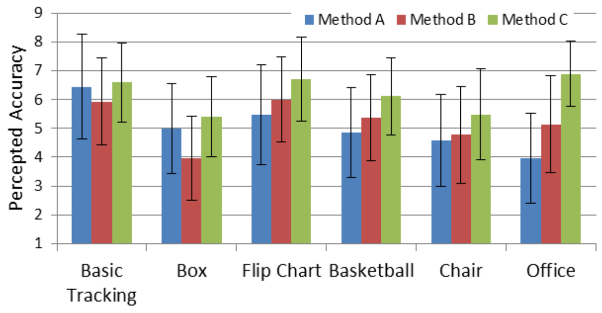


Fig. 6. The perceptual accuracy according to user studies for the data sets using different reconstruction methods.

We conducted an evaluation experiment with 27 participants. The objective of the experiment was to evaluate the relative perceptual accuracy among the three methods. During the experiment, the participants were presented with the Kinect color video, as well as the synthesized characters using method A, B, and C one after another. The participants were not told which method to be our algorithm. They graded the accuracy by comparing the movement of the character and that of the

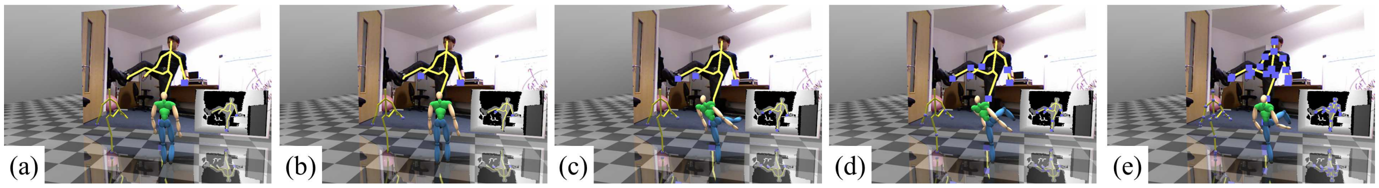


Fig. 5. Reconstruction using (a) the left hand and foot, (b) both hands, (c) hands and feet, (d) hands, feet, elbows and knees, and (e) all body parts.

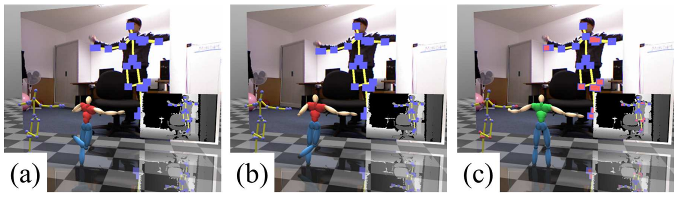


Fig. 7. The reconstructed postures using (a) method proposed in [1], (b) our method without the reliability values, (c) our method.

Setup	Parts Used	Difference (cm)
(a)	Left Hand, Left Foot	6.38
(b)	Hands	5.60
(c)	Hands, Feet	4.52
(d)	Hands, Feet, Elbows, Knees	4.15
(e)	All Parts	3.90

TABLE III
ACCURACY OF THE POSTURE RECONSTRUCTION FRAMEWORK.

Class	Duration (sec)	Diff. Raw (cm)	Diff. Reconstructed (cm)	Percentage Enhanced
(a)	53.5	9.59	7.26	24.3%
(b)	52.8	10.89	7.44	31.6%
(c)	60.4	12.41	7.76	37.5%

TABLE IV
ACCURACY OF THE OVERALL SYSTEM.

Kinect user in a 9 point scale, with 1 being inaccurate and 9 being accurate.

Figure 6 shows the average accuracy rating from participants for each testing data set, in which the vertical lines represent standard derivation. Our proposed algorithm consistently outperforms the other two. Considering all data sets, the overall average perceptual accuracy of method A, B and C are 5.06, 5.20 and 6.20 respectively, with the standard derivation of 1.63, 1.55 and 1.38.

We observe that for movement with less occlusion, such as the basic tracking and the box data sets, our method does not perform significantly better than method A. This is because the major source of inaccuracy in simple environments is the signal noise, and method A is good at filtering vibration in body parts. However, in the office data set where serious occlusion occurs, our method performs far better. Method A does not construct a natural posture space for posture optimization. The reconstructed postures are notably different from the actual user movement (Figure 7a). Method B cannot fully estimate the true posture of the user, as it is influenced by the incorrectly tracked parts (Figure 7b). Method C accurately identifies parts with low reliability value and corrects them accordingly (Figure 7c).

C. Numerical Accuracy Analysis

Here, we analyze the numerical accuracy of the posture reconstruction framework, as well as the overall accuracy including the Kinect hardware error.

We define an error function between two postures by considering the body parts position relative to their parents in the skeleton hierarchy, as such a representation has been shown effective in [46]:

$$e = \frac{\sum_{f=0}^{f_{total}} \sum_{i=0}^{i_{total}} \left| \left(p_i^1(f) - p_{i'}^1(f)\right) - \left(p_i^2(f) - p_{i'}^2(f)\right) \right|}{f_{total} \times i_{total}} \quad (18)$$

where $p_i^1(f)$ and $p_i^2(f)$ are positions of part i at frame f of posture 1 and 2 respectively, i' denotes the parent part, i_{total} is the total number of body parts, f_{total} is the total number of frames considered.

The first analysis measured the accuracy of our posture reconstruction framework. We carefully recorded 20 seconds of data from Kinect, trying our best to minimize Kinect tracking error. Next, we reconstructed the postures using different subsets of parts, which are shown as purple body parts in Figure 5. We calculated the average error between the reconstructed posture and the Kinect raw posture over the whole capture using Equation 18. The results are shown in Table III. We found that even with only a few body parts such as setup (c), our system could reconstruct the full body postures while maintaining a small error. However, setup (a) and (b) sometimes failed to reconstruct the movement, especially when the moving parts were not considered.

The second analysis evaluated the accuracy of the overall system, considering the Kinect hardware as part of the system component. We captured user movements with Kinect and the Polhemus Liberty magnetic mocap at the same time. The magnetic mocap was a wired system with 16 6D sensors. We did not use an optical mocap because its infrared emitters interfered with the Kinect one, severely degrading the quality of both systems. We classified the captured movements into three classes: (a) self-occluded and un-occluded movements, (b) movements being occluded by environment objects, (c) movements interacting with, and being occluded by, environment objects. Using Equation 18, we compared the Kinect raw postures and the reconstructed postures with those obtained from the magnetic mocap. The results are shown in Table IV. As expected, the error of the raw postures was large in general, and the value was even larger for more complex movements such as those in class (c). For all classes of movement, our system can correct the posture to a consistent quality.

Notice that due to the hardware implementation, the body

Parameter	Value
Number of Posture in Motion Database	21590
Number of Posture in Filtered Motion Database	2574
Number of Neighbour in K Nearest Neighbour Search	30
Dimensionality of the Natural Posture Space	20
Maximum Number of Optimization Step	20
Number of Sample Per Optimization Step	100

TABLE V
PARAMETER USED IN OUR SYSTEM.

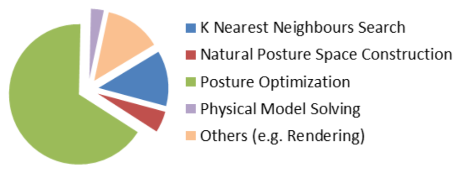


Fig. 9. The proportion of computational cost for different processes.

part positions detected by Kinect were never accurate, especially in the depth dimension. This explains the relatively high error with respect to the magnetic mocap. Our system obtained input from Kinect and inherited the error from Kinect when reconstructing the postures. Still, our system managed to bring the postures towards to true ones.

D. Performance Analysis

Here, we analyze the effects of various parameters considering the reconstruction quality and computational time. We also evaluate the computational cost of the overall system.

We first recorded 30 seconds of data from Kinect. Using the recorded data, we tuned the parameters of the system and evaluated the reconstructed postures (Figure 8). The average optimization score (blue line) and the average frame time (red line) were plotted. Notice that the optimization score in the plots was in negative values.

Figure 8a shows that if the database is heavily filtered, the reconstruction performance is poor, due to the lack of relevant posture that can be used to construct the natural posture space. With a database of reasonable size, which is roughly 2000 postures, we can obtain much better results. The optimization score does not improve significantly when the database size increases further, because the postures become redundant and do not contribute to the reconstruction process. Figure 8b shows that using excessive number of neighbours to construct the natural posture space has a negative effect on the optimization score. This is because the system may not be able to find so many neighbours that are similar to the current Kinect posture. Figure 8c shows that given the same amount of computational power, using a large number of dimensions to represent the space results in under-sampling, and thus the final results are degraded. Figure 8d shows that the optimization score converges after around 20 optimization steps. The frame time does not increase further beyond that point because we terminate the optimization process once the optimal solution is found. Figure 8e shows that the performance of the optimization becomes consistent when the number of sample per optimization steps is larger than 100.

Because of the intrinsic dependency among the parameters, tuning the parameters requires multiple iterations. In each iteration, they are tuned one by one for the best value. This process is repeated until the system cannot be improved further. Table V summarizes the values we used in our system.

Our tuned system runs in 27.7 ms per frame (36.7 frame per second), which has a higher frame rate than real-time (30 frame per second). The proportions of computational cost of different processes are shown in Figure 9. Optimizing the posture is the most computational costly, accounting for roughly 66% of the total processing time, while other processes are relatively fast.

IX. CONCLUSION AND DISCUSSIONS

In this paper, we propose a new framework to automatically reconstruct full body motion from corrupted motion capture data. Our method evaluates the reliability of every tracked body parts and creates a natural posture space to synthesize a valid posture. We demonstrate the effectiveness of our approach by reconstructing motion from incomplete and noisy motion data acquired from the Microsoft Kinect. Experimental results show that our method can successfully correct the input motion captured from extreme conditions when a large portion of the body is occluded. Our framework is computationally efficient and achieves real-time performance, making it applicable to a wide variety of interactive applications such as motion gaming and sport training.

Our system can potentially enhance the user experience in multi-player applications, because we can reconstruct the postures when one player is occluded by another. This enables dense user-user interactions in applications like dancing games. The practical problem that has to be solved is the computational cost, because the reconstruction time is linearly proportional to the number of user in the scene. GPU based optimization can be a potential solution.

We assume that the database contains postures that are similar to the user performed ones. In case such postures are not available, the natural posture space constructed may not accurately estimate the correct posture of the user, and the quality of the reconstructed motion may drop. This is a general problem of data driven algorithms. One possible solution is to insert correctly tracked Kinect postures into the motion database during run-time, which enables the system to learn and adapt to unexpected movements.

Fast movement reconstruction with Kinect is a challenging problem. The depth camera of Kinect suffers from motion blur when the user performs fast movement such as punching. Figure 10 (Left) demonstrates the blurring effect of a waving hand in comparison of a steady one. While our system can reconstruct those postures, the decrease in Kinect tracking accuracy does impact the resultant quality. Furthermore, because of the 30 frame per second capturing limit, per frame velocity of fast moving body parts is relatively large and contains more artifacts. One may consider tuning the weight of the movement continuity term to enhance system responsiveness.

Another challenging situation is user rotation. Our system can handle roughly 45 degree of rotation, in which Kinect usually manages to recognize a few body parts in the shadowed side. If one side of the body is completely lost, the system can

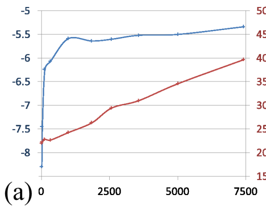
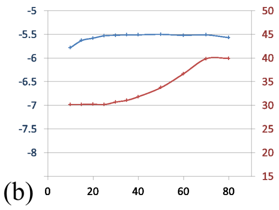
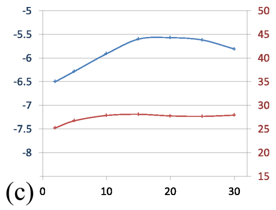
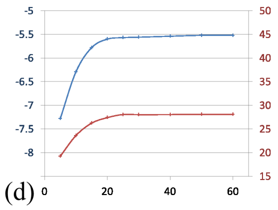
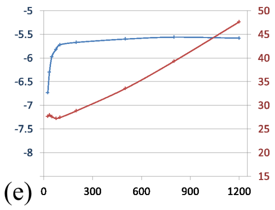
(a)  (b)  (c)  (d)  (e) 

Fig. 8. The optimization score (left axis, blue line) and frame time in ms (right axis, red line) for different number of (a) filtered postures in the motion database, (b) neighbours in the KNN search, (c) dimensions in the natural posture space, (d) optimization steps, and (e) samples per optimization step.

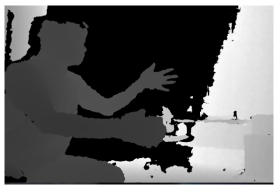
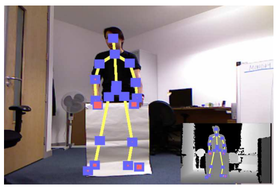
 

Fig. 10. (Left) Motion blur problem of fast movements. (Right) An occasion when the reliability measurement fails.

only predict based on the visible half. The accuracy depends on the movement correlation of the two sides. Practically, this problem can be solved with domain knowledge. In many applications such as console games, the class of user performed motions is known in advance. The system can reconstruct the postures using a database containing only that class of action, instead of the general purpose database we used. This helps to determine postures with few recongnized parts.

Because our measurement considers high level behaviours, in the rare occasions when an external object moves similarly as a body part, it may fail to detect the error. Figure 10 (Right) shows a carefully constructed situation where Kinect regards the flip chart as the lower body of the user. Our measurement cannot detect all incorrect body parts, as they exhibits similar movement features as the legs. One future direction is to assess low level details to assist reliability measurement.

As explained throughout the paper, our framework is general and can be applied in different motion capture systems. Another future direction is to apply it for enhancing the traditional optical motion capture process, where missing or incorrectly tracked markers have to be clean up manually.

Acknowledgment

We thank the anonymous reviewers for their constructive suggestions. This project is partially supported by the Academic Staff Support Grant (Northumbria University) 100700-IO114845, the Science Faculty Research Grant (Hong Kong Baptist University) FRG1/12-13/055 and FRG2/12-13/078.

References

[1] H. Shum and E. S. Ho, "Real-time physical modelling of character movements with microsoft kinect," in *Proceedings of the 18th ACM symposium on Virtual reality software and technology*, ser. VRST '12. New York, NY, USA: ACM, 2012, pp. 17–24. [Online]. Available: http://doi.acm.org/10.1145/2407336.2407340

[2] H. P. H. Shum, T. Komura, and S. Takagi, "Fast accelerometer-based motion recognition with a dual buffer framework," *The International Journal of Virtual Reality*, vol. 10, no. 3, pp. 17–24, September 2011.

[3] J. Chai and J. K. Hodgins, "Performance animation from low-dimensional control signals," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*. New York, NY, USA: ACM, 2005, pp. 686–696.

[4] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, "Robust hand gesture recognition with kinect sensor," in *Proceedings of the 19th ACM international conference on Multimedia*, ser. MM '11. New York, NY, USA: ACM, 2011, pp. 759–760.

[5] S. Kean, J. Hall, and P. Perry, *Meet the Kinect: An Introduction to Programming Natural User Interfaces*, 1st ed. Berkely, CA, USA: Apress, 2011.

[6] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ser. UIST '11. New York, NY, USA: ACM, 2011, pp. 559–568.

[7] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, oct. 2011, pp. 127 –136.

[8] M. Sun, P. Kohli, and J. Shotton, "Conditional regression forests for human pose estimation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 3394–3401.

[9] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 415–422.

[10] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, "Efficient human pose estimation from single depth images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[11] A. Baak, M. Muller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1092–1099. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126356

[12] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *Cybernetics, IEEE Transactions on*, 2013.

[13] Microsoft Corporation, "Kinect for windows SDK programming guide version 1.5," 2012.

[14] 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," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1297–1304.

[15] M. Raptis, D. Kirovski, and H. Hoppe, "Real-time classification of dance gestures from skeleton animation," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '11. New York, NY, USA: ACM, 2011, pp. 147–156.

[16] S. W. Bailey and B. Bodenheimer, "A comparison of motion capture data recorded from a vicon system and a microsoft kinect sensor," in *Proceedings of the ACM Symposium on Applied Perception*, ser. SAP '12. New York, NY, USA: ACM, 2012, pp. 121–121.

[17] A. Fern'ndez-Baena, A. Susin, and X. Lligadas, "Biomechanical validation of upper-body and lower-body joint movements of kinect motion capture data for rehabilitation treatments," in *Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on*, sept. 2012, pp. 656 –661.

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

[19] W. Shen, K. Deng, X. Bai, T. Leyvand, B. Guo, and Z. Tu, "Exemplar-based human action pose correction and tagging," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 1784–1791.

[20] H. Liu, X. Wei, J. Chai, I. Ha, and T. Rhee, "Realtime human motion control with a small number of inertial sensors," in *Symposium on Interactive 3D Graphics and Games*, ser. I3D '11. New York, NY, USA: ACM, 2011, pp. 133–140.

[21] J. Tautges, A. Zinke, B. Krüger, J. Baumann, A. Weber, T. Helten, M. Müller, H.-P. Seidel, and B. Eberhardt, "Motion reconstruction using sparse accelerometer data," *ACM Trans. Graph.*, vol. 30, no. 3, pp. 18:1–18:12, May 2011.

[22] D. W. Aha, "Editorial," *Artif. Intell. Rev.*, vol. 11, no. 1-5, pp. 7–10, Feb. 1997.

[23] C. M. Bishop, *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press, 1996.

[24] X. K. Wei and J. Chai, "Intuitive interactive human-character posing with millions of example poses," *IEEE Computer Graphics and Applications*, vol. 31, pp. 78–88, 2011.

[25] Z. Ghahramani and G. E. Hinton, "The EM algorithm for mixtures of factor analyzers," University of Toronto, Technical Report CRG-TR-96-1, 1997.

[26] M. Lau, J. Chai, Y.-Q. Xu, and H.-Y. Shum, "Face poser: interactive modeling of 3d facial expressions using model priors," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ser. SCA '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 161–170.

[27] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," in *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*. New York, NY, USA: ACM, 2004, pp. 522–531.

[28] N. D. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," in *NIPS*, 2003.

[29] X. Wu, M. Tournier, and L. Reveret, "Natural character posing from a large motion database," *Computer Graphics and Applications, IEEE*, vol. 31, no. 3, pp. 69 –77, may-june 2011.

[30] J. Quiñonero Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *J. Mach. Learn. Res.*, vol. 6, pp. 1939–1959, Dec. 2005.

[31] Q. nonero Candela, C. E. Ramussen, and C. K. I. Williams, "Approximation methods for gaussian process regression," *Large-Scale Kernel Machines*, pp. 203–223, 2007.

[32] H. Lou and J. Chai, "Example-based human motion denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 5, pp. 870–879, Sep. 2010.

[33] H. Yue and M. Tomoyasu, "Weighted principal component analysis and its applications to improve fdc performance," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4, dec. 2004, pp. 4262 – 4267 Vol.4.

[34] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "A general framework for increasing the robustness of pca-based correlation clustering algorithms," in *Proceedings of the 20th international conference on Scientific and Statistical Database Management*, ser. SSDBM '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 418–435.

[35] J. F. Pinto da Costa, H. Alonso, and L. Roque, "A weighted principal component analysis and its application to gene expression data," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 8, no. 1, pp. 246–252, Jan. 2011.

[36] K. Forbes and E. Fiume, "An efficient search algorithm for motion data using weighted pca," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ser. SCA '05. New York, NY, USA: ACM, 2005, pp. 67–76.

[37] H. G. Armstrong, "Anthropometry and mass distribution for human analogues. volume 1. military male aviators," 1988.

[38] J. C. Spall, *Introduction to Stochastic Search and Optimization*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[39] F. J. Solis and R. J.-B. Wets, "Minimization by random search techniques," *Math. Oper. Res.*, vol. 6, no. 1, pp. 19–30, 1981.

[40] A. Witkin and M. Kass, "Spacetime constraints," in *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1988, pp. 159–168.

[41] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Optimizing walking controllers," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 168:1–168:8, Dec. 2009.

[42] H. P. H. Shum, T. Komura, T. Shiratori, and S. Takagi, "Physically-based character control in low dimensional space," in *Proceedings of the Third international conference on Motion in games*, ser. MIG'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 23–34.

[43] K. Yin, S. Coros, P. Beaudoin, and M. van de Panne, "Continuation methods for adapting simulated skills," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 81:1–81:7, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360680

[44] R. Smith, "Open dynamics engine," 2008, http://www.ode.org/. [Online]. Available: http://www.ode.org/

[45] L. Hoyet, R. McDonnell, and C. O'Sullivan, "Push it real: perceiving causality in virtual interactions," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 90:1–90:9, Jul. 2012.

[46] B. Yao and L. Fei-Fei, "Action recognition with exemplar based 2.5d graph matching," in *Proceedings of the 12th European conference on Computer Vision - Volume Part IV*, ser. ECCV'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 173–186.

Hubert P. H. Shum is a Senior Lecturer (Assistant Professor) in the Northumbria University. Before joining the university, he worked as a Lecturer in the University of Worcester, a post-doctoral researcher in RIKEN Japan, as well as a research assistant in the City University of Hong Kong. He received his PhD degree from the University of Edinburgh. His research interests include character animation, machine learning and human computer interaction.

Edmond S. L. Ho received the BSc (2003), MPhil (2006) and PhD (2011) degrees from the Hong Kong Baptist University, City University of Hong Kong and University of Edinburgh respectively. He is currently a Research Assistant Professor in the Department of Computer Science, Hong Kong Baptist University. His research interests include character animation, robotics, and human activity understanding.

Yang Jiang is a Lecturer (Assistant Professor) of Creative Media Technology in the Northumbria University. Her current research is focused on developing intelligent digital media processing algorithms, via exploring the boundaries across computing, media, digital entertainment, artificial intelligence, and image processing.

Shu Takagi is a Professor in the University of Tokyo. He received his Doctor of Engineering from the University of Tokyo in 1995. Since then, he had worked as a research associate, a lecturer, as well as an associate professor in the same university. His research interests include fluid mechanics, computational biomechanics and medical ultrasound.