國立交通大學

資訊科學與工程研究所

碩 士 論 文

利用少量慣性感測器監控人物動作之研究

Action Surveillance Using Sparse Wearable Inertial Sensors

研 究 生：林世祐

指導教授：林奕成

中 華 民 國 一〇一 年 九 月

利用少量慣性感測器監控人物動作之研究

Action Surveillance Using Sparse Wearable Inertial Sensors

研　究　生：	林世祐	Student：	Shih-Yu Lin
指導教授：	林奕成	Advisor：	I-Chen Lin

國　立　交　通　大　學

資　訊　科　學　與　工　程　研　究　所

碩　士　論　文

A Thesis

Submitted to Institutes of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2012

Hsinchu, Taiwan, Republic of China

中華民國一〇一年九月

利用少量慣性感測器監控人物動作之研究

學生：林世祐　　　　　　　　　　　　　指導教授：林奕成

國立交通大學資訊科學與工程研究所

摘　　　要

利用從感測器取得的資料來重建動作已經是一個相當普遍的研究技術。在這篇論文當中，我們呈現一個流程圖是藉由綁在使用者四肢跟軀幹上的四到五隻慣性感測器來重建整個人體動作。基於這些收集的資料，我們建立一個包含著十多萬幀線上 k 元樹的架構並從中取得最適當的動作片斷來決定目前使用者的全身動作。然而由於少量且有雜訊的感測資料通常會造成我們判斷動作的誤差，因此有著發生動作之間不連續的可能性。為了防止這項限制，我們將運動領域概念來避免發生此情況，並能達到更為合體的動作轉換，我們利用即時的動作合成機制將多個動作候補依據其比重關係來混和，使其能夠重現出更為自然且平順的動作，我們的主要目的是利用少量的慣性控制器來達到運用高昂儀器所作出的準確度，並且有著不受環境影響和自我屏蔽的限制。

關鍵字：動作重建、感測器、 k 元樹、Wii 控制器

Action Surveillance Using Sparse Wearable Inertial Sensors

Student：Shih-Yu Lin　　　　　　Advisor：Dr. I-Chen Lin

Institute of Computer Science and Engineering
National Chiao Tung University

Abstract

Motion reconstruction from sensor data is a notable research field. In this thesis, we present a framework to reconstruct full-body human motion by four to five inertial sensors that attached to the user's four limbs and torso. Based on the gathered data, we construct an online *k*-dimensional tree (kd-tree) index structure which consists of hundred thousands of frames, and find the most appropriate motion fragment as user's current full-body motion. However, the sparse and noisy sensing data cause high ambiguity for our motion estimation. It then results in gaps between poses continuous. Consequently, we include the concept of *motion fields* for more reasonable motion transition. This run-time motion synthesis mechanism merges the candidates of the motion sequences by weighted combination, and generates natural and smooth motions.

Keyword: Motion reconstruction　、Sensors、kd-tree、Wii remotes

Acknowledgement

感謝在這兩年來幫助我的指導教授，林奕成教授。不管是在學習上或是研究上都能給予我正確的指導，讓我在遇到挫折時都能克服萬難並有著面對未來挑戰的自信與技術，我相信在成長的路途中，一定是我的研究所路途中最大的收穫。

我也要感謝陪伴我的同學。在研究實驗上幫助我不少，使我能在研究上有著極大的助力，且在學習途中有著互相切磋的砥礪，而有著在漫長的求學不畏艱難的力量。

最後我要感謝我的家人，不管我做甚麼決定，都能在背後支持著我，讓我無後顧之憂，成為我最重要的精神支柱。為了表達內心至高無上的謝意，畢業後將時時期許自己，以專業知能回饋於社會，而做為一個對社會和國家具有貢獻的交大人。

Content

List of Figures

List of Table

Symbol Description

n : Frame number of one motion clip

t : time

α^t : the current frame of 3D accelerations at time t

K : the number of nearest neighborhood

M : the number of last sensing data

S : the storage of the nearest neighborhood

I : the connect path

C : the cost of connect path

Q : the joint angle

X : the joint position

V : the velocity of pose

A : the acceleration of pose

q : human pose

E : the energy minimization term

ω : the weight of energy term

Chapter 1.

Introduction

The whole world is gradually moving towards an aging society now. Therefore, health care becomes an essential and unavoidable topic. However, to care for the elders needs to spend a lot of time and labor. For example, to apply for a dedicated nursing work to care elders, etc. To alleviate the burden, we propose an approach to remotely survey user's motion from sensing device for caring elders or other monitoring issue.

Motion reconstruction using pre-record motion capture data is an important topic in computer animation. However, the most popular technique to reconstruct full-body motion is through vision or magnetic-based motion capture device which is high-cost, time-consuming for setup, and applicable only in constrained environment. The technique does not meet our primitive goal. Therefore, we prefer to choose the sensing devices which are low-cost, portable and fewer environment constraints.

In our system, we use Wii Remotes with MotionPlus as our motion sensor (Figure 1.1, Figure 1.2), where accelerometer and the gyroscope inside provide us information of 3D accelerations, and angular velocities. This motion capture device is the primary controller for Nintendo's Wii console. A main feature of the Wii Remote is its motion sensing capability, which allows the users to interact with objects on screen via gesture recognition and pointing through accelerometer and optical sensor technology. The accelerometer captures net forces on Wiimote in the range from -3*g* to 3*g*, where *g* is gravitational acceleration.

The Wii Remote assumes a one-hand remote-control-based design instead of the traditional gamepad controllers in previous gaming consoles. The controller communicates wirelessly with the console through short-range Bluetooth radio up to 10 meters away from console.

Figure 1.1: Wii Remote

Figure 1.2: Wii Remote with MotionPlus

These inertia sensors can easily be worn by users as input, and it does not affect the daily lives. Inertia controllers provide us accelerations and angular velocities, and can be used to deduce the motion of user at present. We have implemented an approach [Tautges et al 2011] to match the most appropriate result by gathered data. This data-driven technique is built up a *Lazy Neighborhood Graph* in an online fashion based on the sparse accelerometer input. However, an obvious limitation of this method is that occasionally jumps between poses may occur. The problem often occurs when we receive ambiguous sensing data sequences.

In [Lee et al 2010], a technique called *motion field* is proposed. This novel run-time motion synthesis mechanism allows a natural handling of several ambiguous candidate sequences by calculating the distance between candidate sequences and synthesizing with weighted average. Therefore, we can modify *Online Lazy Neighborhood Graph* achieve motion transition rapidly by the concept of this technique, we solve above limitation and make the result appear smooth for better action surveillance. (Figures 1.3)

We focus on the daily behavior like walking, lifting, sitting, etc, and plan to recognize the abnormal behaviors of user. In order to achieve the goal of monitoring, we use the Bluetooth to transfer data, and it can transmit sensing data more than dozens meters. Our approach takes advantage of the inexpensive and portable motion capture device and can still provide comparable accuracy to other technique using high-cost devices.

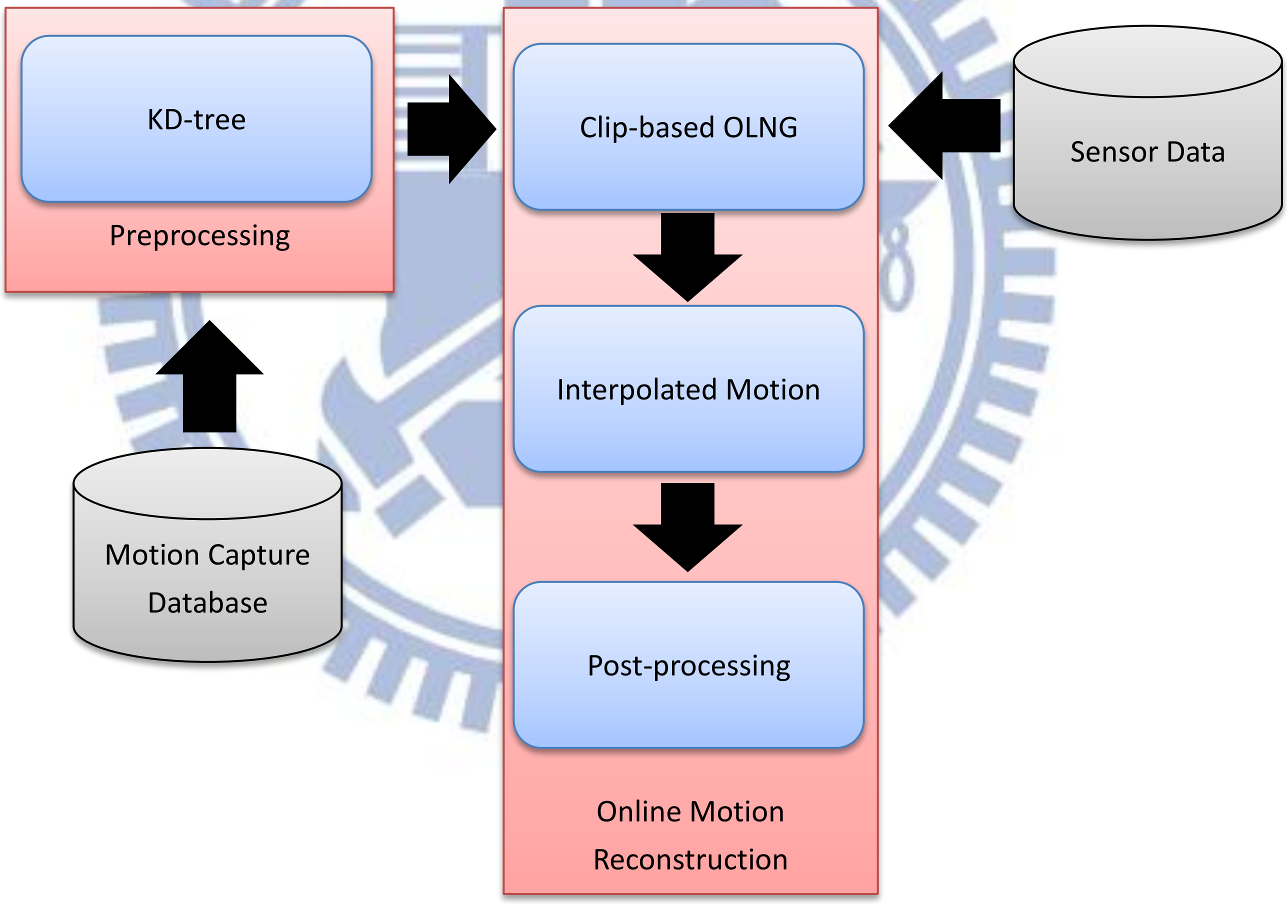


Figure 1.3: Overview of system

Chapter 2.

Related Work

In last decades, various motion capture devices and techniques have been proposed. Each motion capture technique has its own advantages and weaknesses. Take [SH08a] for example, they attached low-cost inertial sensors on user's limbs. By measuring the acceleration from human motion, they can match the closet motion clip in their database. However, the acceleration computed from full-body pose is often disturbed by noise.

Another technique like optical marked-based MoCap systems can typically provide accurate positional information for joint coordinates or rotational information of joints in [PhaseSpace10]. However, the method requires an array of calibrated high-resolution cameras as well as high-cost garment equipment. They usually need manual data clearing for occlusion and ambiguity problems.

Low-dimensional sensor input is often used for acquiring full-body information in computer animation [BHG93]. Siratori, T. [SH08b] introduces using inertial-based control data to evaluate a small number of parameters in physically-based character animation. Data-driven approaches show promising results to generate high-dimension character motion with limited-dimensional control data. Feng, W. –W. [FKY08] proposed an approach using sparse control points and an example-based model to deform complex geometries. The above mentioned techniques reconstruct virtual character pose through pre-record mocap data. The data in mocap database usually have huge dimension to represent full-body human of one frame. It is time-consuming to search high dimensional data. Therefore, how to use the low-dimensional sensors input to retrieve suitable motion sequences from a database which contains of high-dimensional data is main issue of our method.

Andoni, A. [AI06] stated the kd-tree is well suited for nearest-neighbor searches. By kd-tree structure, they can efficiently identify the pose in the pre-record database. The technique inspires us to retrieve the full-body human pose from high-dimensional knowledge database with a given sensing input. Kruger, B. [KTW10] extended above technique and introduced *Lazy Neighborhood Graph(LNG)*. This method is used in the reconstruction step to compute the current frame of the outputted animation. However, we want to identify optimal subsequences from the knowledge database for every point in time. To construct LNG for every frame of sensing data is of high computation cost and do not improve the visual quality.

Tautges, J. [TZK11] improved LNG. LNG can be built up incrementally making its construction efficient and online capable, which called *Online Lazy Neighborhood Graph(*OLNG*)*. Therefore, the technique allows us for handling the newest sensor readings and updating the LNG immediately.

In addition to motion reconstruction, we have to deal with the problem of discontinuous gaps between poses caused by ambiguous data streams. OLNG finds the best *K* nodes of the current time by *K*-nearest neighbor algorithm(KNN) and uses them to calculate energy minimization. It is possible to have several diverse sequences with similar low energy cost. To reconstruct motion with smooth transition, there are some methods that use nonparametric methods to learn the dynamics of character motion in a fully continuous space. Arikan, O. [AFO03] presented an algorithm synthesizing motions by users specifying what actions should occur during the motion as well as specifying modifiers on the actions. However, when this method anticipates some types of upper-body pushes, the character may not react at all to hand pulls or lower-body pushes. Another group of methods use nonparametric models to learn the dynamics of character motion in a fully continuous space [YL10, CH05]. These techniques are able to synthesize character motions starting from the initial states and make themselves apply physical disturbances. These models are used to estimate the most likely character motion from a number of possible candidate postures. As a result, we can utilize above concepts to construct smooth motion transition without jitter.

In the thesis, we incorporate temporal coherence by *Online Lazy Neighborhood Graph*(OLNG) and attempt to overcome its shortcoming. We extend the concept of *motion field* proposed by [LWB10] to do motion blending to avoid jittering. The difference between *motion field* and [YWB10, CH05] is that instead of building a model of the most possible single motion, they attempt to model the set of possible motions at each character state and select the single state at run time by calculating the value function. Their work combines the concepts of near-optimal character control presented in graph-based methods with those of nonparametric motion estimation techniques. Simply, we propose *Online Lazy Neighborhood Interpolation Graph* by extending the concept of *motion field* into original *Online Lazy Neighborhood Graph.*

In out thesis, we gather acceleration data by sparse inertial sensors worn on the user's body instead of using those high-cost optical motion capture devices for input. The advantages of Wii Remote are portable to not hinder daily behavior and adaptable in constraint environments. Our approach is capable of handling variations that are not explicitly specified in the given database.

Chapter 3.

Pre-processing

3.1 Motion Capture Database

In this system, we select motion capture data as our training data from CMU Mocap Lab. The database consists of various motions including sport motions and common behaviors. To focus on everyday life of users, we choose the action like walking, jumping, sitting, and lifting as our training input. Every motion is in the Biovision Hierarchy(BVH) format and consists of thousands and even ten thousands of frames. In the following, we rename the different knowledge bases by the same naming pattern that group the database simply.

{Motion}{Index}.bvh

At first, we use unprocessed database as input to construct *Online Lazy Neighborhood Graph*. However, it is time-consuming to clip motion sequences from such a large database. To deal with this problem, we calculate the distance between two frames in each motion by Euclidean distance. If two poses are too close to each other, they are then considered as redundant data. We skip the frame whose distance between itself and the next frame is smaller than a threshold. The threshold is 0.3 *cm* that can reduce up to 14 ten thousand to 7 ten thousand frames.

3.2 Motion Clips

In our work, training data is a huge database that we need a lot of time to construct OLNG for every frame. Therefore, we clip the training data into several sequences to mitigate the time of construction of OLNG. A sequence consists of *n* frames, 31 joint angles, and root positions. It is intriguing to define the number of a clipped sequence. In our system, we decide *n* is 10 to 15. First of all, we divide whole set of motions into several parts evenly. Each clip has *n* frame. *n* is user-defined. In the second step, it selects the first frame of each clip as training data that constructs kd-tree structure. By only using one frame of motion clip, we can substantially make the system speed up. Therefore, sensor readings find out the appropriate output through kd-tree structure that the output belongs to a first frame of motion sequence. The motion sequence becomes the candidate of motion synthesis. As a result, we only process nearly one-tenth training data to construct OLNG.

When one index is chosen, we synthesize the whole motion clip belonging to the index. In addition, we use OLNG to choose coherent index to avoid ambiguity problem. Because we use clip-based motion reconstruction, our overall animation is smoother than the per-frame-based motion reconstruction.

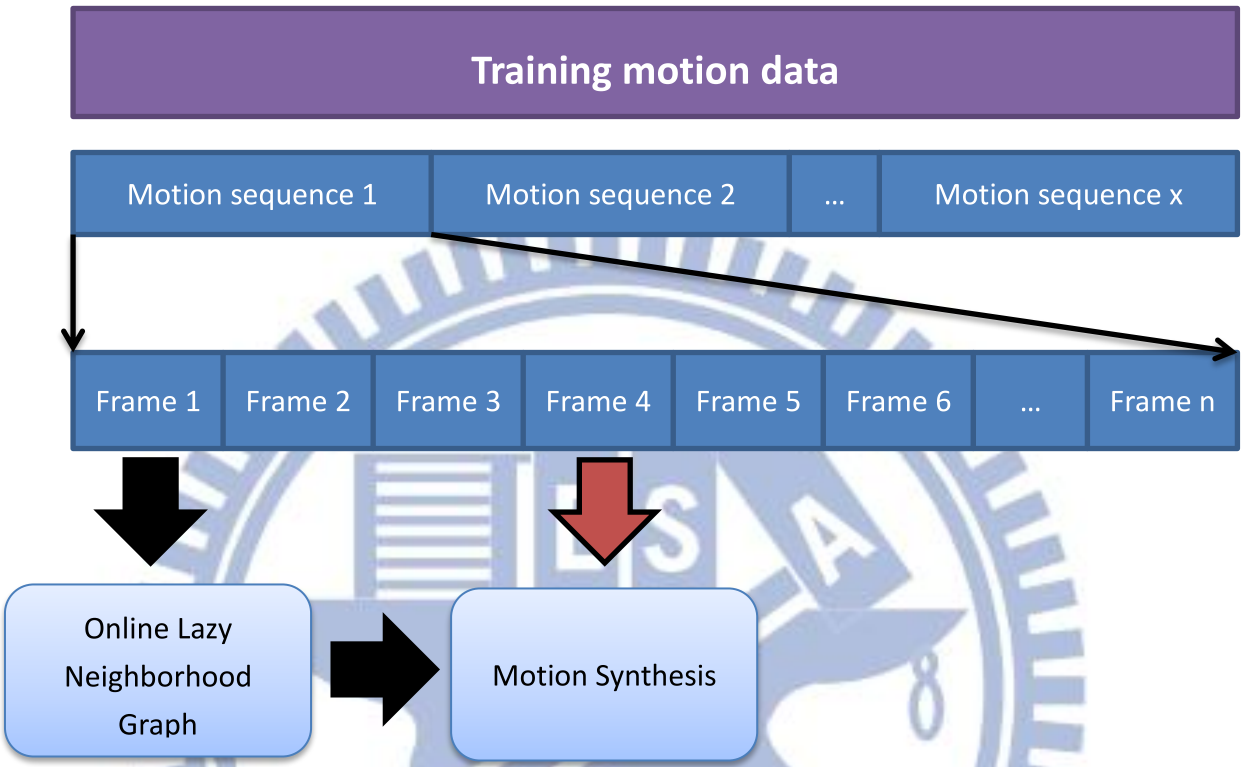
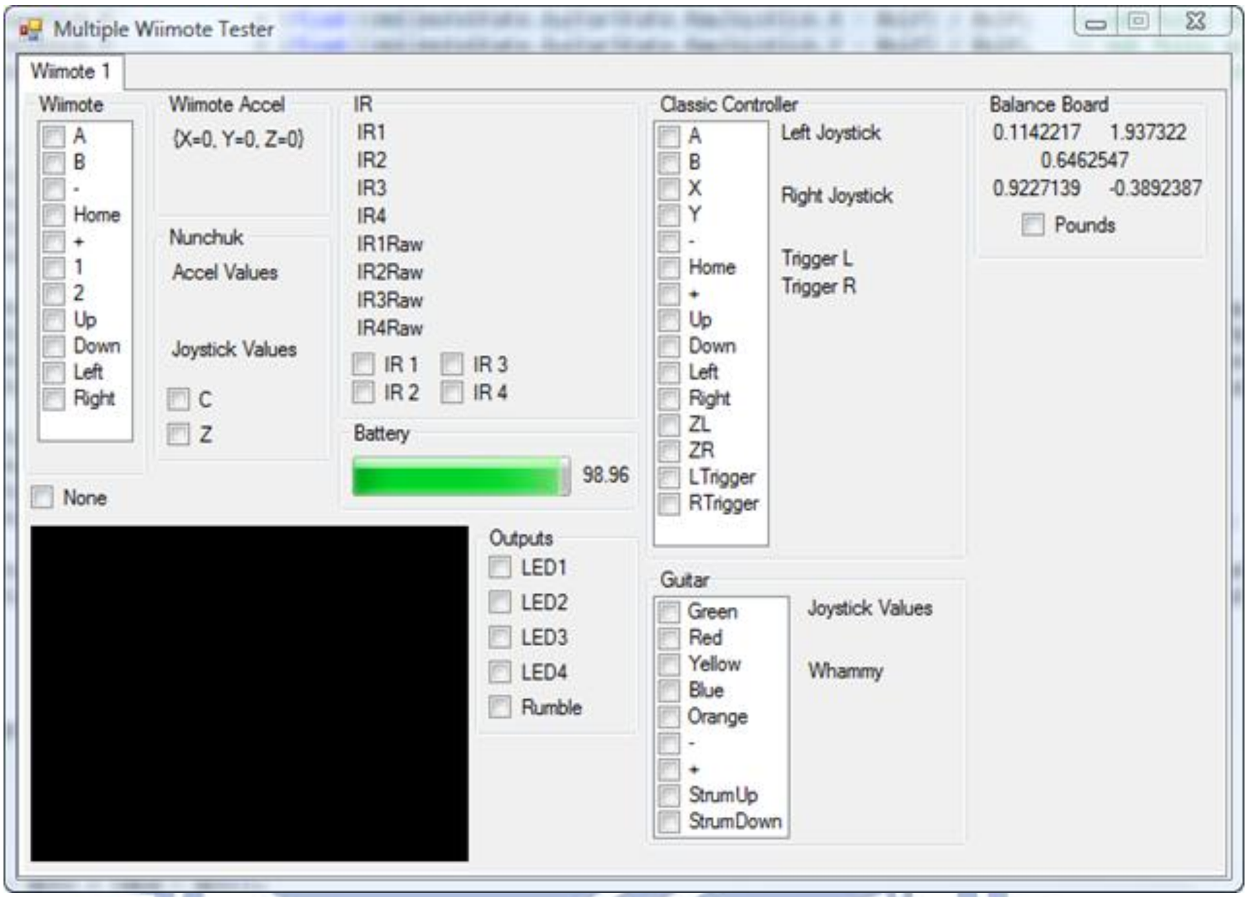


Figure 3.1 We clip the data to several parts and let the first frame of each clip be index. We use index to construct OLNG. Each index presents *n* frames.

3.3 Sensor Reading Collection

We ask a user to wear Wii remotes with MotionPlus on two wrists, legs, and chest. By WiiYourself library, we can gather the acceleration from Wii remotes via Bluetooth technology. This library supports multiple Wii remotes and provides us UI for convenient management as shown in Figure 3.2. Wii remotes send and receive various data, and all of them are 22 bytes in length. WiiYourself has a *FileStream* to communicate with and read or write to the Wii remote. In addition, because data are sent and received almost constantly, asynchronous I/O operations are used. To implement in .NET, the process is to start an asynchronous read operation and provide a callback method to be run when the buffer is full. When the callback function is run, the data from Wii remote is handled and the process is repeated.

For initialization, we require a user to stand in a T pose to do calibration before motion capture as shown in Figure 3.3. Inappropriate initial state usually cause biased reconstruction results. Then, a user moves freely within the range of Bluetooth can be reach transmission. Wii remotes worn on user's two wrists and legs, send 3D accelerations and angular velocities to the system. However, the Wii remote worn on user's chest only send orientation without 3D accelerations. We use the chest Wii Remote to determine root orientations, it makes estimated full-body pose smoother. Because the acceleration signal is noisy, we apply a low pass filter to denoise before the following data analysis.



{X=0, Y=0, Z=0}

98.96

0.1142217 1.937322
0.6462547
0.9227139 -0.3892387

Figure 3.2 WiiYourself UI

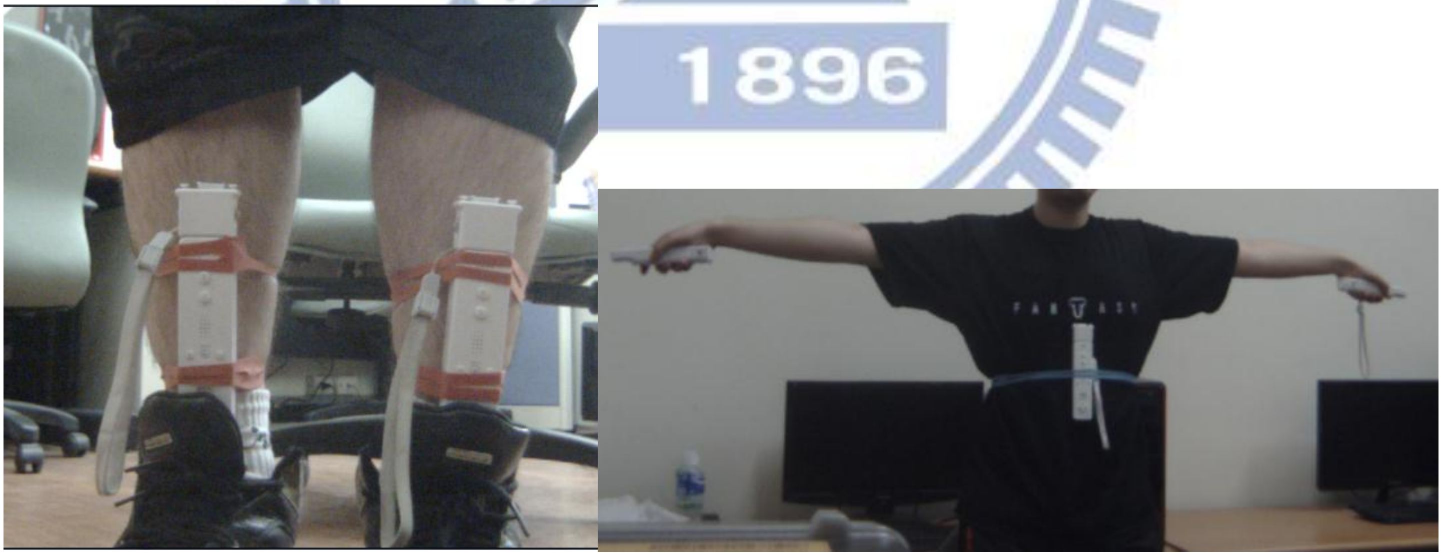


Figure 3.3 Wii remotes worn on user’s body

Chapter 4.

Implementation of Online Lazy Neighborhood Graph

4.1 Overview

In our work, we mainly divide the whole system into four stages. In the first stage, we construct *online lazy neighborhood graph* with acquiring fixed-length sequences of training data. Second stage is to use sparse low-dimensional control input to infer full-body motion. We formulate the motion reconstruction as an energy minimization problem and acquire the most possible candidate from training data. However, defect of the data-driven best-match approach is that occasional jumps between poses may occur. To deal with this drawback, we combine the *motion field* into our original approach to achieve rapid motion transition in the third stage. At the final stage, we do post-processing to make the character smooth. First, we handle the problem by detecting heights of two feet and keep supporting points stationary at ground plane. Furthermore, we use Gaussian filter to diminish unnatural full-body motion. The main advantage of our sensor-based approach is that we can still acquire reliable results in constraint environment like obstacle or dark environment, and keep connection up to 10 meters away from consoles. Consequently, our approach has higher applicability than those using calibrated high-resolution cameras as well as high-cost garment equipment.

4.2 Clip-based Online Lazy Neighborhood Graph

In this stage, we use 3D accelerations of four Wii remotes as the input of our reconstruction framework. The sensing data are represented in the unit m/s^2. Before constructing OLNG, we would use the training database to build kd-tree structure. As section 3.2 mentioned, we use the first frame of motion sequence as one node and calculate the 3D acceleration and angle velocities of two twists and legs. The 3D accelerations of each joint are then placed in kd-tree structure. The dimensions of kd-tree is 4(Wii Remotes) · 3(xyz) · *t*(time)=12*t*. The kd-trees are well suited for last nearest neighbor searches. (Figure 4.1)

Now, we assume that the control input consists of continuous stream of sensor accelerations $(\ldots,\alpha^{t-2},\alpha^{t-1},\alpha^{t},\alpha^{t+1},\ldots)$, where α^{t} denotes the current frame of 3D accelerations at time *t*, and *t* is an integer. We fix the number *K* of nearest neighbors and let S^{t} be the storages of the *K* nearest neighbors of α^{t}. We consider the last *M* sensing data $(\alpha^{t-M+1},\alpha^{t-M+2},\ldots,\alpha^{t})$ for a fixed number $M \in \mathrm{N}$. In our work, we choose *M* is 4. Then, the nodes of OLNG can be presented by *M* x *K* array. If there are two nodes in adjacent column, we should connect them as a path. Suppose that the OLNG has been constructed for the reading $(\alpha^{t-M+1},\alpha^{t-M+2},\ldots,\alpha^{t})$ and a new data α^{t+1} arrives. First, we should construct the path of last *M* column by the above concept. Then, we acquire the *K* nearest neighbors and store in S^{t+1}. We search whether the last frame of node is adjacent to the new node and form a connection. Finally, the nodes are corresponding to α^{t-M+1} as well as the involved edges are removed to obtain the updated OLNG. (Figure 4.2a, Figure 4.2b)

During real-time updating, our approach can process the newest sensing data. However, the K nearest neighbors of initial sensing data will bias the connection of path of following sensing data. So we take T pose as our initial sensing data to make result reliable.

In summary, the OLNG allows for various adjustments to speed up whole approach. Take the sliding window M for example, we can linearly speed up by changing M. Moreover, we reduce the operation of OLNG substantially by clipping the motion and only choosing the first frame of motion sequence as input. The space complexity of kd-tree is O(N) and OLNG is O(KM). Furthermore, each update step requires only O($K \log N$) operation. Therefore, OLNG with kd-tree is well suitable to huge datasets.

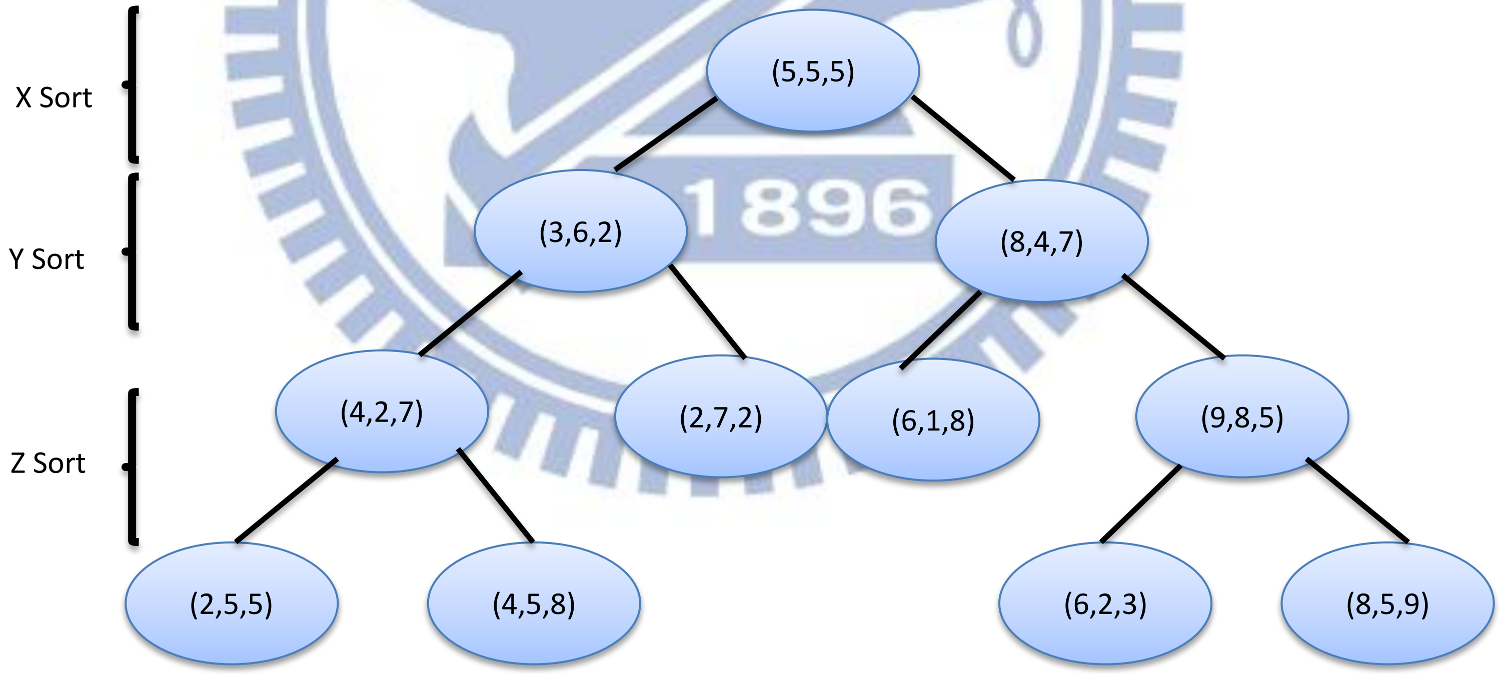


Figure 4.1 The resulting 3D kd tree

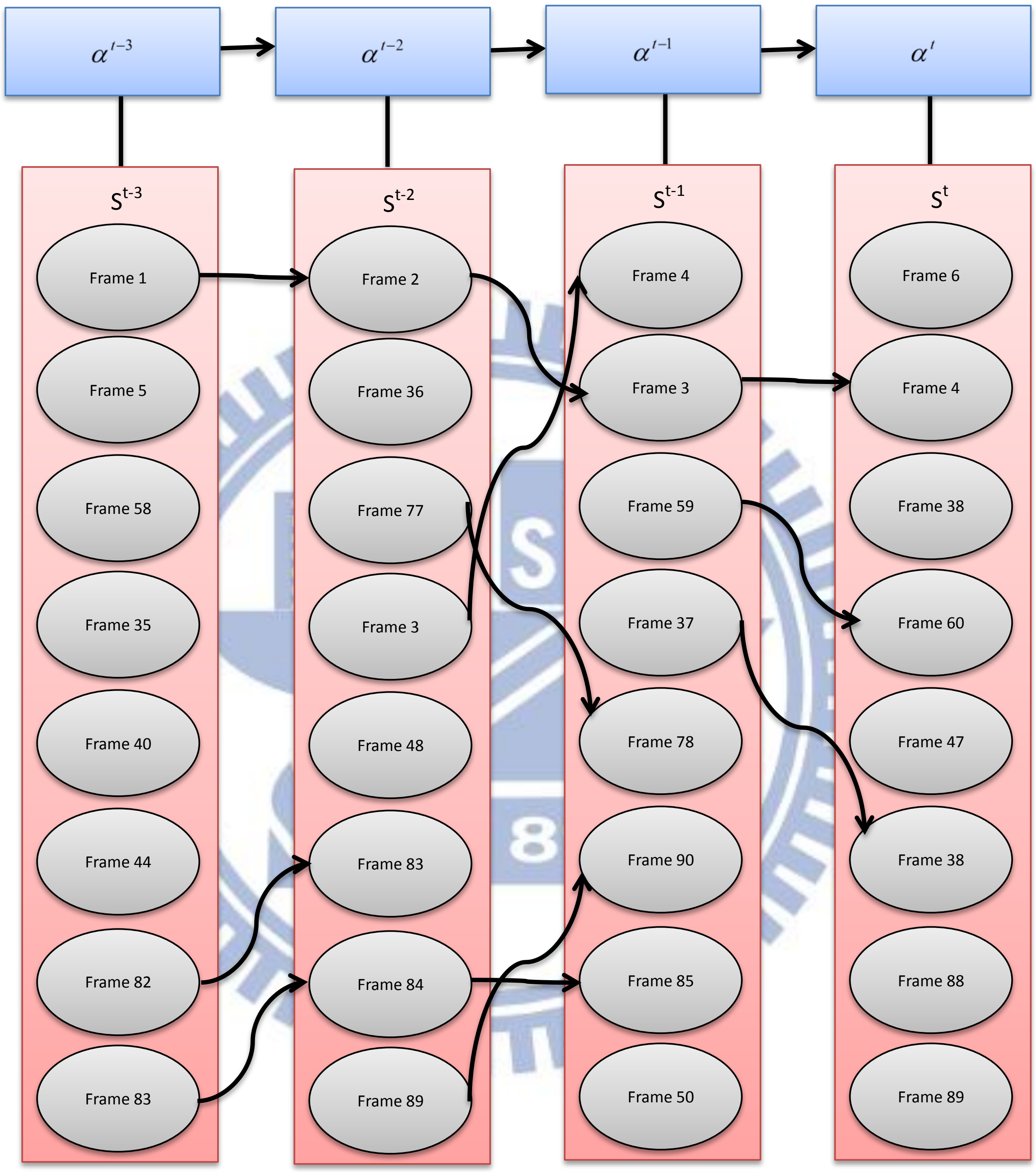


Figure 4.2a: Implementation of the OLNG
We search whether the last frame of node is adjacent to the new node and form a connection

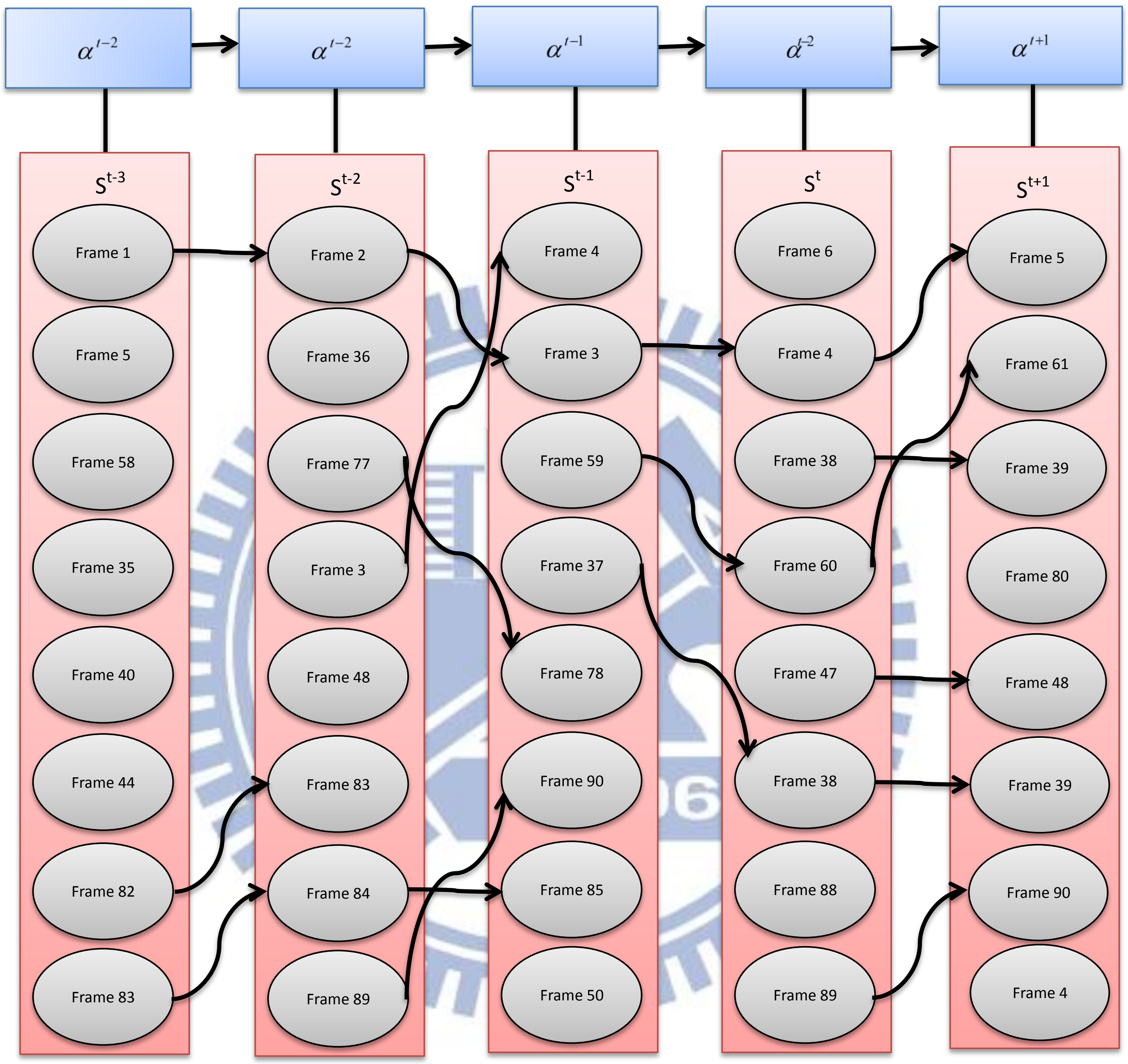


Figure 4.2b: Implementation of the OLNG
The nodes are corresponding to α^{t-M+1} as well as the involved edges are removed to obtain the updated OLNG

4.3 Motion Reconstruction

In this stage, we use a low-dimensional input to infer high-dimensional motions. We introduce the concept as follows. OLNG is proposed by Tautges, J. [TZK11]. The notations are the same as OLNG paper. When the new sensing data arrives, we connect *I* paths with existing paths. Then we consider how much those *I* paths cost. We denote $C^t = \{C_1^t, ..., C_I^t\}$ to be the cost of these *I* paths at time *t*. The cost of one path is calculated by the sum of Euclidean distance between two frames in this path. As mentioned above, we denote $Q^t = \{q_1^t, ..., q_I^t\}$ as the set of joint angles given by all these paths at time *t*, $X^t = \{x_1^t, ..., x_I^t\}$ as the positions, $V^t = \{v_1^t, ..., v_I^t\}$ as the velocities, and $A^t = \{a_1^t, ..., a_I^t\}$ as the accelerations of the joints with respect to the root coordinate system. These parameters were already computed in the Section 3.1. Then, based on the costs $C^t = \{C_1^t, ..., C_I^t\}$, we calculate normalized weights denoted by $W^t = \{w_1^t, ..., w_I^t\}$, where the value of each weight w_i^t is given by

$$w_i^t = \frac{\max(C^t) - C_i^t}{\sum_{j=1}^{I} (\max(C^t) - C_j^t)} . \tag{1}$$

Now, we acquire the costs of these *I* paths at frame *t*. When a new sensing data input α^{t+1} arrives from sensors, we formulate the motion reconstruction as an energy minimization problem to choose the suitable pose. First, the OLNG is updated and we would acquire Q^{t+1} and W^{t+1} from updated OLNG. Furthermore, we attempt to find a pose q_{best} that optimally satisfies constraints imposed by the observation, and the pose must be consistent with similar motion clips retrieved from the database.

$$q_{best} = \arg\min_{q}(\omega_{prior} \cdot E_{prior}(q) + \omega_{contr} \cdot E_{contr}(q)). \quad (2)$$

The two weights ω_{prior} and ω_{contr} are user-defined parameters, E_{prior} is energy minimization of prior term, and E_{contr} is energy minimization of control term. First, we discuss the prior term. The prior term is composed of three components as pose prior, motion prior, and smooth prior. In addition, for a huge database, a data-driven approach uses a-prior likelihood based on the motions given by knowledge base. Therefore, the method is used which can avoid implausible result. Now, we analyze those three terms one by one. First, the pose prior according to joint angles to characterizes the probability of a pose. Second, the motion prior according to joint positions of a pose to regards the temporal evolution of a motion. Last, the smooth prior according to accelerations to calculate continuity between two poses to reduce jerkiness. Using above three terms, we can compute E_{prior} with three weights ω_{pose}, ω_{motion}, and ω_{smooth}

$$E_{prior}(q) = \omega_{pose} \cdot E_{pose}(q) + \omega_{motion} \cdot E_{motion}(q) + \omega_{smooth} \cdot E_{smooth}(q) \quad (3)$$

Here, the cost of pose prior is computed by a kernel based approach. We approximate the likelihood p_{pose} of a synthesized pose candidate *q*.

$$p_{pose}(q) \propto \sum_{i=1}^{I} \omega_i^{t+1} \cdot \kappa(|\, q_i^{t+1} - q \,|) \quad (4)$$

where κ is a symmetric kernel function. In our work, we suppose that p_{pose} is maximized for poses that are likely according to the training motion input and re-formulates p_{pose} to suitable for energy minimization.

$$E_{pose}(q) = \sum_{i=1}^{I} \omega_i^{t+1} \cdot \sqrt{|\, q_i^{t+1} - q \,|} \quad (5)$$

Then, we discuss the cost of motion prior. Besides being plausible on a pose level, the motion reconstruction should be consistent with motions in reality. In other word, our reconstructed pose should be within the feasible space of human posture. The movement of joint would be natural and convincing. When the new sensing data arrives, we can measure the angle velocities V^{t+1} and 3D accelerations A^{t+1} from database poses included in Q^{t+1}. By second order Taylor expansion, we estimate a probability density distribution for x^{t+1} with V^{t+1} and A^{t+1}. For the i-th sample ($i \in \{1,\ldots,n\}$) the estimated position $x_i'^{t+1}$ are then given by

$$x_i'^{t+1} = x^t + v_i^{t+1} \cdot \Delta t + \frac{1}{2} a_i^{t+1} \cdot \Delta t^2 . \tag{6}$$

Like p_{pose} function, we use a kernel-based approach to present p_{motion}. Moreover, we substitute joint position x for joint angle q because of the energy minimization.

$$E_{motion}(x) = \sum_{i=1}^{I} \omega_i^{t+1} \cdot \sqrt{| x_i^{t+1} - x |} \tag{7}$$

Last, we discuss the cost of smooth prior. Energy minimization would acquire plausible results and is high frequency jitter may occur between two poses. To reduce this situation, we attempt to enforce smoothness by minimizing joint accelerations and make use of a-prior knowledge provide by training database. A pose q is assumed to be plausible, if its joint accelerations are consistent with the joint accelerations of neighboring database samples. Like p_{pose} function, the likelihood of a pose candidate is measured by kernel based density estimation

$$E_{smooth}(a) = \sum_{i=1}^{I} \omega_i^{t+1} \cdot \sqrt{| a_i^{t+1} - a |} \tag{8}$$

where

$$a = \Delta t^{-2} \cdot (x - 2x^t + x^{t-1})$$

Through pose prior, motion prior, and smooth prior, we can infer high-dimensional full-body pose by a-prior likelihood using low-dimensional acceleration space.

Next, we discuss the control term. In our work, we use 3D accelerations to retrieve the most appropriate motion sequence as result. However, in the control term, a direct use of 3D accelerations as input is not suitable because accelerations are not powerful enough to reconstruct motion. Therefore, the control term is computed based on joint positions that acquired by Wii remotes.

First, we let $\langle y \rangle$ be the projection of a vector *y* to the subspace formed by the components related to those joints which are next to the sensors. Assuming we know the proper positions x^t at frame *t*. We can estimate the probability density distribution of the next joint position at frame *t*+1 by the set of velocities V^t at time *t*

$$\tilde{x}_i^{t+1} = \langle x^t + v_i^t \cdot \Delta t \rangle + \frac{1}{2} \hat{a}^t \cdot \Delta t^2 \tag{9}$$

where $\hat{a}^t$ is computed by transforming control signal reading a^t at to root frame coordinates by using the local frames induced by the previously synthesized pose q^t and subtracting gravity. We use $\{\tilde{x}_i^{t+1} \mid i \in [1:I]\}$ to derive the energy term to be minimized

$$E_{contr}(x) = \sum_{i=1}^{I} \omega_i^{t+1} \cdot \sqrt{| \tilde{x}_i^{t+1} - \langle x \rangle |} \tag{10}$$

we can avoid overshooting effects and synthesize smooth motion transition by using velocities.

Last, we incorporate prior term and control term into energy minimization problem. The function is in Equation 2. We define the weights for energy minimization: $\omega_{contr} = 1, \omega_{prior} = 5, \omega_{pose} = 0.6, \omega_{motion} = 0.2, \omega_{smooth} = 0.2$. We can slightly change those weights to adjust the reconstruction results. The motion clip with highest probability can be extracted by our energy minimization. Through the concept of OLNG from [TZK11] we attempt to add motion blending into OLNG to avoid jitter.

We also consider the continuity of velocity between the first frame and the central frame of one motion clip and attempt to modify the energy minimization. The modified energy minimization algorithm is as follow:

$$q_{best} = \arg\min_{q}(\omega_{prior} \cdot E_{prior}(q) + \omega_{contr} \cdot E_{contr}(q) + \omega_{veloc} \cdot E_{veloc}(q)).$$

$$\omega_{veloc} = 1$$

$$E_{veloc}(q) = \sum_{i=1}^{I} \omega_i^{t+1} \cdot \sqrt{|v_i^{t+1} - v|} \qquad (11)$$

However, the continuity of velocity does not increase the accuracy through our experimental result as shown in Figure. 4.3.

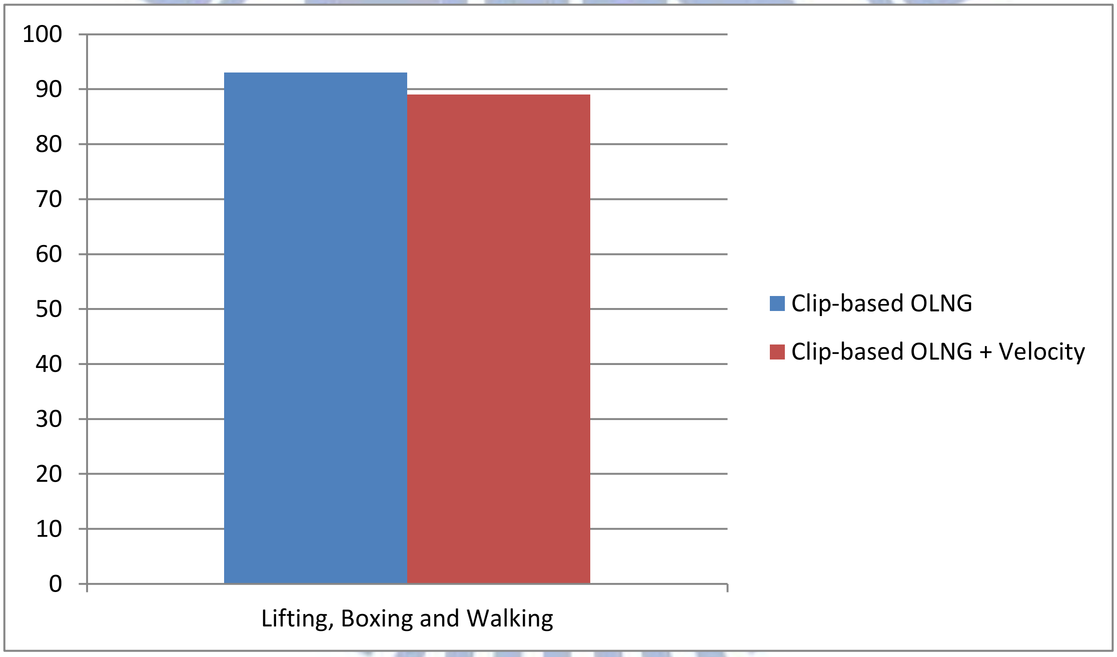


Figure 4.3 : A set of motion state

Chapter 5.

Interpolated Clip-based OLNG

5.1 Overview

In this chapter, we attempt to modify OLNG to handle a smooth motion transition with noise disturbed motion. We adopt a structure called a *motion state* and attempt to let the first frame of all candidate motion clips be one state. After the OLNG step, we find the highest priority of character pose, but there are other candidates of motion clips from the newest sensing data. To reconstruct motion without jitter, we make use of all valid candidates to synthesize output motion. By estimating the distance between the pose of highest priority candidate and the other candidates of motion clip, we synthesize output motion by weighted combination and make the animation look natural. Therefore, this approach prevents the character from replaying training data of database and allows the character to perform more flexible pose that are not explicitly specified in the given database. Furthermore, because there are always multiple candidates of motion clips to be considered, the character constantly has a variety of paths to perform rapid motion transitions. In this chapter, we utilize the concept of *motion field* to modify *Online Lazy Neighborhood Graph*, we would reconstruct more variety of character animation to achieve natural posture.

5.2 Interpolated Motion State

We let the first frame of motion clips as shown in chapter 3.2 be one state. The state consists of 3D root positions and joint orientations at top frame. A pose $x = (x_{root}, p_0, p_1, ..., p_n)$ represents this state, where x_{root} is 3D root position vector, p_0 is root orientation, and $p_1, ..., p_n$ are joint orientations. Then, we define a motion state $m = x$ as a pose. After OLNG step, we connect the closest paths and have I candidates of motion clips. We construct a set of motion states $\{m_i\}_{i=1}^{I}$ termed a motion database as shown in Figure 5.1.

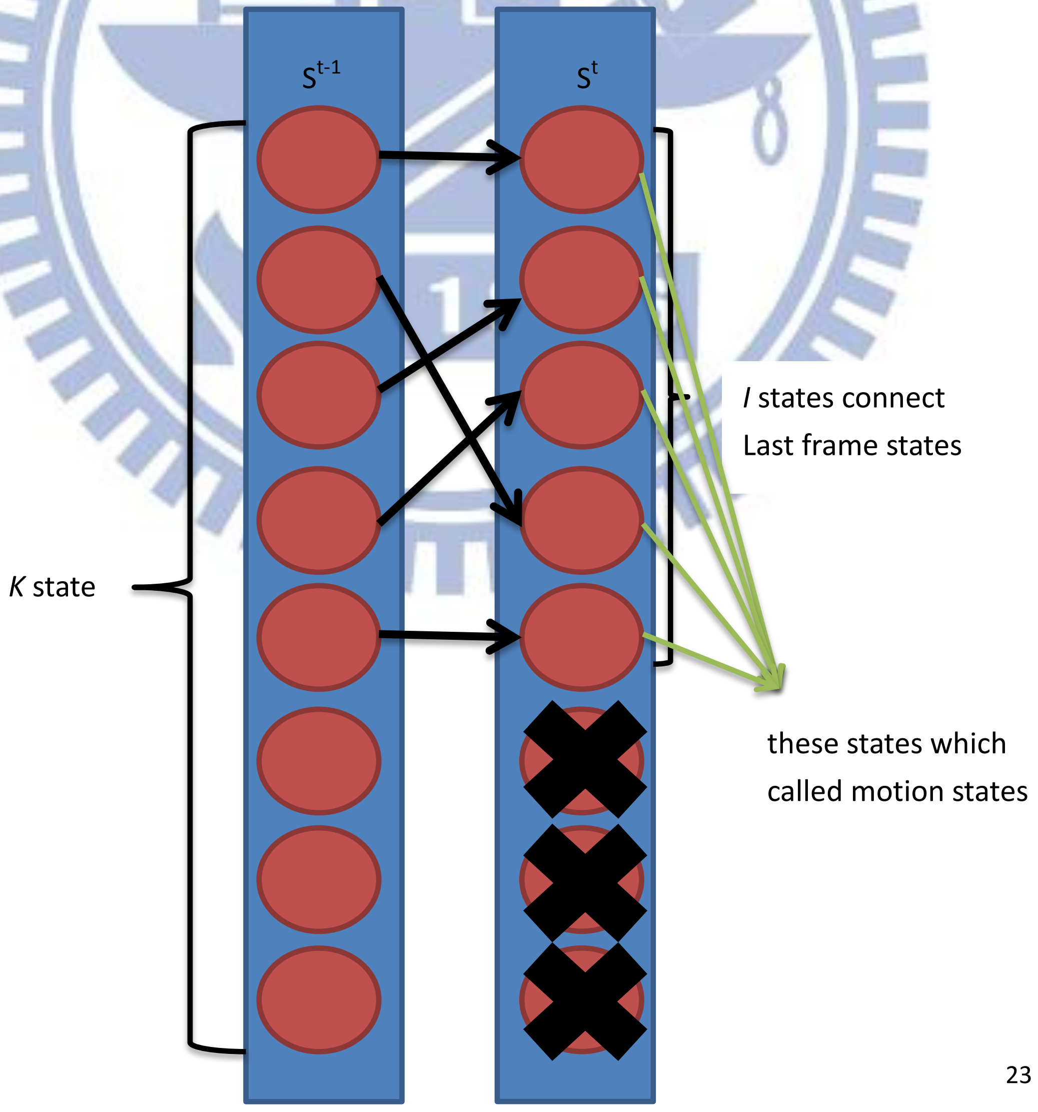


Figure 5.1: A set of motion state

Now, we should calculate the distance between two motion states. Given a motion database, we computer a *neighborhood* $N(m) = \{m_i\}_{i=1}^{I}$ of the I most similar motion states via a *k*-nearest neighbor query from the database. We choose the highest priority of candidate as motion state $m_{highest\_priorty}$ and the other candidates as motion state m'. In our approach, we use $I << K$ by $K = 8$. We calculate the similarity by

$$d(m_{highest\_priorty}, m') = \sqrt{\beta_{root}\left\|v_{highest\_priorty\,root} - v'_{root}\right\|^2 + \sum_{i=1}^{N}\beta_i\left\|p_i(\hat{u}_{highest\_priorty}) - p'_i(\hat{u})\right\|^2}$$

(12)

where N is the number of joints, $\hat{u}$ is the distance between two joints of angle, v_{root} is the velocity of root, $p(\hat{u})$ means the rotation of $\hat{u}$ by p, and the weights $\beta_{root}, \beta_1, \ldots, \beta_I$ as user-defined scalar parameters. We set β_{root} as 1000 and β_i as bone lengths of the body at the joint *i*. The bone length is computed by the difference of 3D absolute coordinates between the current bone and its previous hierarchy of bone.

Since we allow the character to deviate from motion states in the database, we frequently have to interpolate data from our neighborhood $N(m)$. We use the *similarity weights* $[w_0, w_1, \ldots, w_I]$ since they measure similarity to the current state *m*

$$w_i = \begin{cases} \frac{1}{\eta}\cdot\frac{1}{4d(m_{highest\_priorty}, m_i)^2}, if\ m_{highest\_priorty} \neq m_i \\ 0.75, if\ m_{highest\_priorty} = m_i \end{cases}$$

(13)

where m_i is the *i*-th candidate of m and $\eta = \sum_i \frac{1}{4d(m_{highest\_priorty}, m_i)^2}$ is a normalization factor to ensure the weights sum to 0.25. We set the weight as 0.75 if this motion state is the highest priority of candidate because the synthesized motion should be similar to our result of energy minimization problem. In other words, we attempt to adjust the pose of the highest priority by other candidates to match user's real pose.

Finally, we weighted combine the set of all candidates of motion clips via similarity weights.

$$p_{combine} = \sum_{i=1}^{I} w_i \sum_{j=1}^{n} p_j \qquad (14)$$

where p_j is the *j*-th joint angle of pose. (Figure 5.2)

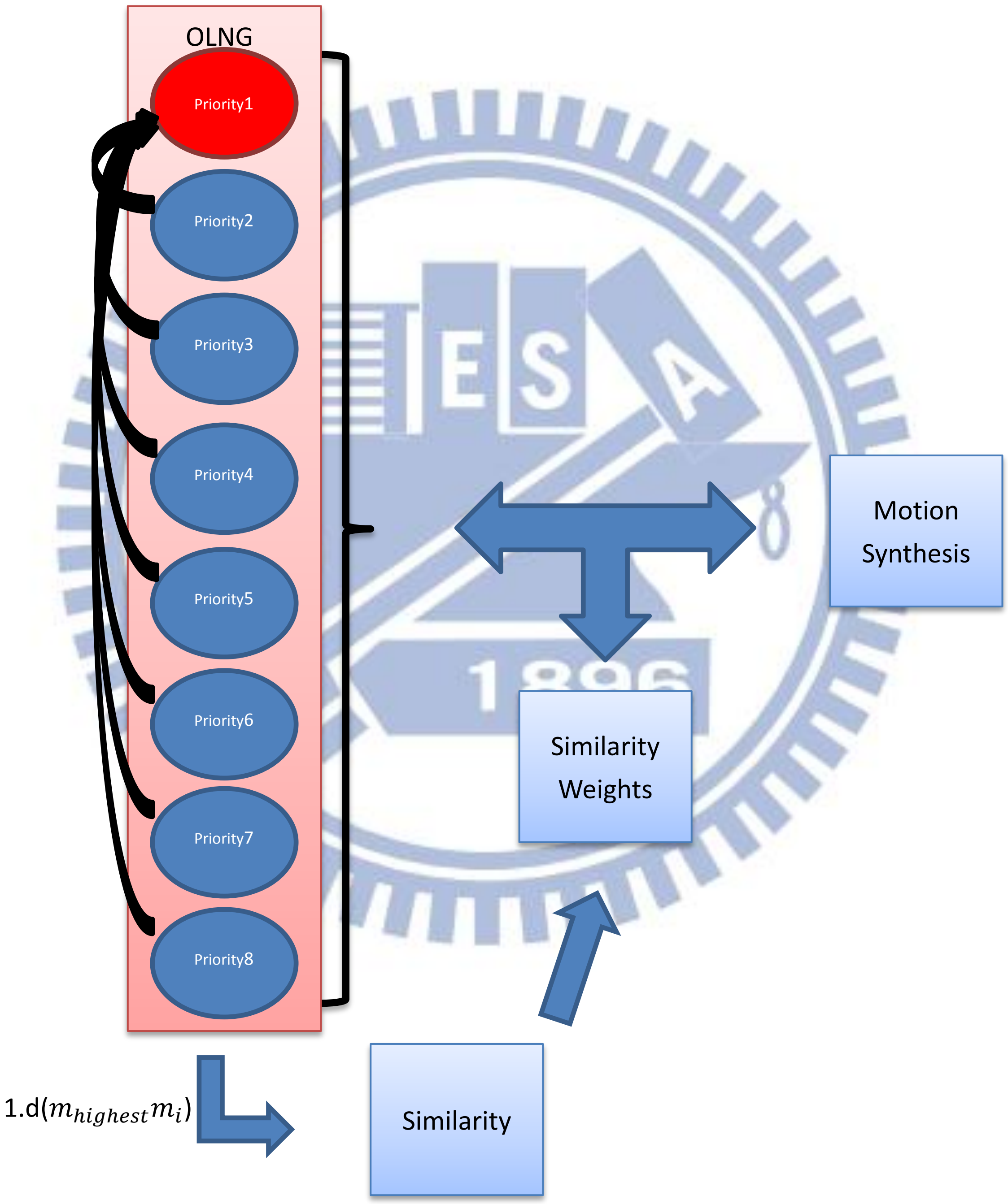


Figure 5.2 Framework of blending motion

5.3 Post-processing

Although we synthesize smooth transition motion by modifying *Online Lazy Neighborhood Graph*. However, we still have to deal with foot contact and unusual high frequency motion to make our result more natural. In this section, we discuss foot skating and low-pass filter problem. First, since we blend a variety of path of motion clips, the result occasionally have visual artifacts. For example, if we blend motion clips of walking and climbing, the character may look like walking in the air. However, the two actions are visually similar to each other but *y*-axes of those are different. Therefore, we attempt to fix the contact foot on the ground if the user stands on the ground.

In the beginning, we acquire the 3D coordinates of two tiptoes from synthesized motion. Then, we determine which tiptoe is lower and record the 3D world coordinates. If the y-axis coordinate is below the ground plane, we raise the full-body character pose until the height of tiptoe is identical to ground plane. Otherwise, if the y-axis of coordinate is above the ground plane, we estimate the character action and determine whether we move the synthesized motion to ground plane. By this technique, our reconstruct motion would be smoother without ups and downs. (Figure 5.3)

Finally, after removing the foot skating, we attempt to make the set of synthesized character motion smoother. Although our candidates of motion clips are continuous with last frame, we mix them into a new pose by similarity weights and let them be visually discontinuous in velocity changes. To make two adjacent frame of synthesized pose look natural, we use low-pass filter to temporally blend poses with convoluted filtering.

$$p_{low-pass\_filter}(n) = \sum_{i=n-3}^{i=n+3} w_{i-n+3}\, p(i) \qquad (15)$$

where $p(i)$ is the pose at frame I and w is the set of $[w_0, w_1, ..., w_6]$. We set $w_0 = w_6 = 0.006, w_1 = w_5 = 0.061, w_2 = w_4 = 0.242, w_3 = 0.383$. The temporal blending technique is a kind of Gaussian filter and it alleviates the discontinuous motions in real-time.

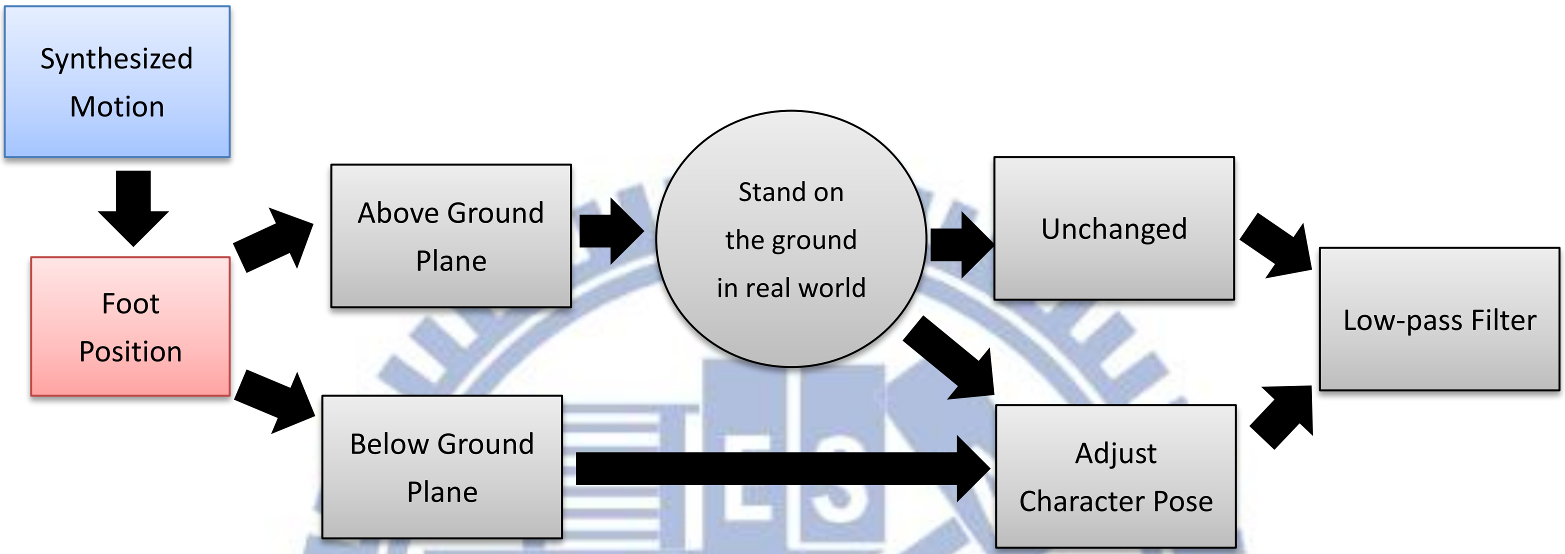


Figure 5.3: Framework of foot-skating removal and low-pass filter

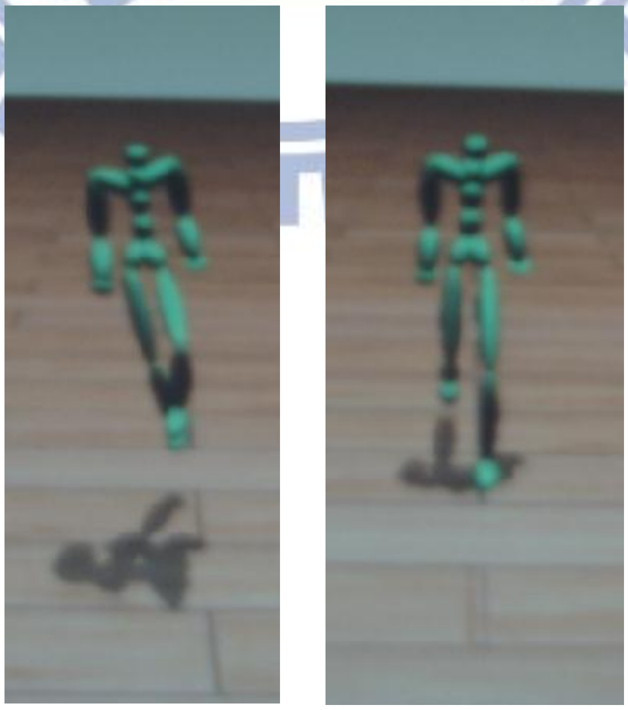


Figure 5.4: left: motion without dealing foot-skating
right: motion with dealing foot-skating

Chapter 6.

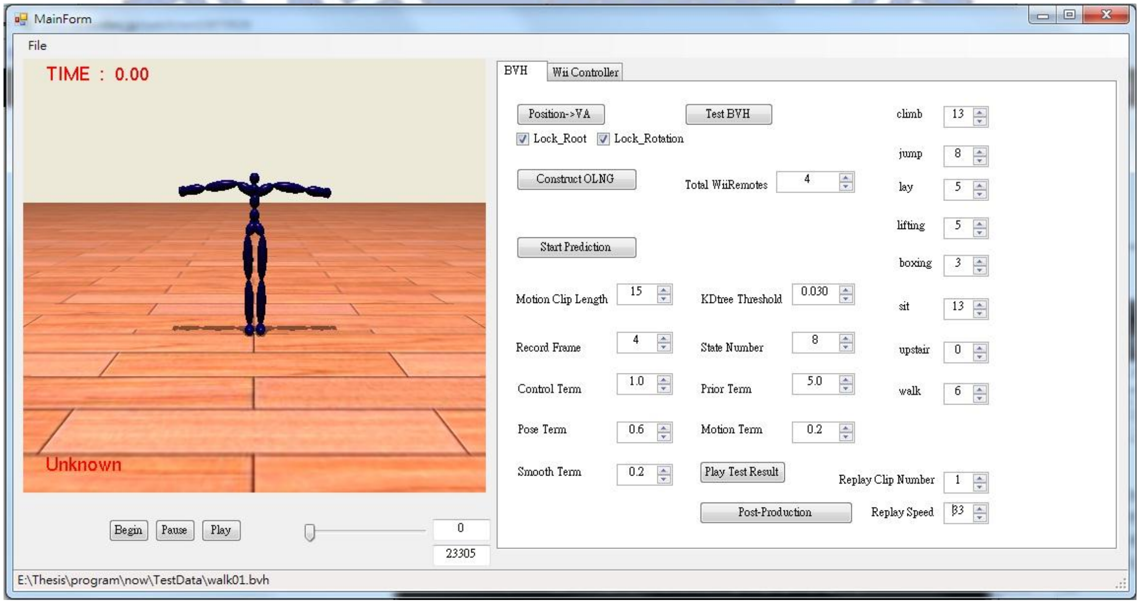
Experiments and Results

Our system is implemented by using C++ language and built based on Visual Studio 2008. OpenGL, MATLAB, and WiiYourself libraries are also used in our system. There are seven motions which totally have 135942 frames in training database, which are climbing, jumping, lying, lifting, boxing, sitting, and walking. Our approach needs to spend about 200 seconds producing 3600 frames animation.

Figure 6.1 is the screenshot of our system. The synthesized motion is shown on the screen and estimated what the motion to be. Our system is composed of two tab page, which are BVH page and Wii controller page as shown in Figure 6.2 and Figure 6.3. In BVH page, the user-defined parameter can be adjusted here. Moreover, we can detect the information of Wii Remote in Wii Controller page.

	Frame Number	Joint Number
Climbing	30219	31
Jumping	3062	31
Lying	3625	31
Lifting	13964	31
Sitting	39882	31
Boxing	16556	31
Walking	28634	31

Table 6.1 Training Database Detail



[✓] Lock_Root [✓] Lock_Rotation

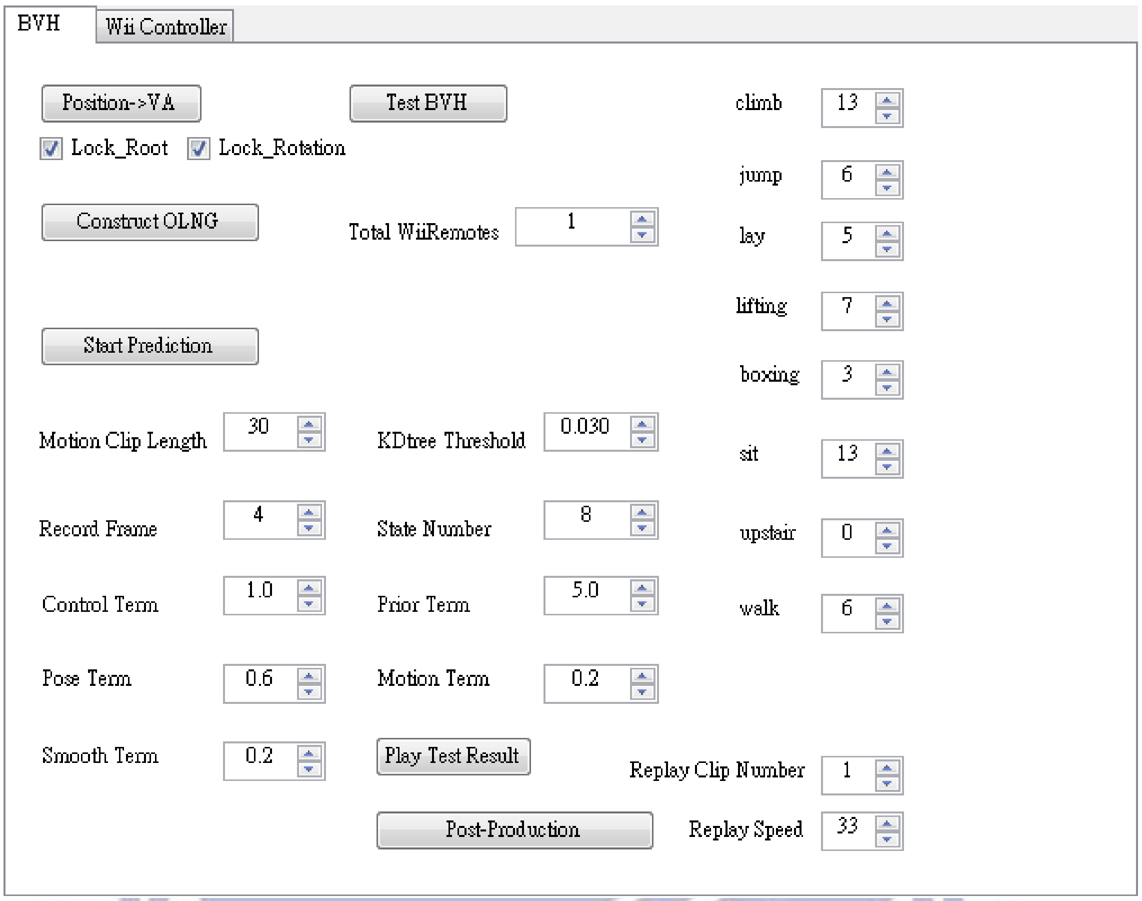
Total WiiRemotes	4			climb	13
				jump	8
				lay	5
				lifting	5
				boxing	3
Motion Clip Length	15	KDtree Threshold	0.030	sit	13
Record Frame	4	State Number	8	upstair	0
Control Term	1.0	Prior Term	5.0	walk	6
Pose Term	0.6	Motion Term	0.2		
Smooth Term	0.2			Replay Clip Number	1
				Replay Speed	33

0

23305

E:\Thesis\program\now\TestData\walk01.bvh

Figure 6.1 screenshot of our system


BVH | Wii Controller

Position->VA | Test BVH

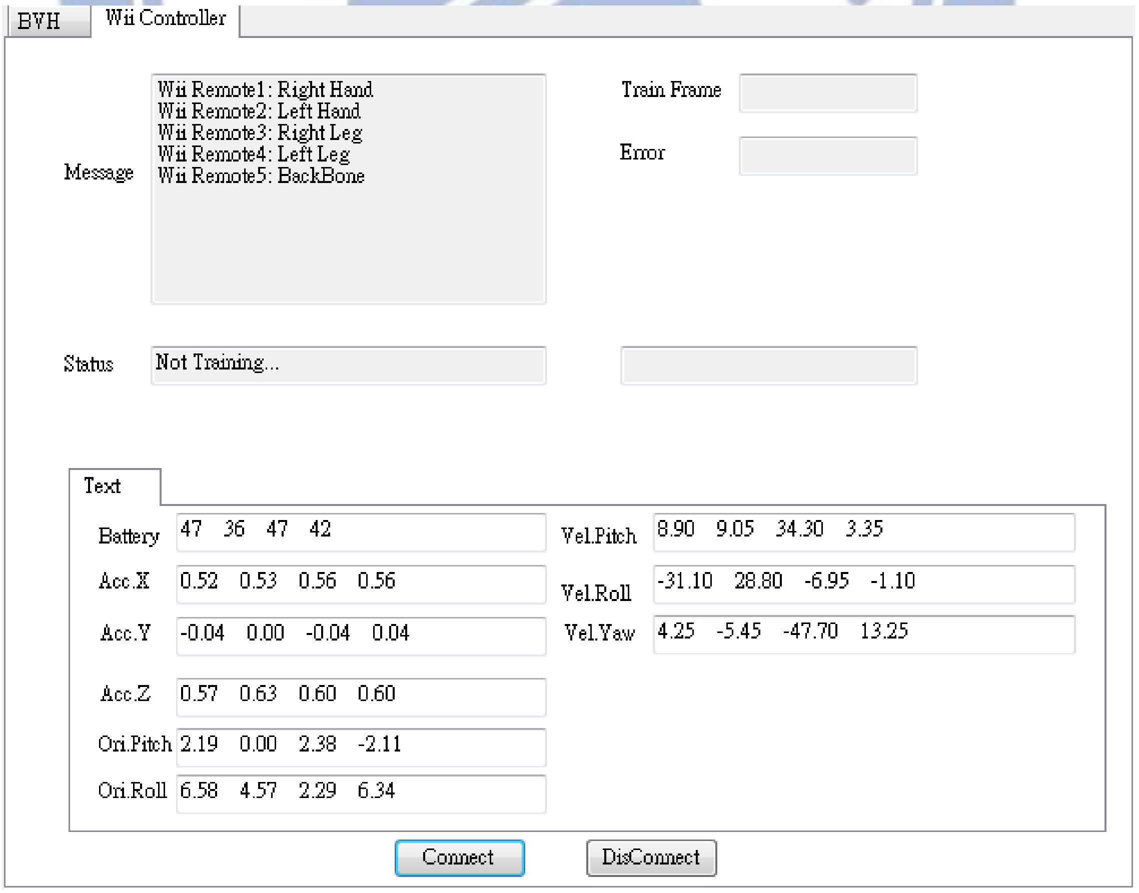
☑ Lock_Root ☑ Lock_Rotation

Construct OLNG

Start Prediction

Total WiiRemotes	1			climb	13
				jump	6
				lay	5
				lifting	7
				boxing	3
Motion Clip Length	30	KDtree Threshold	0.030	sit	13
Record Frame	4	State Number	8	upstair	0
Control Term	1.0	Prior Term	5.0	walk	6
Pose Term	0.6	Motion Term	0.2		
Smooth Term	0.2	Play Test Result		Replay Clip Number	1
		Post-Production		Replay Speed	33

Figure 6.2 Tab Page of BVH


BVH | Wii Controller

Message:
Wii Remote1: Right Hand
Wii Remote2: Left Hand
Wii Remote3: Right Leg
Wii Remote4: Left Leg
Wii Remote5: BackBone

Train Frame

Error

Status: Not Training...

Text

Battery	47 36 47 42	Vel.Pitch	8.90 9.05 34.30 3.35
Acc.X	0.52 0.53 0.56 0.56	Vel.Roll	-31.10 28.80 -6.95 -1.10
Acc.Y	-0.04 0.00 -0.04 0.04	Vel.Yaw	4.25 -5.45 -47.70 13.25
Acc.Z	0.57 0.63 0.60 0.60		
Ori.Pitch	2.19 0.00 2.38 -2.11		
Ori.Roll	6.58 4.57 2.29 6.34		

Connect | DisConnect

Figure 6.3 Tab Page of Wii Controller

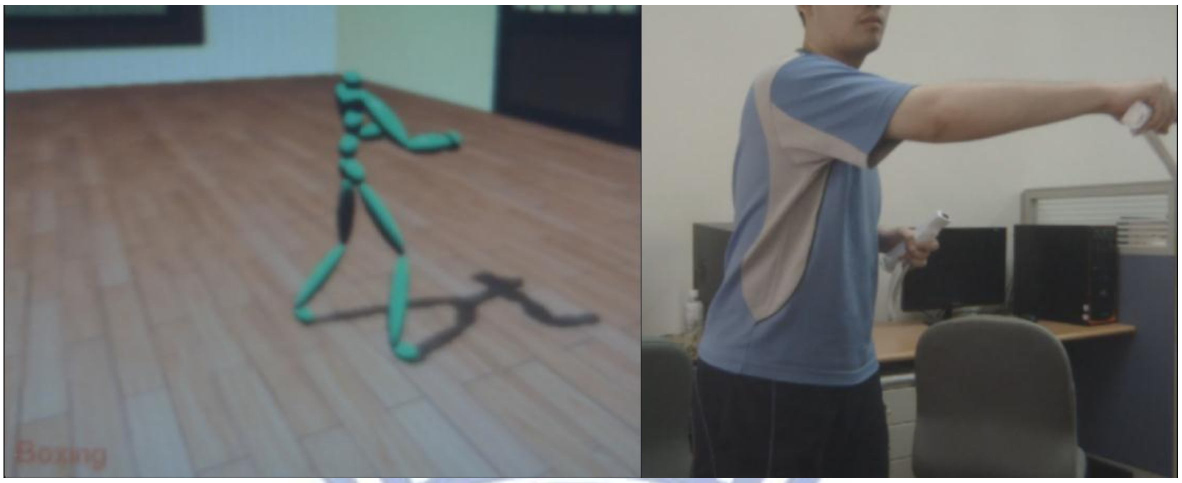


Figure 6.4 Reconstructing boxing motion

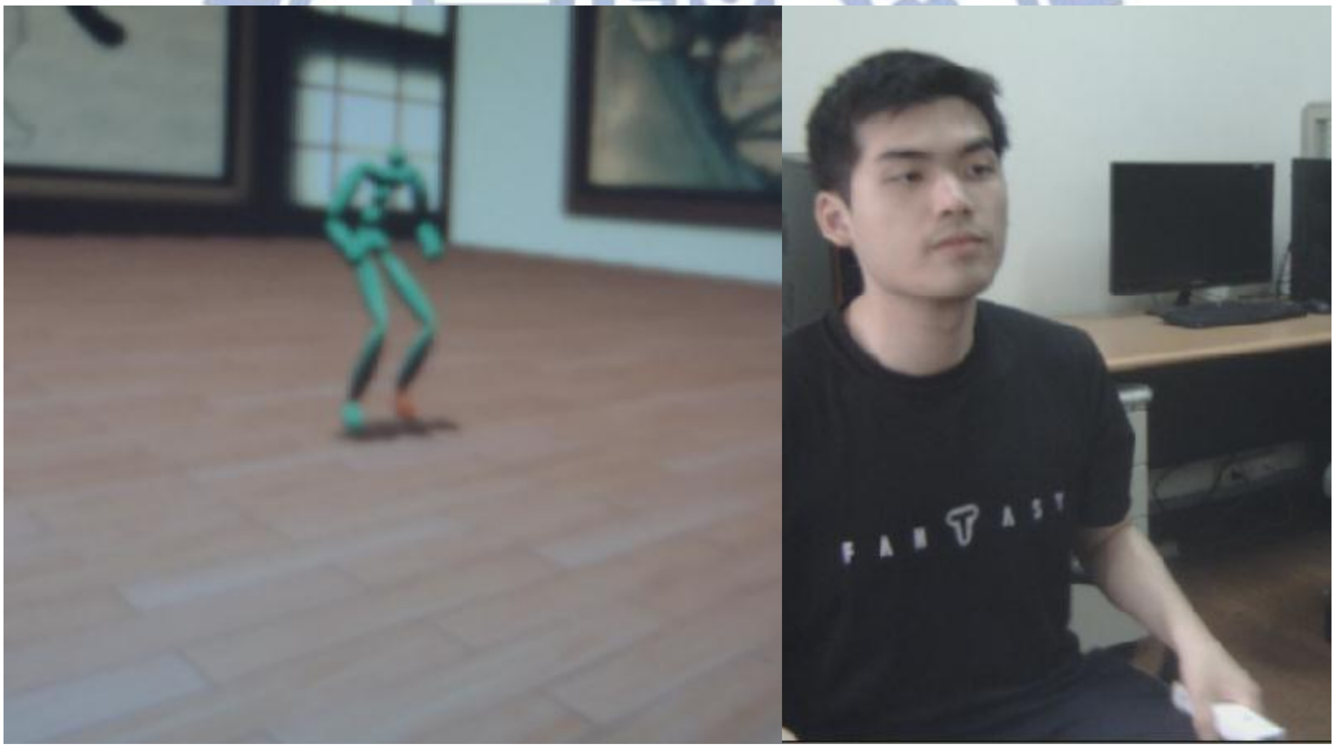


Figure 6.5 Reconstructing sitting motion

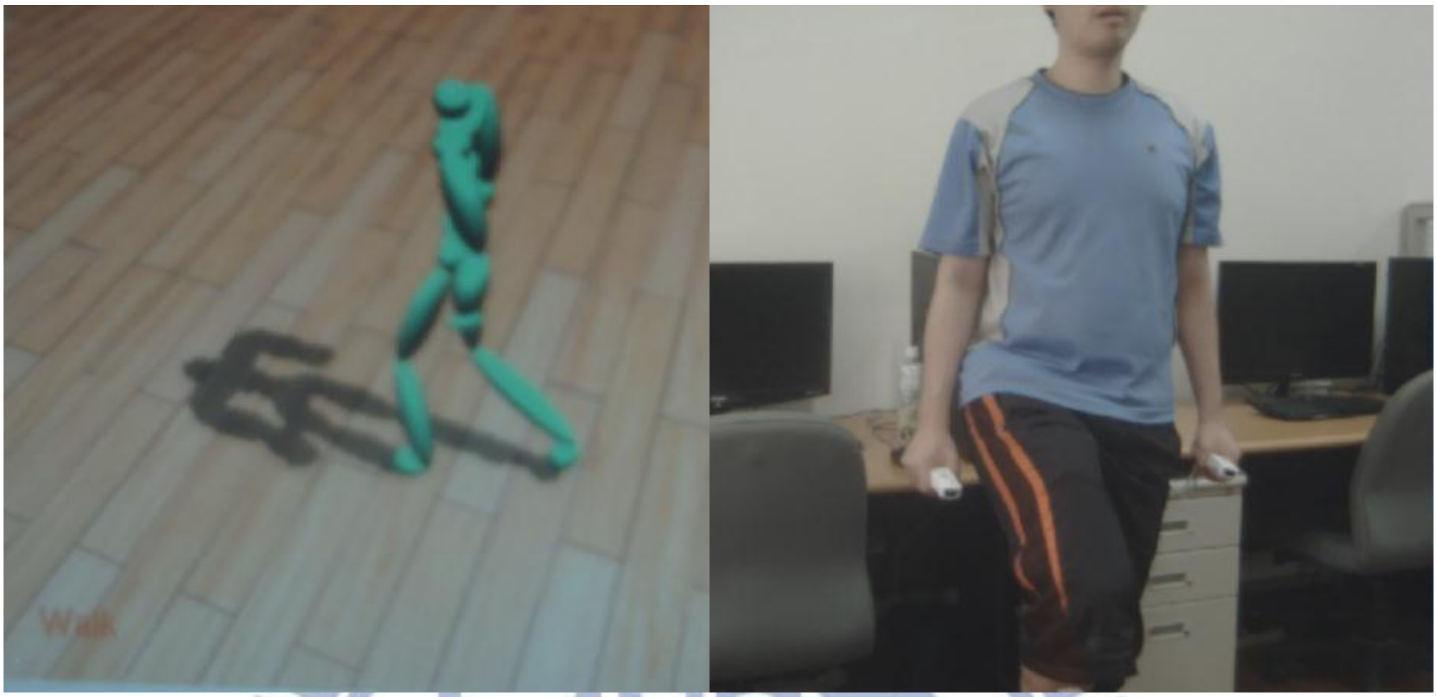


Figure 6.6 Reconstructing walking motion

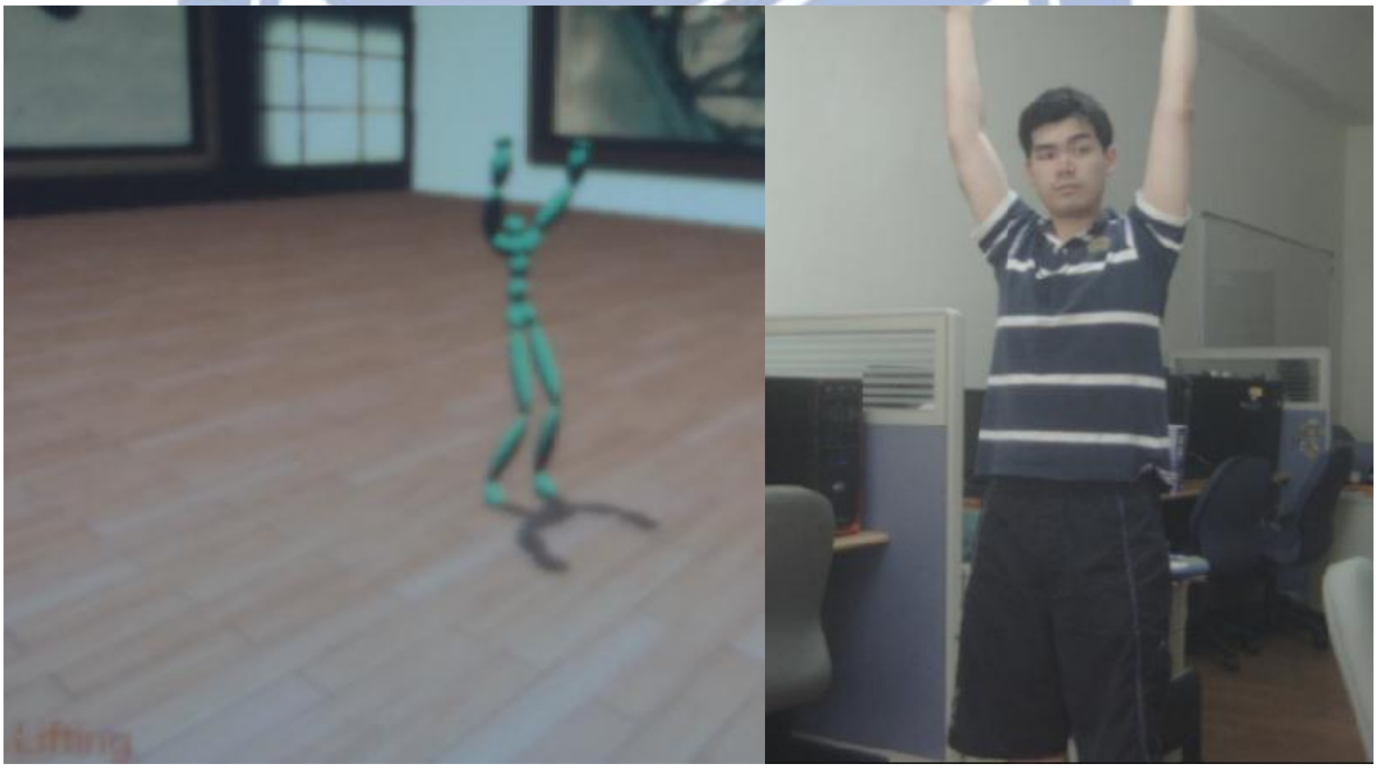


Figure 6.7 Reconstructing lifting motion

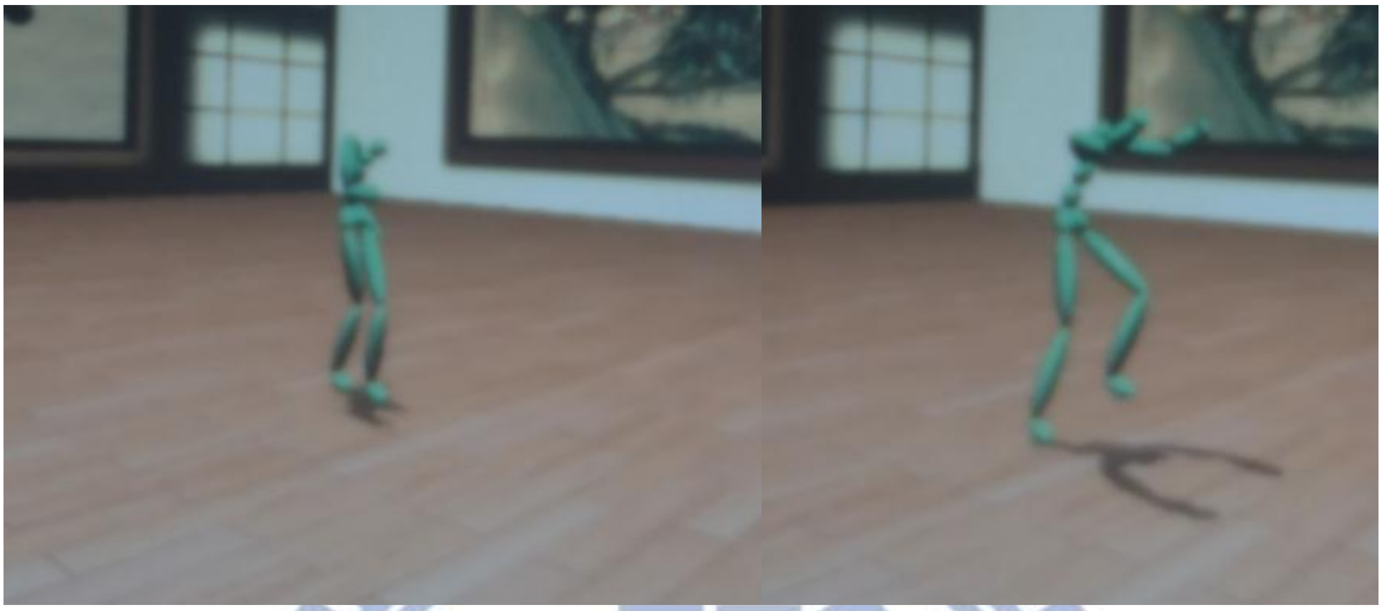


Figure 6.8 Reconstructing motion by *Online Lazy Neighborhood Graph*

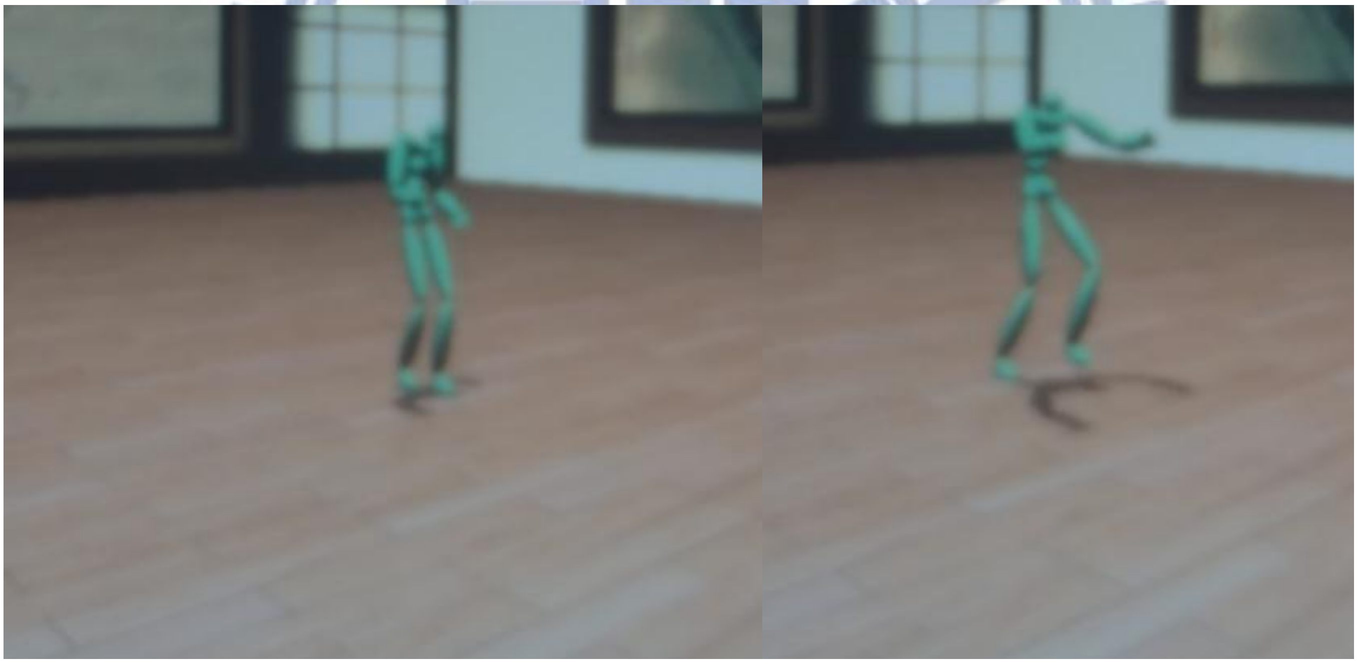


Figure 6.9 Reconstructing motion with *Online Lazy Neighborhood Interpolation Graph*

As shown in Figure 6.8 and Figure 6.9, we can see the difference between *Online Lazy Neighborhood Graph* and our *Interpolated Clip-based Online Lazy Neighborhood Graph*. The former looks unnatural because its transition would not be smooth and rapid. The pose of Figure 6.9 is convergence because we synthesize a variety of similar motion and make the result natural. Finally, we use low-pass filter to diminish unnatural full-body motion.

Our accuracy is shown in Table 6.1. We let users see the result video and point out the inaccurate sequences. Then, we calculate the proportion of accurate clips to total clips. Finally, the accuracy is almost over 80%. We can say our approach provides reliable result.

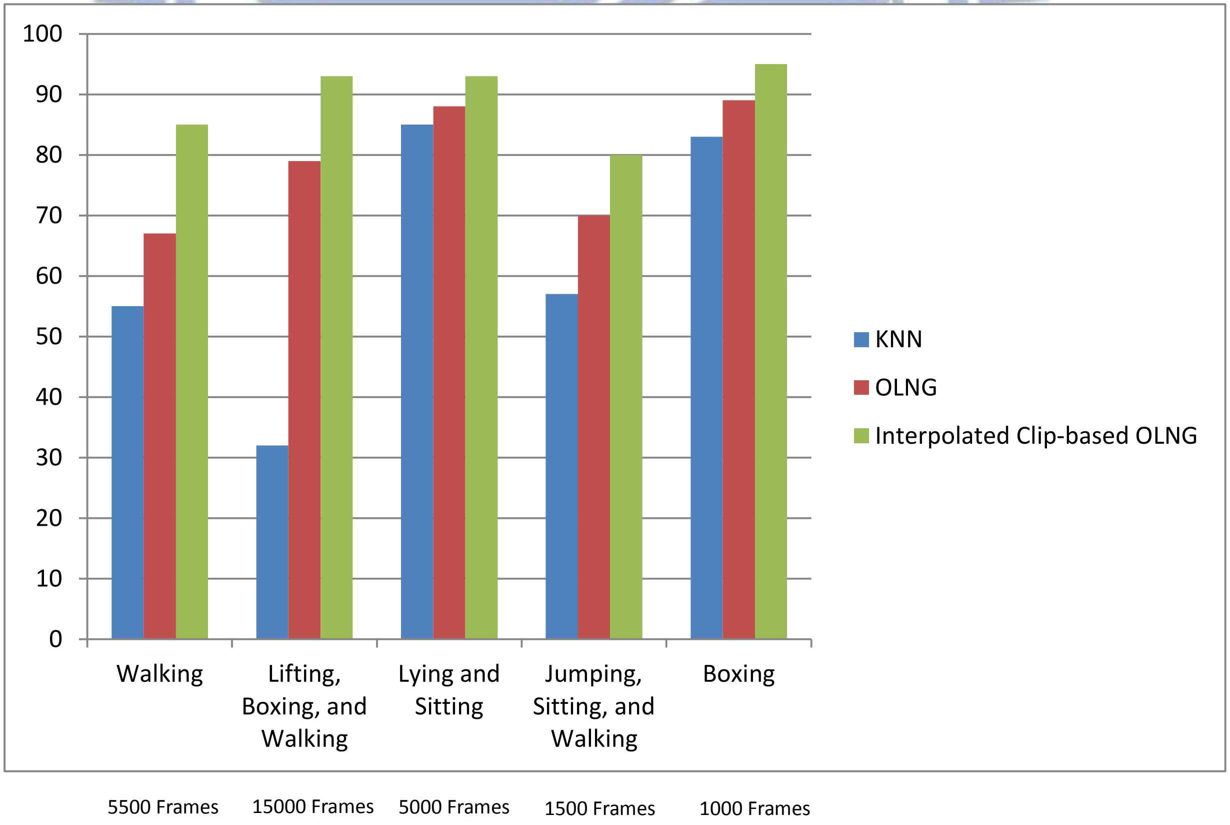


Table 6.2 Accuracy of our approach

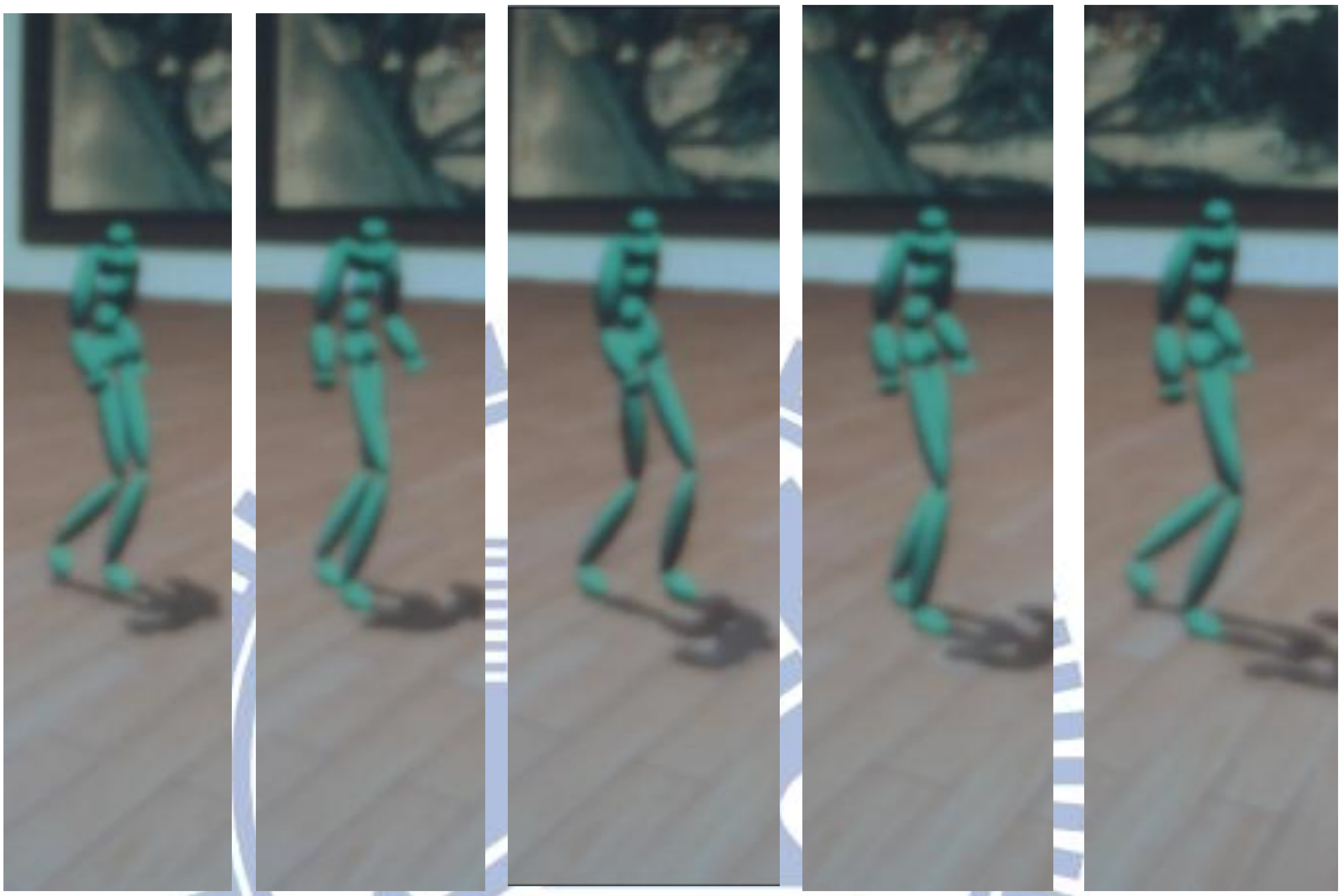


Figure 6.10 A sequences of walking

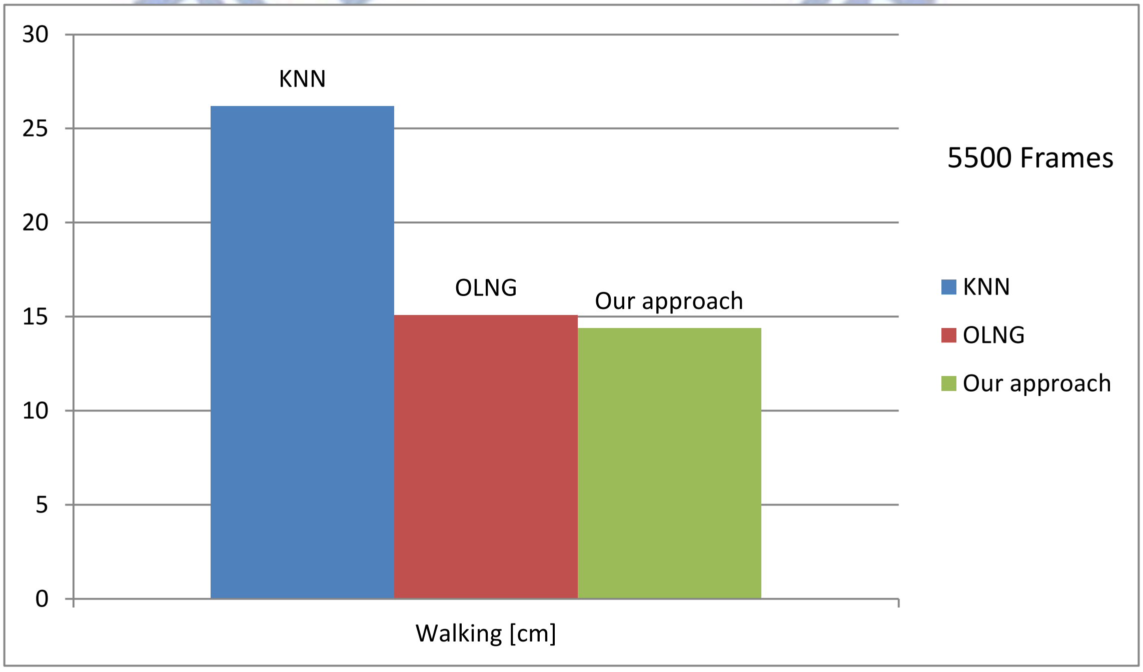
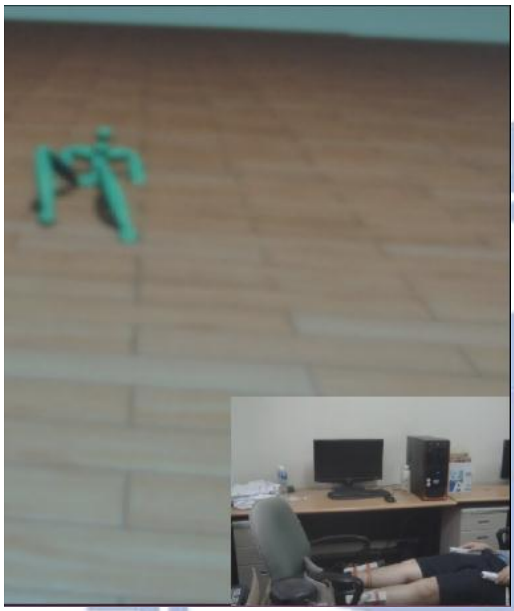
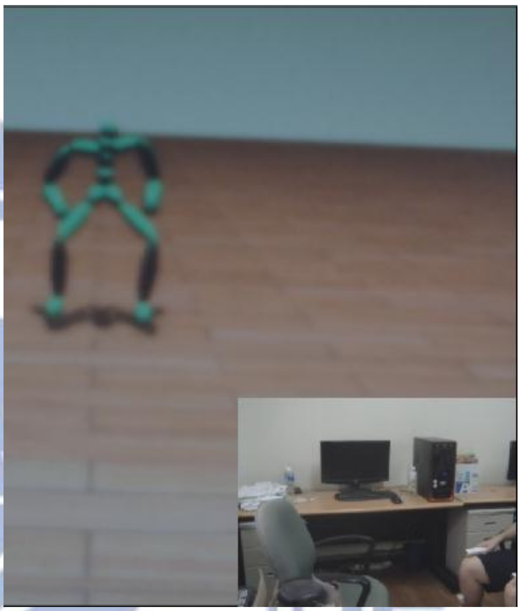


Table 6.3 Ground truth BVH with random noise(-0.01 ~ 0.01 m/s^2)





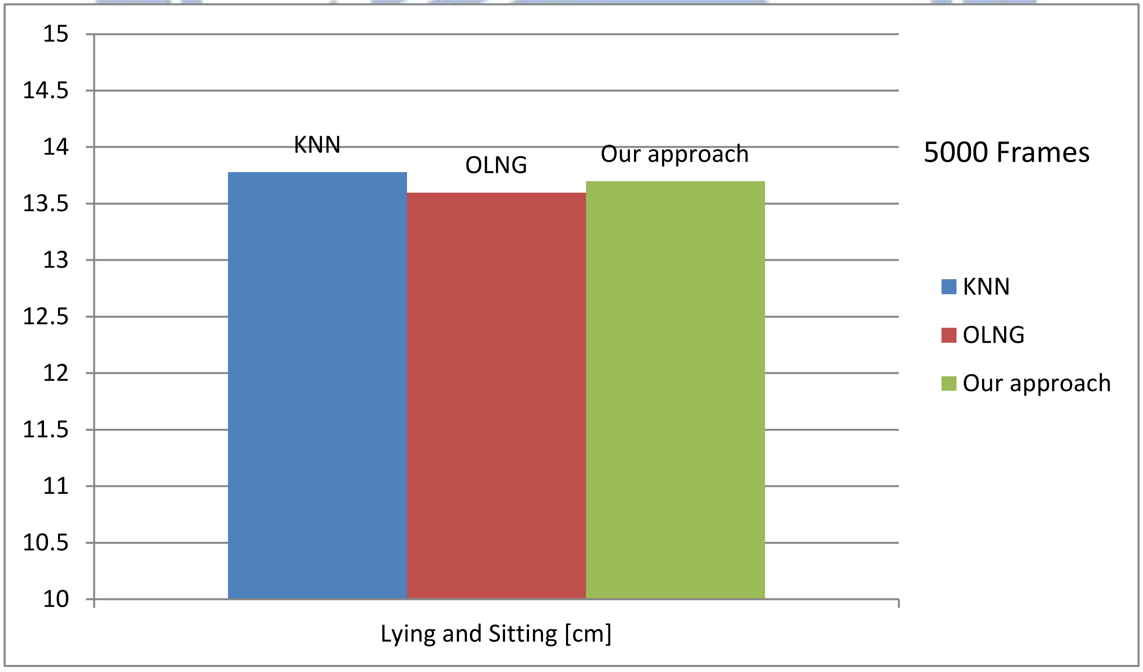
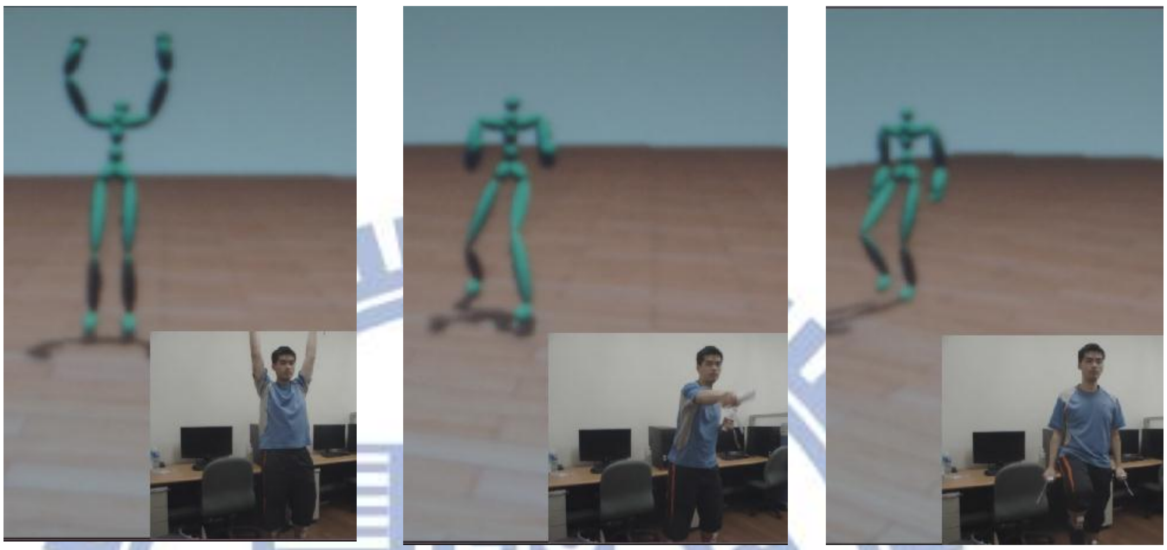


Table 6.4 Ground truth BVH with random noise(-0.01 ~ 0.01 m/s^2)



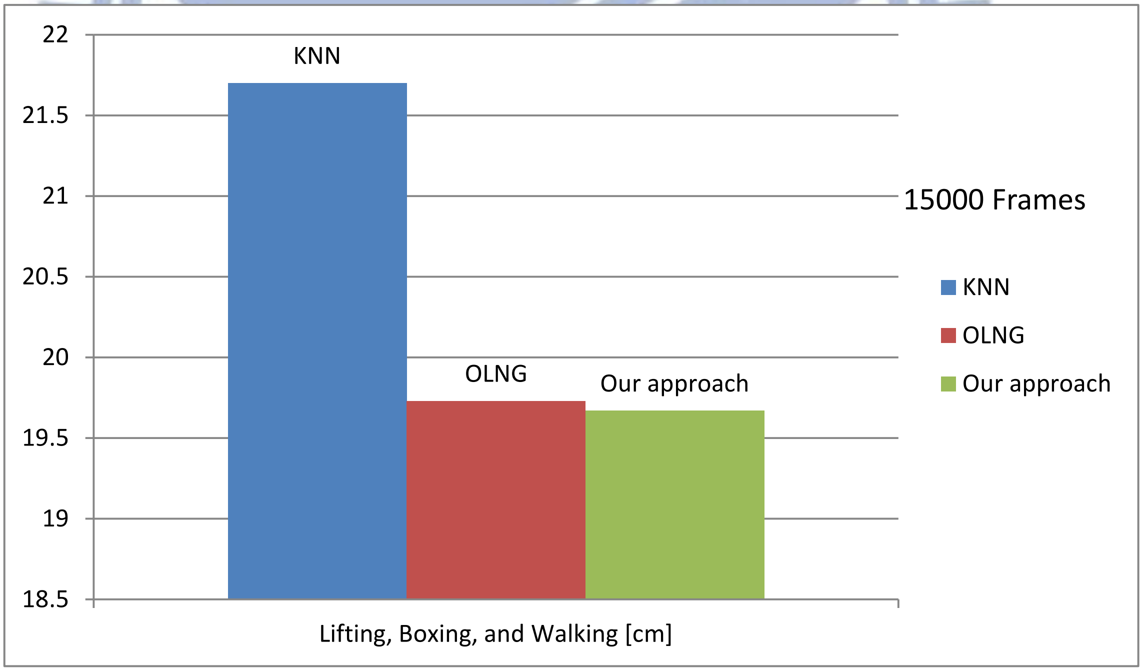


Table 6.5 Ground truth BVH with random noise(-0.01 ~ 0.01 m/s^2)

Table 6.3, 6.4 and 6.5 are the average reconstruction errors of our system. We can find that our approach is generally more accurate than *Online Lazy Neighborhood Graph*. We attempt to use the training data which is not explicitly specified in the given database as our input. In addition, to simulate the situation of motion capture device with noise, we add noise into our input. In the Table 6.4, because sitting is a static pose, there are a small amount of severe motion transitions. Therefore, the differences of three approaches are not obvious. In general, our approach is less reconstruction errors and less inaccurate sequence, compared with the other two methods.

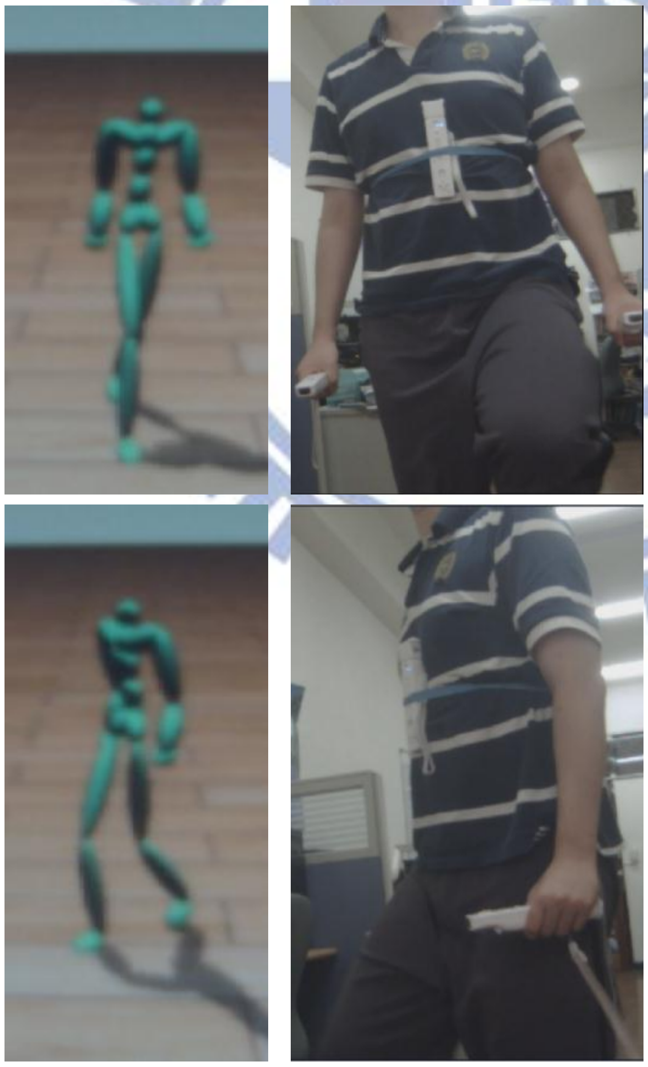


Figure 6.11 We can use the fifth Wii Remote to detect the orientation of user

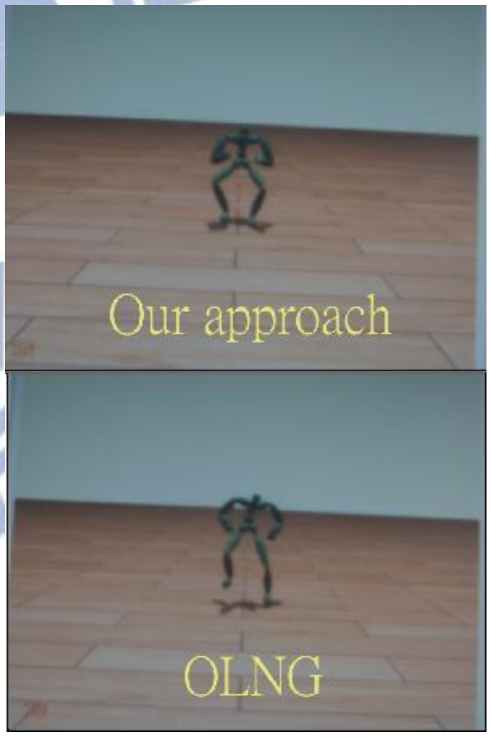


Figure 6.12 The difference between our approach and *Online Lazy Neighborhood Graph*

Because of the limitation of hardware, we use the CMU database as input to simulate the experiment with 2 to 10 sensors. Table 6.6 is the sensor joint which we defined.

	Sensor joints
2 Sensor	Right hand, Right foot
4 Sensor	Hands, Foots
6 Sensor(upper)	Hands, Foots, Clavicles
6 Sensor(lower)	Hands, Foots, Hip joints
8 Sensor	Hands, Foots, Clavicles, Hip joints
10 Sensor	Hands, Foots, Clavicles, Hip joints, Radiuses

Table 6.6 Sensor joints

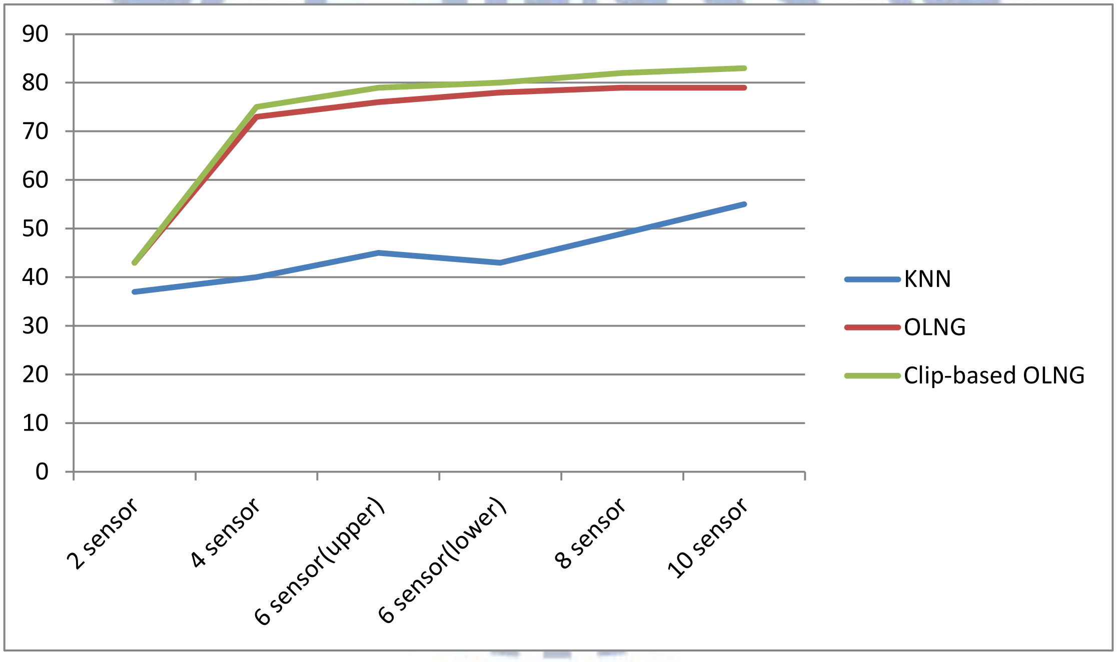


Figure 6.13 Accuracy of sensor number with noise(-0.01 ~ 0.01 m/s^2)

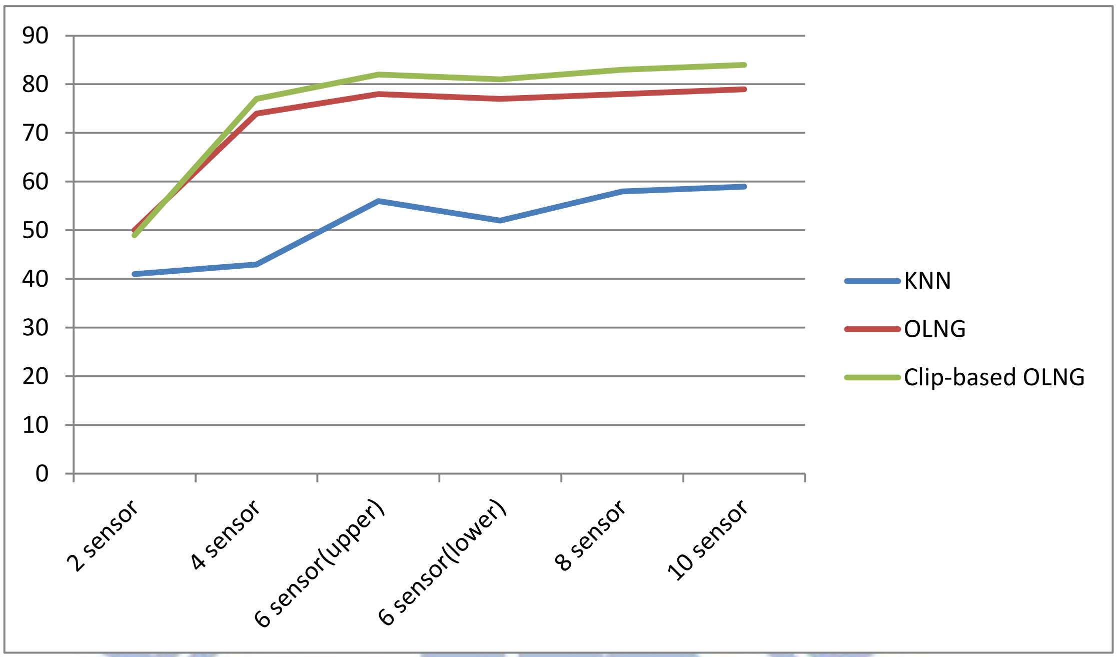


Figure 6.14 Accuracy of sensor number without noise

We can infer that the more sensors, the result is more accurate.

We also change the performer to play motion and prove our approach is general. Our experimental result is as shown in Figure 6.17. We can infer our approach is still above 70% accuracy with different actor.

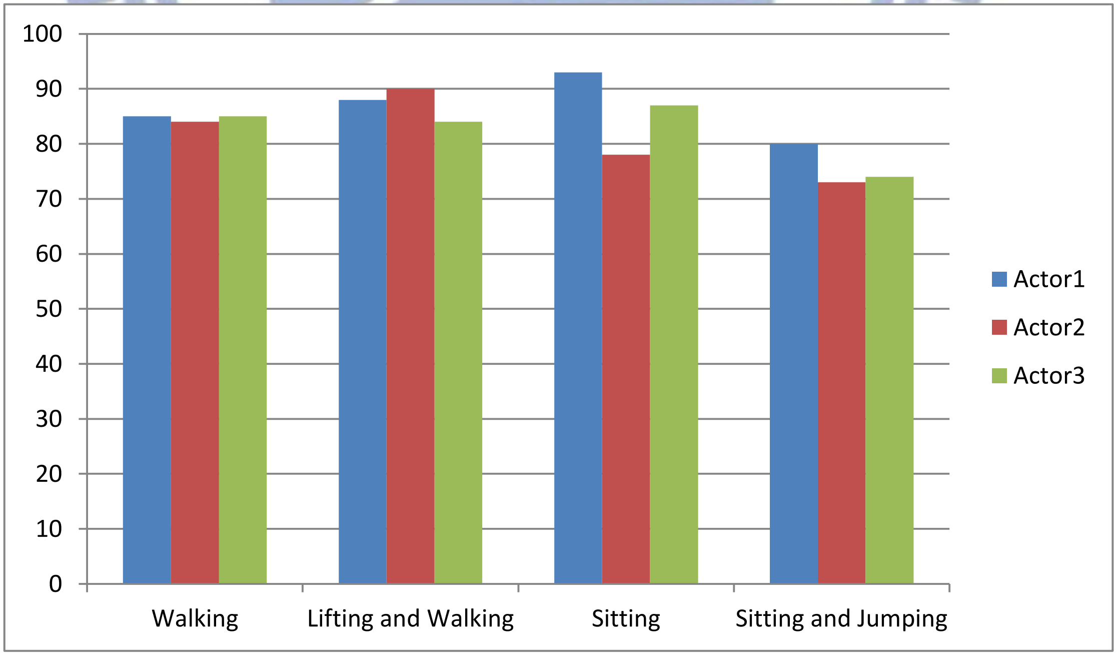


Table 6.7 Accuracy of different actor

Moreover, we try to take the accelerations of half body to our system for shorting the cost time. However, we only have the acceleration of two sensors, the system cannot determine the motion transition easily. The accuracy would be lower when rapid motion transition as shown in Table 6.8.

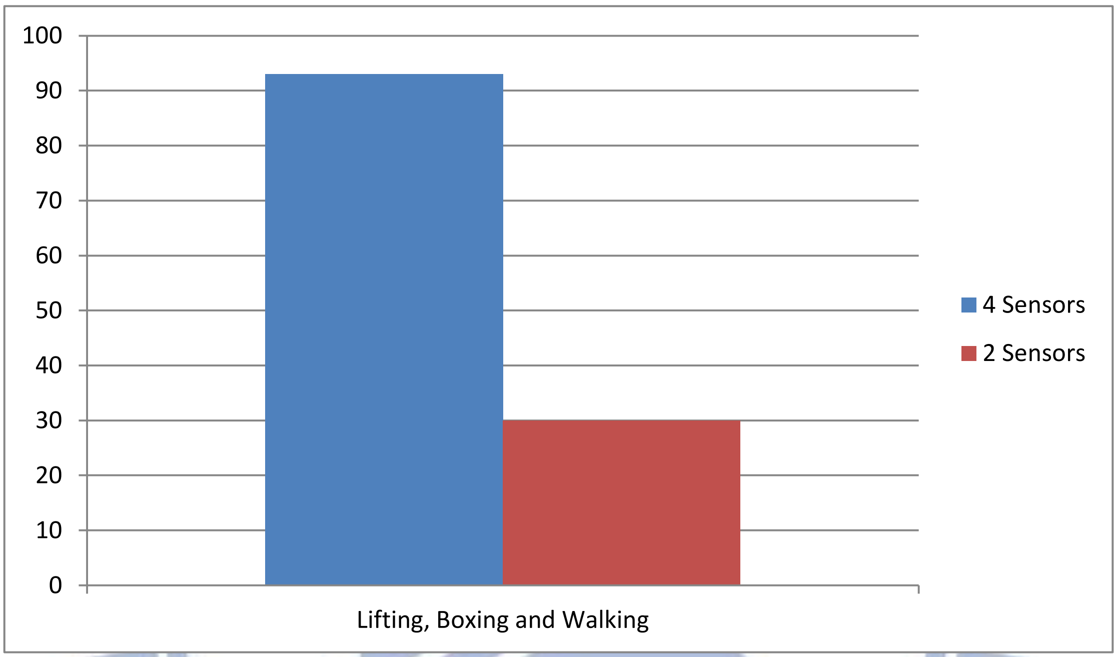


Table 6.8 Accuracy of 4 sensors and 2 sensors

Chapter 7

Conclusion

Our system is performed on a desktop with Intel® Core™2 Duo CPU, 4GB main memory, and NVIDIA GeForce 9800 GT graphic card. In our system, we successfully improve *Online Lazy Neighborhood Graph* with the concept of motion blending and using clip as operating unit. By constructing kd-tree, we can efficiently search the match training data in a large database. Then, we receive the newest data from Wii Remote to update *Clip-based Online Lazy Neighborhood Graph* and find the highest priority of candidate motion clip as our result. We solve the jitter problem by using interpolated and weighted combining motions to achieve rapid motion transition. The reconstruction motion is reliable and capable of handling variations that are not explicitly specified in the given database.

Wii Remote is as our sensing device whose advantage is portable, does not hinder daily behavior and adaptable in constraint environments. Finally, our approach is able to monitor user's motion and provide reliable accuracy and natural synthesized motion. Although acceleration data of motions contains less information, we use the concept of *motion field* to extend *Online Lazy Neighborhood Graph* and overcome its limitation to provide the reliable accuracy like other techniques using high-cost devices.

Reference

[AI06] Andoni, A. and Indyk, P. “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions” Foundations of Computer Science, 2006. Pages 459-468.

[AFO03] Arikan, O., Forsyth, D. and O’Brien, J. F. “Motion synthesis from annotations” ACM Transactions on Graphics, Vol 22 Issue 3, July 2003, Pages 402-408.

[BHG93] Badler, N. I., Hollick, M. J. and Granieri, J. P. “Real-time control of a virtual human using minimal sensors” Presence: Teleoperators and Virtual Environments Vol 1, Pages 82-86, 1993.

[CH05] Chai, J. and Hodgins, J. K. “Performance animation from low-dimensional control signals” ACM Transactions on Graphics, Vol 24 Issue 3, 2005, Pages 686-696.

[CMUMocap] CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu/

[EGW05] Ernst, D., Geurt, P., Wehenkel, L. and Littman, L. “Tree-based batch mode reinforcement learning” Journal of Machine Learning Research Vol 6, Page 503-556, Apr 2005.

[FKY08] Feng, W. -W., Kim, B. -U. and Yu. Y. “Real-time data driven deformation using kernel canonical correlation analysis” ACM Transactions on Graphics, Vol 27, 2008, Pages 91:1-91:9.

[HBL11] Ha. S., Bai, Y. and Liu, C. K “Human motion reconstruction from force sensors” Eurographics/ACM SIGGRAPH Symposium on

Computer Animation 2011.

[InvenSense] MEMS Gyro | Gyroscope | Motion Plus | Processing-InvenSense Home. http://invensence.com/index.html

[KCH10] Kelly, P., Conaire, C. O., Hodgins, J. and O'Conner, N. E. "Human motion reconstruction using wearable accelerometers" (poster) Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2010.

[KTW10] Kruger, B., Tautges, J., Weber, A. and Zinke, A. "Fast local and global similarity searches in large motion capture databases" Eurographics / ACM SIGGRAPH Symposium on Computer Animation 2010, Pages 1-10.

[LWB10] Lee, Y., Wampler, K., Bernstein, G., Popovic, J. and Popovic, Z. "Motion fields for interactive character locomotion" ACM Transactions on Graphics – Proceedings of ACM SIGGRAPH Asia 2010, Volume 29 Issue 6, December 2010, Article No. 138.

[Nintendo10] Nintendo of America Inc. Headquarters are in Redmond, Washington. http://www.nintendo.com/wii

[PhaseSpace10] PhaseSpace motion capture. Accessed March 10th, 2010. http://www.phasespace.com

[S10] Smith, R. Open dynamic engine, May 2010. http://ode.org/ode.html

[SH08a] Slyper, R. and Hodgins, J. "Action capture with accelerometers" In proceedings of the 2008 Eurographics/ACM SIGGRAPH Symposium on Computer Animation.

[SH08b] Siratori, T. and Hodgins, J. K. “Accelerometer-based user interfaces for the control of a physically simulated character” ACM Transactions on Graphics, Vol 27 2008, Pages 123:1-123:9.

[SKL07] Sok, K. W., Kim, M. and Lee, J. “Simulating biped behaviors from human motion data” ACM Transactions on Graphics, Vol 26 Issue 3, July 2007, Article No. 107.

[TWC09] Tournier, M., Wu, X., Courty, N., Arnaud, E. and Reveret, L. “ Motion compression using principal geodesics analysis” Computer Graphics Forum Vol 28 Issue 2, Pages 355-364. EUROGRAPHCIS 2009.

[TZK11] Tautges, J., Zinke, A., Kruger, B., Baumann, J., Webber, A., Helten, T., Muller, M., Seidel, H. P. and Eberhardt, B. “Motion reconstruction using sparse accelerometer data” ACM Transactions on Graphics, Vol. 30 Issue 3, May 2011, Article No. 18.

[WiiYourself] WiiYourself-gl.tter’s native C++ Wiimote library. http://wiiyourself.gl.tter.org/

[YL10] Ye, Y. and Liu, K. “Synthesis of responsive motion using a dynamic model” Computer Graphics Forum Vol 29 Issue 2. EUROGRAPHCIS 2010.

[ZMC05] Zordan, V. B., Majkowska, A., Chiu, B. and Fast, M. “Dynamic response for motion capture animation” ACM Transactions on Graphics, Vol. 24 Issue 3, 2005, Pages 697-701