Single View Human Pose Tracking

by

Zhenning Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

Abstract

Recovery of human pose from videos has become a highly active research area in the last decade because of many attractive potential applications, such as surveillance, non-intrusive motion analysis and natural human machine interaction. Video based full body pose estimation is a very challenging task, because of the high degree of articulation of the human body, the large variety of possible human motions, and the diversity of human appearances.

Methods for tackling this problem can be roughly categorized as either discriminative or generative. Discriminative methods can work on single images, and are able to recover the human poses efficiently. However, the accuracy and generality largely depend on the training data. Generative approaches usually formulate the problem as a tracking problem and adopt an explicit human model. Although arbitrary motions can be tracked, such systems usually have difficulties in adapting to different subjects and in dealing with tracking failures.

In this thesis, an accurate, efficient and robust human pose tracking system from a single view camera is developed, mainly following a generative approach. A novel discriminative feature is also proposed and integrated into the tracking framework to improve the tracking performance.

The human pose tracking system is proposed within a particle filtering framework. A reconfigurable skeleton model is constructed based on the Acclaim Skeleton File convention. A basic particle filter is first implemented for upper body tracking, which fuses time efficient cues from monocular sequences and achieves real-time tracking for constrained motions. Next, a 3D surface model is added to the skeleton model, and a full body tracking system is developed for more general and complex motions, assuming a stereo camera input. Partitioned sampling is adopted to deal with the high dimensionality problem, and the system is capable of running in near real-time. Multiple visual cues are investigated and compared, including a newly developed explicit depth cue.

Based on the comparative analysis of cues, which reveals the importance of depth and good bottom-up features, a novel algorithm for detecting and identifying endpoint body parts from depth images is proposed. Inspired by the shape context concept, this thesis proposes a novel Local Shape Context (LSC) descriptor specifically for describing the shape features of body parts in depth images. This descriptor describes the local shape of different body parts with respect to a given reference point on a human silhouette, and is shown to be effective at detecting and classifying endpoint body parts. A new type of interest point is defined based on the LSC descriptor, and a hierarchical interest point selection algorithm is designed to further conserve computational resources. The detected endpoint body parts are then classified according to learned models based on the LSC feature. The algorithm is tested using a public dataset and achieves good accuracy with a 100Hz processing speed on a standard PC.

Finally, the LSC descriptor is improved to be more generalized. Both the endpoint body parts and the limbs are detected simultaneously. The generalized algorithm is integrated into the tracking framework, which provides a very strong cue and enables tracking failure recovery. The skeleton model is also simplified to further increase the system efficiency. To evaluate the system on arbitrary motions quantitatively, a new dataset is designed and collected using a synchronized Kinect sensor and a marker based motion capture system, including 22 different motions from 5 human subjects. The system is capable of tracking full body motions accurately using a simple skeleton-only model in near real-time on a laptop PC before optimization.

Acknowledgements

I would like to thank my supervisor Dr. Dana Kulić for her great support, guidance, and expertise throughout this project. I am truly grateful for all the advices and suggestions given in our various meetings, email correspondences, and for all the hours spent editing my writing. I really appreciate her great responsibility in supervising me and the great patience in helping me.

I am also indebted to her for funding my travels both domestic and abroad so that I was able to gain exposure to both national and international computer vision and robotics research communities. Her approval for me to do research interns at other institutions is also greatly appreciated.

Special thank goes to Yun-qian Miao and my lab mates for all their support and great discussions. I have learned so much from you. I would also say thank you to the participants for volunteering in helping me to collect the motion data, which is essential for the system verification. I would also thank the researchers who set up the Carnegie Mellon University Graphics Lab Motion Capture dataset and the Stanford Time-of-Light Motion Capture dataset, which are used in the earl-stage experiments.

Lastly, I would like to thank my family and friends for their unwavering support and encouragement to help me to overcome all the difficulties and to find my way. This thesis would not have been possible without any of you.

Dedication

This thesis is dedicated to this 43 months of time and every single one involved. What I have learned is way more than what appears here.

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

AMC	Acclaim Motion Capture
ASF	Acclaim Skeleton Files
BME	Bayesian Mixture Expert
CMU MOCAP	Carnegie Mellon University Graphics Lab Motion Capture
DOF	Degree of Freedom
EM	Expectation-Maximization
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
ICP	Iterative Closest Point
IR Point	Interest Reference Point
LSC	Local Shape Context
LSW	Local Sliding Window
MAP	Maximize a Posteriori Estimation
MC	Monte Carlo
MOCAP	Motion Capture
NUI	Natural User Interacting

PCA	Principle Component Analysis
PDF	Probability Density Function
RGB	Red Green Blue
RVM	Relevance Vector Machine
SFS	Shape from Silhouette
SIR	Sampling Importance Resampling
SIR	Sequential Importance Re-sampling
SIS	Sequential Importance Sampling
SMD	Stochastic Meta Descent
TOF	Time-of-Flight

Symbols

α	Weight for the edge cue
α_x	Offset angle about x axis for ASF model
α_y	Offset angle about y axis for ASF model
α_z	Offset angle about z axis for ASF model
β	Weight for the distance cue
β_x	Joint angle rotation about x axis
β_y	Joint angle rotation about y axis
β_z	Joint angle rotation about z axis
$\mathbb{N}$	The set of natural numbers
$\mathbb{R}$	Real number space
ϕ_x	Roll angle
ϕ_y	Pitch angle

ϕ_z	Yaw angle
A	Base of a weighting function
$b_{ij'}$	Template blue intensity of the jth sampled point in the ith body part in a partition
b_{ij}	Binary value for the jth pixel along the projected edge
b_{ij}	Observed blue intensity of the jth sampled point in the ith body part in a partition
d^C	Distance of colour cue
d^{De}	Distance of depth cue
d^E	Distance of edge cue
d_{ij}	Depth difference for the jth point of the ith body part in a partition
d_i	Distance for the ith particle
$g_{ij'}$	Template green intensity of the jth sampled point in the ith body part in a partition
g_{ij}	Observed green intensity of the jth sampled point in the ith body part in a partition
L	Offset angle rotation for ASF model
L_h	Offset angle rotation matrix for left humerus
M	Transformation matrix
M_h^c	Transformation matrix from left humerus to left clavicle frame
M_i	Total number of sampled points in body part i in a partition
m_i	Number of pixels of the projected edges for the ith body part
N	Number of body parts in a partition
p^c	The point in the left clavicle frame
p^h	A point in the left humerus frame
P^i	A point in local frame i
p^{world}	The point in world frame

R	Joint angle rotation matrix
R_h	Joint angle rotation matrix for left humerus
$r_{ij'}$	Template red intensity of the jth sampled point in the ith body part in a partition
r_{ij}	Observed red intensity of the jth sampled point in the ith body part in a partition
R_x	Rotation matrix about x axis
R_y	Rotation matrix about y axis
R_z	Rotation matrix about z axis
T	Translation matrix
T_h^c	Translation matrix from left clavicle to left humerus
w^B	Weight of blob distance cue
w^B	Weight of ground distance cue
w^C	Weight of colour cue
w^{De}	Weight of depth cue
w^E	Weight of edge cue
w_i	Weight for the ith particle
w_i^D	Weight for the ith particle from distance cue
w_i^E	Weight for the ith particle from edge cue
w_{li}	Final weight for the lth particle in the ith partition
x_c	Translation along x axis from left clavicle to left humerus
y_c	Translation along y axis from left clavicle to left humerus
z_c	Translation along z axis from left clavicle to left humerus

Chapter 1

Introduction

1.1 Motivation

The recovery of human pose from videos has become a highly active research area in the last decade due to the fast developments in hardware, computer vision algorithms and machine learning algorithms, as well as the very attractive potential applications, such as surveillance, non-intrusive motion analysis and natural human machine interaction [1, 2, 3, 4, 5]. However, the design of an accurate, efficient and robust algorithm for full body pose recovery is still a very challenging task.

The human body is a highly articulated system. A realistic full body human model needs to contain at least 25 Degrees of Freedom (DOF) to model gross body movements [6]. The motion range for each joint is wide, with at least 90 degrees range in each joint for a healthy person. If motion type is not constrained, this leads to a huge span of possible motions. Occlusions caused by folding body parts, *e.g.* crossing arms in front of torso, may appear frequently in daily movements. In addition, the variation in appearance between different individuals can be significant, due to different body sizes, different hair styles, different clothing, *etc.*

Several commercial vision based Motion Capture (MOCAP) systems have already been used in research, as well as film making, game designing and sports training industries [7, 8, 9, 10, 11, 12]. Accuracy is the main design consideration for such systems and high frame rates are usually preferable. Most of them are based on markers that are attached to human body, either active [10, 11, 12] or passive [7, 8, 9]. A multi-camera (typically 8 to 12) setup is normally required, with expensive high speed cameras fixed to permanent locations.

Calibration is needed prior to each capture to locate the world frame and all the cameras. After each capture session, post processing needs to be performed to remove phantom markers and to interpolate for the missing ones. These tedious preparations, expensive equipment and complicated post processing severely restrict their application scenarios, especially for everyday use. With the recent advances in computer vision, commercial marker-less motion capture systems have emerged in the market [13]. However, a multi-camera setup and a special studio environment are still required.

As a result, marker-less human pose recovery from videos has attracted extensive research attention in the recent years, from both computer vision and robotics communities. The approaches can generally be classified into two categories according to the underlying methodologies, *i.e.* discriminative approaches and generative approaches. A straightforward idea for solving the problem is to find a mapping between the image feature space and the human pose space, and this naturally leads to the discriminative approach. The estimation is usually achieved by either searching for nearest neighbours within a large set of exemplars [14, 15], or regressing a mapping function from a large set of training data [16, 17]. The search based algorithms can avoid the training step, but suffer from lower testing efficiency when the exemplar set is large [15]. The regression based methods can be very efficient during testing, but the training process is more computationally expensive. The regression based methods also need to deal with the high-dimensional, highly non-linear and possibly multi-valued mapping between the two spaces [18, 17]. The discriminative approaches have the potential of generalizing to different body shapes, but this will further increase the size of the exemplar or training dataset [19]. They can work on individual images, but the accuracy and motion range are inherently limited by the training data.

In contrast to discriminative approaches, generative approaches [20, 21] adopt explicit human models of varying complexity. The model can be as simple as a stick model [22, 23], or as complicated as a deformable surface model with underlying skeleton [24, 25, 21]. This type of methods makes use of the inherent kinematic structure of human body, and works under the temporal continuity assumption of human motion. The problem is usually solved by optimizing a deterministic cost function [26], Maximizing a Posterior (MAP) likelihood function [25], or estimating the posterior distribution [6]. These approaches can also be combined within a multi-layer framework [21]. Local optimization algorithms are adopted when the objective function is formulated analytically [26, 27, 25], while sampling based methods, such as particle filters [6] and particle swarm optimization [28, 29], are applied otherwise. A range of visual cues have been explored for tracking, mainly low-level features. Although the explicit modelling of the human body and the optimization nature of these approaches entail higher computational cost, pose recovery for arbitrary motion can be achieved with high accuracy. In particular, Bayesian filtering provides a rigorous framework for estimating the posterior distribution of hidden variables. For the video based human motion tracking problem, the human body is usually modelled as a skeleton described by the joint angles. The system cannot be assumed to be linear with Gaussian noise because of the articulation of the human skeleton, the complexity of human dynamics, and the projection process during image formation. The particle filter, also known as the CONDENSATION algorithm is well suited for this context [30].This motivates us to consider generative approaches, and build our work within a particle filter framework.

A major problem with the particle filter, however, is the curse of dimensionality [31]. The number of particles required for successful tracking increases exponentially with the increase of the number of DOF in the human model. Machine learning techniques have been applied to obtain motion-specific dynamic priors in order to reduce the search space [32, 33]. Alternatively, variations of the particle filter have been proposed, such as the Annealed Particle Filter [6] and the Partitioned Particle Filter [34]. To maintain the capability of storing multiple hypotheses from a multi-modal distribution, partitioned sampling is adopted in our work. In addition, it has been demonstrated that by including stronger, higher level cues, the performance of a tracking system can be improved significantly [35]. The knowledge of the possible body part locations can not only act as a strong cue for pose inference, it can also provide the possibility of automatic initialization and tracker failure recovery. This motivates us to develop a body part detection algorithm, which is incorporated into our pose tracking framework.

Different camera setups have been explored in the existing literatures, mainly depending on the potential application scenarios. Some systems use multiple cameras [6, 24, 26, 36], while others work from a single view [32, 35, 25, 37]. In a multi-camera setup, the cameras observe the human body from different directions. As a result, the self-occlusion problem can be considerably reduced and therefore a high accuracy is easier to achieve [27]. However, this approach also entails heavier computational cost and thus makes it difficult to achieve real-time. The multi-camera setup also severely limits the portability of a system.

Human pose recovery system from a single view is usually desirable for applications such as mobile robots [38] and human machine interaction [39]. For humanoid robots, real-time motion data capture will enable the development of on-line autonomous motion learning algorithms, which is an innovative and efficient way to teach a humanoid robot human-like motions [40]. While human pose estimation using monocular image sequences has been studied intensively [32, 22, 41], the missing depth information hinders accurate and robust performance. Therefore stereo cameras have been investigated, especially in the systems designed for robot applications [35, 38, 42]. The recent advances in depth sensing provide better depth maps more efficiently, and consequently there is an increasing research effort devoted to human pose estimation from depth maps in recent years [25, 43, 19]. In this thesis, human pose tracking from a single view will be studied, including the usage of monocular colour camera, stereo camera, Time-of-Flight (TOF) depth sensor and Kinect depth sensor.

1.2 Overview

The motivating application for the work in this thesis is human pose tracking in a human machine interaction scenario. Therefore, a single-view camera setup is assumed. A particle filter framework is adopted, with partitioned sampling to track full body human motions efficiently. Different sensing modalities are explored, including monocular camera, stereo camera and depth sensors. A novel depth-based body part detection and identification algorithm is proposed, which is incorporated into the tracking framework as an effective cue. A new dataset is collected containing colour and depth videos, as well as ground truth from multi-camera motion capture.

1.2.1 Contributions

First, we present a model-based full body human pose tracking system using a particle filter. A configurable human model is constructed, including a skeleton model and an outer shape model. A basic particle filter is implemented for upper body tracking, which fuses time efficient cues from monocular image sequences. The system achieves real-time tracking for constrained motions. Then 3D surfaces are added to the human model, and a full body tracking system is developed for more general motions. Partitioned sampling is applied to deal with the high dimensionality problem, and the system is capable of running in near real-time. Multiple visual cues were investigated, including a newly developed depth cue. A comprehensive study on the relative importance of different cues is conducted, and the quantitative results are reported.

Second, we propose a novel algorithm for detecting and identifying endpoint body parts from depth images. Inspired by the shape context concept proposed by Belongie *et al.* [44], we propose a novel Local Shape Context (LSC) Descriptor specifically for describing the shape features of body parts in depth images. This descriptor describes the local shape of different body parts with respect to a given reference point on the human silhouette, and is shown to be effective at detecting and classifying endpoint body parts, while being computationally efficient. A new type of interest point is defined based on the LSC Descriptor. A hierarchical interest point selection algorithm is designed to further conserve computational resources. The detected endpoint body parts are then classified according to learned models of each class based on the LSC feature.

Finally, we describe a novel full body tracking algorithm which incorporates the body part detection algorithm. The LSC feature is generalized: not only endpoint body parts are recognized, the limbs are also detected simultaneously. The use of this new feature enables efficient and accurate tracking of full body motions by using a very simple, skeleton-only human model. A new dataset is collected for evaluating the algorithm, containing synchronized colour videos, depth videos, and ground truth captured from a marker based commercial MOCAP system. The dataset contains 22 different motions from 5 human subjects, and provides a comprehensive benchmark for algorithm evaluation.

1.2.2 Outline

The remainder of the thesis is organized as follows. In Chapter 2, related work on camera based human pose recovery will be reviewed, covering discriminative approaches, generative approaches, body part detection and recognition, integration of body part recognition with model based tracking, and recent advances utilizing the Kinect.

In Chapter 3, the fundamentals of particle filtering and partitioned sampling, which forms the basis of subsequent chapters, will be reviewed. The probabilistic system model is first described, and solved as a Bayesian estimation problem. The mathematical derivation of particle filters will be described, including the Monte Carlo method and importance sampling, the Sequential Importance Sampling (SIS) algorithm, the Sampling Importance Resampling (SIR) algorithm, and the particle filter with partitioned sampling.

In Chapter 4, a model based human motion tracking framework using particle filters is proposed. The construction of a skeleton model compatible with the Acclaim Skeleton File (ASF) convention is described, followed by the description of an upper body tracking algorithm capable of tracking constrained motions in real-time. Then the algorithm is improved in terms of the human model, the projection model, and the tracking algorithm, and is shown to achieve near real-time tracking for more complex full body motions.

In Chapter 5, a new method for fast detection and classification of endpoint body parts is proposed, based on the novel Local Shape Context (LSC) Descriptor. An efficient algorithm is developed for locating the reference interest points and classifying the points. The system is shown to achieve a high precision and recall rate with a processing speed of 100Hz on standard PC.

In Chapter 6, a human pose tracking system using monocular depth sequences is proposed, integrating the body part recognition algorithm (Chapter 5) with the partitioned particle filter framework (Chapter 4). The LSC descriptor is generalized to detect not only the endpoint body parts, but also the torso and the limbs. A simplified skeleton model is designed to improve computational efficiency of the forward kinematics. To validate the final system, a new dataset is collected from a synchronized Kinect sensor and a commercial marker based MOCAP system. The system is shown to be able to accurately and robustly track various motions using a skeleton-only model at a high speed.

Finally, conclusions and future work will be presented in Chapter 7.

Chapter 2

Related Work

In this chapter, related work for camera based human pose recovery will be reviewed. Starting with discriminative approaches, both search based methods and regression based methods are summarized in Section 2.1. References for including temporal constraints and regressing from voxels will also be provided. In Section 2.2, generative approaches are surveyed in more details, including systems with a multi-camera setup and a single view setup. In the single view case, different sensing modalities are covered, including monocular camera, standard stereo camera and other depth sensors. Approaches range from Bayesian filtering to Iterative Closest Point (ICP) based methods. This is followed by a review of algorithms for extracting higher level image features by detecting and recognizing body parts in Section 2.3. Two methods for integrating body part recognition algorithms into model based pose estimation will be described briefly in Section 2.4. Recent research advances conducted at Microsoft Research based on the Kinect technology will be reviewed in Section 2.5, and a summary is provided at the end of the chapter (Section 2.6).

2.1 Discriminative Approaches

Discriminative approaches attempt to find a mapping between the image feature space and the human pose space. Without using an explicit human model, the estimation is usually achieved by either searching for nearest neighbours within a large set of exemplars [14, 15], or regressing a mapping function from a large set of training data [16, 17]. Most of the algorithms are designed to work on individual monocular images, but temporal information can also be included [45, 46]. Voxel based regression for a multi-camera setup has also been investigated [36].

In [14], Mori *et al.* propose a search based method for human pose recovery. Their work focuses on image matching and pose refinement. The shape context descriptor is adopted and object deformation is considered for localizing each joint location. Since shape context based matching needs to be performed with every exemplar, this method is very computationally expensive. When the exemplar set is large, the computational complexity will rapidly become prohibitively high. Therefore, Shakhnarovich *et al.* propose a Parameter-Sensitive Hashing algorithm based on the efficient Locality-Sensitive Hashing [15], which increases the search efficiency by a significant margin. Exemplars can also be videos as described in [47]. The correlations between the input video and the video templates are computed, and an exemplar sequence is selected. Joint positions are then estimated with approximate inference using Gibbs sampling and gradient ascent.

Compared to search based methods, regression based methods can be very efficient during testing. In [16], Agarwal *et al.* propose to learn a direct mapping from shape descriptor vectors extracted from human silhouettes to human poses by non-linear regression. Histograms of shape contexts are used to encode the shape features, and both a regularized least square based regressor and a Relevance Vector Machine (RVM) regressor are evaluated. However, direct regression from silhouettes can cause ambiguities due to missing information. This will lead one silhouette to correspond to several possible poses. To address this problem, Agarwal *et al.* propose to first cluster the poses of the training data, and then learn a mixture of regressors which is able to provide multiple hypotheses with probabilities [18]. The trained model is combined with a CONDENSATION algorithm based tracking system to automatically detect tracker failures and re-initialize the system if necessary. Similarly, Ning *et al.* use Bayesian Mixture Expert (BME) regression to deal with the possibly multi-valued mapping, as described in [48].

Another challenge for regression based methods is that the mapping between the image feature space and the pose space is usually highly non-linear. To deal with this non-linearity, Rosales *et al.* propose to use a supervised learning model named the Specialized Mappings Architecture [17]. Several specialized forward mappings and an inverse mapping are regressed, with each forward mapping for a certain domain of the input space. The difficulty, however, is how to optimally learn the specialized domains and mapping functions.

Moreover, supervised learning based methods will suffer when the training dataset becomes too large. This can happen when the system is designed to cover a large variety of human shapes and motions, especially for activity independent pose estimation applications. In [49], Urtasun *et al.* propose to use a local mixture of Gaussian Processes with small local neighbourhoods to achieve sparse probabilistic regression. The on-line regression scheme can also cope with complex, high-dimensional, and multi-modal mappings efficiently. The proposed algorithm is able to determine when the training examples are redundant and ready for pruning.

Temporal information can also be included into discriminative approaches. Bissacco *et al.* exploit both appearance and motion information by defining suitable features of individual images and their temporal neighbours [45]. However, the rough position of the person needs to be provided. A regression map from image features to human pose is learned using boosting techniques. In [46], Sminchisescu *et al.* formulate 3D human motion in monocular video sequences as a tracking problem based on observations encoding the image silhouette. Temporal constraints are enforced in the discriminative framework by constructing density propagation rules. The multi-modal state distributions are learned using conditional BME models.

Different features can be extracted from an image. Silhouettes are commonly used because of their simplicity. In [50], Poppe *et al.* compare three shape descriptors computed from a silhouette, namely the Fourier descriptor, the Shape Context descriptor and the Hu moment descriptor. Features other than silhouette have also been proposed, such as the Appearance and Position Context feature proposed in [48]. Voxel based regression is also explored, such as in [36]. 3D shape context descriptors are extracted from the reconstructed voxels, and Mixture of Probabilistic Principle Component Analysis (PCA) is used to transform them into a compact feature vector. Multi-Variate RVM is adopted for regressing a single mapping from the feature to a low dimensional representation of full body pose.

2.2 Generative Approaches

Generative approaches adopt explicit human models, and human pose estimation is usually formulated as a tracking problem. By making use of the prior knowledge of human kinematics, body shape, and temporal motion continuities, systems of this type are able to avoid the need for a large corpus of training data and to achieve accurate pose recovery for arbitrary motions. Considering the difficulty in obtaining training data and the goal of recovering general human motions, the work in this thesis follows the generative approach. In the following subsections, algorithms and systems within this scope are reviewed, categorized by the camera setup.

2.2.1 Multi-Views

Many systems proposed in the literature consider a multi-camera setup. Human motion is observed from multiple directions, and therefore the effect of occlusion is mitigated. One popular approach is to fuse images from multiple cameras within a particle filter framework [20, 24, 51]. Alternatively, this setup provides the possibility of reconstructing 3D voxels of the human body, which can provide a strong cue for the subsequent search or optimization.

One of the main challenges in full body human motion tracking is the high dimensional search space. In [6], Deutscher *et al.* propose a modified version of the particle filter to cope with this problem, termed the Annealed Particle Filter. The Simulated Annealing concept is introduced into the particle filer, which is achieved by filtering in multiple layers with increasing sharpness in the fitness function and decreasing randomness for the particles. In this way, the particles become more concentrated after going through weighting and resampling, and the number of particles required is consequently reduced significantly. Later, two improvements are made based on this work [52], *i.e.* an algorithm for automatic search space partitioning and a crossover operator for parallel searching in different partitions [20]

To evaluate different human motion tracking algorithms quantitatively, Balan *et al.* provide the first public dataset containing synchronized multi-camera, calibrated videos and 3D human motion ground truth [53]. A basic particle filter and an Annealed Particle Filter were implemented as baseline tracking algorithms. Different likelihood functions and prior motion models are compared and discussed based on the proposed error metrics. Based on this dataset, a new image likelihood function is developed by Balan *et al.* based on the visual appearance of a tracked subject to loosen the assumption of good foreground segmentation [54]. The dataset is then further extended to the HumanEva dataset, which becomes a benchmark for evaluating human motion tracking algorithms with a multi-camera setup, as summarized in [55].

The benefit of using a more sophisticated human model is demonstrated in [24]. Bandouch *et al.* present a marker-less 3D human motion tracking system using multiple cameras based on an industrial anthropometric human model. The model consists of a 51 DOF skeleton model and a precise 3D surface mesh model with torsion deformations. Other ergonomic considerations are also included, such as forces and motion limits. A Partitioned Particle Filter is used for tracking, with the silhouette used as the only cue for the particle weight evaluation. They demonstrate that using silhouette only can be sufficient in some multi-camera scenarios. In [51], they extend the previous work by comparing the Annealed Particle Filter and a particle filter with partitioned sampling, and conclude that both methods have complementary strengths. A combined algorithm is proposed to track human motion with higher robustness, but consuming a lower computational effort. However, the complicated human model entails high computational cost and makes it difficult to achieve a real-time implementation.

Multiple cameras also provide the possibility of 3D voxel reconstruction. The benefit of this operation before further processing is shown in [56], [57] and [26]. In [56], 3D voxels are reconstructed using a standard Shape-from-Silhouette (SFS) approach. The resulting 3D visual-hull is used in both human model estimation and human motion tracking. In [57], a hierarchical and statistical method based on colour was proposed during the reconstruction. Blobs which are attached to a kinematic model are matched with the 3D voxels using an Expectation-Maximization (EM) algorithm. The human pose is finally estimated by computing the inverse kinematics based on the matched blobs. This system is reported to run in real-time on a single PC. The system proposed in [26] is also based on volumetric reconstruction. After obtaining a 3D voxel representation, an objective function is established, which consists of both surface matching and edge matching. The tracking is achieved by optimizing this function using the Stochastic Meta Descent (SMD) algorithm. SMD is a gradient decent scheme with adaptive and parameterized step sizes, which is proven to perform well in solving local optimization problems.

Although the multi-camera setup provides advantages as presented above, the synchronization and coordination of the multiple cameras increases the system complexity. Moreover, the pre-calibrated cameras are mounted at permanent locations, limiting the portability of the system and incurring higher hardware expenses.

2.2.2 Single View

Instead of using multiple cameras, research effort has also been devoted to the more challenging problem of tracking human motion from a single view. This setup is usually preferable for robot applications and human machine interaction applications. While some researchers attempt to track from monocular image sequences [32, 22], more recent works tend to incorporate depth information from a stereo camera or a depth sensor. With the rapid advances in real-time depth sensing technology, using depth sensors has become popular in close range indoor scenarios.

Human motion tracking using a monocular camera only is challenging because of the missing depth information due to camera projection. Sidenbladh *et al.* propose a particle filter based human pose tracking system to cope with this situation [32]. A generative model is constructed, including a 3D articulated human model, a pinhole camera model and a dynamic model. The human subject is modelled as a kinematic tree structure, with cylinders and spheres describing the shapes of the body parts. The perspective projection model is adopted to capture the effect of projection from different depths. Two dynamic models, a general one and a learned motion specific one, are investigated. A basic particle filter is applied for the state evolution and estimation. In the prediction step, the appearance of each limb is formulated from the previous frame, and is used to predict the appearance of the current image based on the predicted pose. A likelihood function is designed based on the grey-level difference between the observed and predicted image, which essentially models the image motion in a non-parametric way. Note that a known background is not assumed in this system, benefiting from the explicit modelling of the limb appearances.

Chen *et al.* present a human motion tracking system with a monocular camera mounted on a moving robot in [39]. To achieve faster performance, the task is simplified to upper body tracking only. The head of the tracked subject is first detected and the scale is estimated to approximate the depth. A particle filter is applied for configuration estimation with the search space partitioned into three subspaces. Three cues are fused during tracking: a colour cue, an edge cue and an optical flow cue. Instead of assuming a static camera, the camera is considered to be mobile and the system tries to benefit from the change in view angle. Both the motion of the image and the ego-motion of the camera are accounted for during the extraction of the various cues and during the final estimation. However, only very simple, in-plane movements are tested in the paper.

Monocular tracking of complex human pose is obviously difficult. There are generally three approaches to tackle this difficulty: by learning a more constrained motion prior, by improving the particle filter, and by using depth information from depth sensors.

Urtasun *et al.* propose a learning approach for acquiring priors of human motion dynamics to set strong constraints on the human pose tracking problem [33]. They advocate using Scaled Gaussian Process Latent Variable Models for learning from a small amount of training data, by projecting the high dimensional pose onto a low dimensional embedding. The learned model is applied to monocular image based human pose tracking, and achieved promising results using a simple stick human model and some 2D joint trackers. They further investigate the use of an alternative Gaussian Process Dynamical Model (GPDM) for accomplishing a similar task in [23], where a Balanced Gaussian Process Dynamical Model is proposed for learning a smoother motion model from diverse training motions.

There is also effort in improving the particle filter itself to achieve more efficient tracking. In [58], Lin *et al.* propose a modified particle filter termed Progressive Particle Filter. The idea is to apply the mean shift algorithm to guide the particles in approaching the local likelihood maxima. In addition, the configuration search of the human model is conducted in a hierarchical manner to further reduce the computational cost. The algorithm is tested on monocular input and demonstrates better performance than a standard particle filer.

Instead of using monocular cameras only, research has increasingly focused on incorporating depth information obtained from stereo cameras and other means of depth sensing.

In [35], Azad *et al.* propose a stereo camera based upper body tracking system, which uses the stereo setup implicitly. A dense depth map is not computed, but instead, the 3D locations of certain points are calculated through triangulation. Moreover, at each time step, the information from the two cameras is fused within a particle filter framework. Multiple cues are evaluated for weighting the particles, including a newly developed distance cue, which detects and tracks the head and hands based on skin-colour. A likelihood function is designed based on the distance between the predicted locations and the detected locations of these body parts. This cue is shown to be very effective by significantly reducing the search space implicitly, both through a 2D simulation and qualitative experimental results. The detection of these body parts also enables an automatic initialization for the tracking via inverse kinematics. During the particle weight calculation, the predicted model is compared to both images from the stereo camera pair, and the final weight is obtained by integrating both image streams. The system is able to track relatively simple upper body movements in real-time.

In a follow-up work, Azad *et al.* improved the system by applying hierarchical search to further reduce the number of particles required [59]. The pose of the head and torso is first estimated, followed by the estimation of the arms by using separate filters. However, to cope with the accumulative error generated by the imperfect human model and torso pose estimation, various extensions have to be added to achieve a smooth and robust tracking, including prioritized cue fusion, adaptive noise in the dynamics, extra DOF at the shoulders, and incorporating inverse kinematics in the predictions.

Fontmarty *et al.* present a similar work in [38]. Based on the well-known Annealed Particle Filter, they provide a better theoretical framework by adding Importance Sampling to achieve similar goals as [59]. For each frame, the samples are not only drawn from the estimated distribution from the previous frame, but also from a default initial pose and the pose recovered from inverse kinematics. This amended Annealed Particle Filter algorithm is compared with the existing particle filters, and is shown to outperform the ICONDENSATION algorithm [60]. Although the new algorithm does not show significant increase in accuracy over the standard Annealed Particle Filter, it does provide the capability for re-initialization when tracking is lost. The other contribution of this work is the validation of the advantage of fusing multiple cues, and a simple experiment is conducted to compare the effectiveness of the various cues.

Targeting a human robot interaction application, Sigalas *et al.* propose a different approach for dealing with the high dimensionality issue of upper body tracking by combining particle filters with a Hidden Markov Model (HMM) [42]. This work also assumes a stereo setup. First, the tracking space is reduced to 9 DOF by assuming a straight-up torso whose pose can be determined by localizing the head. The tracking space is partitioned into three subspaces and each arm is tracked independently using a particle filter under each quantified orientation of the torso. The arm tracking results are then used to decide the torso orientation. Obviously, this increases the computational cost by tracking under multiple torso orientation hypotheses.

With the development of efficient stereo algorithms, it has become possible to produce good quality dense depth maps in real-time. Grest *et al.* propose an Iterative Closest Point (ICP) algorithm using the point cloud generated by a stereo camera pair [61]. A 3D human mesh model is first constructed and correspondences between the model and the point cloud of each frame are found though an improved ICP algorithm. Then the pose estimation task is formulated as a non-linear optimization problem. Due to the closed form derivation of the Jacobians of the objective functional, the optimization problem is solved efficiently and this contributes to a near real-time system.

Besides the traditional stereo camera, TOF and structured-light depth sensors have also been used for human tracking. Knoop *et al.* propose a framework for full body motion tracking by fusing a variety of sensor data from a TOF sensor or a standard stereo camera [62]. The proposed method is mainly based on an ICP algorithm to set up point correspondences, and is improved by registering points with certain body parts, reducing the number of points, and adding correspondences obtained from 2D feature tracking when a standard stereo camera is used. The model consists of cylinders, and elastic bands are introduced to represent the joint constraints. The system is reported to run in real-time.

In addition to the ICP approaches for point clouds, the particle filter is also an effective framework for pose tracking with depth data. In [63], Bray *et al.* propose a new approach that wraps a particle filter around multiple Stochastic Meta Descent (SMD) based trackers for hand tracking using range sensor data. The new algorithm combines the complimentary strengths of a local optimizer and a particle filter to improve the efficiency in finding global optima, by enabling each particle to approach a local optimum while keeping multiple hypotheses simultaneously. The algorithm uses a deformable 3D hand mesh model on range sensing data, and is shown to outperform the basic SMD method and an Annealed Particle Filter algorithm with very few particles.

Generative approaches usually assume that the size and the shape of the tracked human body is known *a priori*. However, this assumption is difficult to satisfy for real-world systems. Another major problem for typical generative approaches is automatic initialization and recovery from loss of tracking. Sigal *et al.* propose an approach for solving these two issues simultaneously in [64]. A parametrized triangulated mesh model which is learned from a database of human range scans is used to represent the human body. A conditional mixture of kernel regressors is applied to recover the model parameters from monocular images, including the shape, size, and configuration. This provides an initial estimation for the more precise tracking and shape refinement following a generative optimization.

2.3 Body Part Recognition

Significant effort has been devoted to the detection and identification of body parts from colour and depth images. Some researchers consider body part identification as a standalone problem, while others integrate it as a subsystem of body pose recovery. The most commonly used approach is based on the assumption of skin colour, such as in [35] and [65]. The body parts are first detected using skin colour segmentation, and then additional heuristics are used to obtain a robust result. Obviously, these methods make a strong assumption about the appearance of the body part to be detected, which is not necessarily satisfied in a real application.

An interesting approach that explores more general features of body parts is proposed by Haritaoglu *et al.* [66], who develop a body part labelling system using silhouettes only. The system is based on two observations:

1. The head, hands, elbows, feet and knees are more likely to be found on the silhouette boundary.
2. The human body in any given posture has a topology structure which constrains the relative locations of body parts.

They first classify the human motion into pre-defined posture categories, and then apply convex hull analysis to label the body parts according to the prior knowledge of that posture. However, this method is constrained to known posture classes. Also, when the body parts to be labelled are not on the silhouette boundary, the algorithm will fail. Wu *et al.* propose an edgelet detector [67], which makes use of silhouette orientation information. An edgelet is a short segment of a line or curve, and a set of edgelets can represent the shape of a body part. The system is trained using the edgelet features for head and shoulder contours, torso contour and leg contours, and these body parts are searched for within an image. A joint likelihood is then formed by combining the responses from the body part detectors, and used to detect the full body. However, in this work, the postures are constrained to walking and standing, limiting the generality. Considering additional postures, *e.g.* squatting, will result in edgelet features that can be very different. Furthermore, between straight standing and deep squatting, there are a myriad of continuous postures with varying edgelet features. If the system is trained using a greater range of postures, the detector performance is likely to degrade.

Recently, depth sensors that can produce high quality depth images have gained increasing attention. Based on the depth images obtained from a TOF sensor, Plagemann *et al.* developed an endpoint body part identification and localization system [68]. They define a new type of interest point in 3D, a geodesic extremum on the surface mesh of a human subject. This definition is based on the fact that the geodesic distances from the endpoint body parts to the centre of the body are usually the longest. After the interest points are found, a boosted patch classifier is applied to the patches around each interest point for identification. This system explores an essential feature of the endpoint body parts, but forming the surface mesh and searching for interest points are computationally demanding, requiring 60ms to process each frame.

2.4 Pose tracking with Body Part Recognition

To deal with the automatic initialization and recovery problems for generative approaches, incorporating bottom-up body part recognition results can be very helpful.

Based on the work of [68], Ganapathi *et al.* propose an approach for combining top-down model based local search with bottom-up body part detection to achieve efficient human pose estimation from a single depth sensor [25]. An accurate human model is constructed including a kinematic chain skeleton model and a deformable surface model, while the dynamics and measurements are modelled probabilistically. A smooth likelihood function is designed and a sampling-based hill climbing algorithm is applied for local search. To deal with the failure cases during local optimization, bottom-up body part detection results are incorporated through an Evidence Propagation procedure. The detected body parts are pruned and associated with the model. The hill-climbing algorithm is applied iteratively near the MAP estimated state obtained by calculating a probabilistic inverse kinematics using the detected body parts, to search for the best estimation based on the likelihood measurement. The algorithm is accelerated through GPU implementation and achieves near real-time performance (4-10Hz).

To incorporate body part recognition results into the pose inference process more naturally, Sigal *et al.* proposed a loose-limbed human model [69]. Different from the typical tree structured human kinematic models, a loosely connected human model is essentially an undirected graphical model, in which nodes correspond to body parts, while edges impose constraints from kinematic, penetration, and temporal constraints considerations. Human pose estimation is then achieved by inference in the graphical model using a Particle Message Passing algorithm [70]. Automatic initialization and recovery is achieved and demonstrated by testing the algorithm on the HumanEva dataset [55].

2.5 Recent Advances (Kinect)

The task of interactive human pose tracking has recently been greatly simplified by the introduction of real-time depth cameras. Depth cameras offer several advantages over traditional intensity sensors, *e.g.* operation at low light levels, calibrated scale estimate, colour and texture invariance, and the ability to resolve silhouette ambiguities in pose. They also greatly simplify the task of background subtraction and make it straightforward to synthesize realistic depth images of people for building a large training dataset cheaply [19].

The Kinect sensor makes real-time depth sensing available at a low consumer price [71]. It is a Natural User Interacting (NUI) device designed by Microsoft initially as an accessory for the Xbox 360, subsequently extended to support Windows PCs. A Kinect sensor consists of a Red-Green-Blue (RGB) colour camera, a structured-light depth sensing system, and a microphone array. The structured-light depth sensing system contains an infra-red pattern projector and an infra-red camera. The projector projects a fixed dot pattern, which is perceived by the infra-red camera. The perceived pattern will be distorted when objects are present in the line of sight of the camera, resulting in differences from a previously recorded reference depth. The perceived pattern is compared with the reference pattern, and the dots are matched. The depth is then recovered according to the disparities using a stereo algorithm. The resolution for both the RGB camera and the depth camera is 640 by 480.

Using this hardware advance in depth sensing, Shotton *et al.* maps the difficult pose estimation problem into a simpler per-pixel classification problem by taking an object recognition approach [19]. The goal of the algorithm is to achieve high computational efficiency and high robustness. An intermediate body part representation is designed, and pixels are classified to different body part categories by using a very simple depth difference feature, which is naturally invariant to translation, and very computationally cheap. They generate a dataset containing realistic synthetic depth images of humans of many shapes and sizes in highly varied poses sampled from a large motion capture database. A deep randomized decision forest classifier is trained using hundreds of thousands of training images to avoid over-fitting. Confidence-scored 3D proposals of several body joints are generated by re-projecting the classification result into the 3D space and finding the local modes by a mean shift algorithm. Because of the independent per-pixel feature evaluation, the algorithm can be run in parallel. An optimized implementation runs in under 5ms per frame on the Xbox 360 GPU, at least one order of magnitude faster than existing approaches. The proposed approach is shown to be more accurate than the state-of-the-art algorithms on both real and synthetic data.

As described above, this approach requires a huge training dataset and extensive computational resource for training the classifier. An arbitrary definition of body parts that roughly align with the body joints is also needed. Moreover, the segmentation of body parts lies on the surface, whereas the joints are inside the body. In addition, the location of a joint cannot be estimated when the associated part is occluded.

2.6 Summary

This chapter reviewed the related work on camera based human pose recovery. Starting with discriminative approaches, search based methods and regression based methods were summarized. Approaches for including temporal constraints and regressing from voxels were also overviewed. Generative approaches were surveyed in more details, including systems with a multi-camera or a single view setup. In the single view case, different sensing modalities were covered, including monocular camera, standard stereo camera and other depth sensors. We also reviewed the algorithms for detecting and recognizing body parts, integrating body part recognition algorithms into model based tracking, and the recent research advances based on the Kinect technology.

Chapter 3

Background

In this chapter, we will review the fundamentals of the particle filter and partitioned sampling, which is the main tool used in this thesis for human pose tracking. A state-space approach is taken, and the multi-variable probabilistic system model is first described in Section 3.1. Then the tracking problem is formed as a Bayesian estimation problem in Section 3.2, and its solutions when the distributions are Gaussian are briefly discussed. In Section 3.3, the mathematical derivation and the pseudo-code of the particle filters are described. This includes the Monte Carlo method and importance sampling (Section 3.3.1), the Sequential Importance Sampling algorithm (Section 3.3.2), the Sampling Importance Resampling algorithm (Section 3.3.3), and finally the particle filter with partitioned sampling (Section 3.3.4).

3.1 System Model[1]

We will follow a state-space approach to tackle the tracking problem. For a discrete-time dynamic system, the goal is to estimate the state for each time step by making use of the noisy measurements observed. For a given system, the state contains all the information that is needed to calculate responses to present and future inputs, without reference to the past history of inputs and outputs. A ***State Vector*** x is used to include all the state variables, while a ***Measurement Vector*** z is formed to include all the noisy measurements from which the state is to be inferred. These vector representations make the state-space approach ideal for dealing with multi-variable systems.

[1] [72, 73]

The ***System Model*** usually consists of two parts: a ***Process Model*** which describes the evolution of the state with time, and a ***Measurement Model*** which relates the noisy measurements to the state. If the system is modelled in a probabilistic form, then the Bayesian approach can be applied to make inferences. A general process model can be described using equation:

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{3.1}$$

where k indicates the kth time step, $\{x_k, k \in \mathbb{N}\}$ is a state sequence, $\{v_k, k \in \mathbb{N}\}$ is an independent and identically distributed (i.i.d.) process noise sequence, and $f_k : \mathbb{R}^{N_x} \times \mathbb{R}^{N_v} \rightarrow \mathbb{R}^{N_x}$ is a possibly non-linear function, with N_x and N_v representing the dimensions of the state and process noise vectors. In addition, a general measurement model can be described using the equation:

$$z_k = h_k(x_k, n_k) \tag{3.2}$$

where $\{z_k, k \in \mathbb{N}\}$ is a measurement sequence, $\{n_k, k \in \mathbb{N}\}$ is an i.i.d. measurement noise sequence, and $h_k : \mathbb{R}^{N_x} \times \mathbb{R}^{N_n} \rightarrow \mathbb{R}^{N_z}$ is a possibly non-linear function, with N_x, N_n and N_z representing the dimensions of the state, measurement noise and measurement vectors respectively.

3.2 Bayesian Tracking[1]

To solve the tracking problem for the system modelled in Section 3.1, we seek the filtered estimates of x_k given the set of all available measurements $z_{0:k} = \{z_i, i = 0, \cdots, k\}$ up to time k. The Bayesian approach provides a rigorous framework for accomplishing this task, within which an optimal or sub-optimal solution can be developed.

The Bayesian approach aims at constructing the posterior Probability Density Function (PDF) $p(x_k|z_{1:k})$, based on the periodically received measurements. To achieve on-line processing, recursive filtering is adopted to process the data sequentially rather than as a batch. It is assumed that the initial PDF $p(x_0|z_0) \equiv p(x_0)$, *i.e.* the prior density of the state vector, is available (z_0 is empty). The posterior PDF can be obtained recursively in two steps for each recursion: a ***Prediction Step*** and an ***Update Step***. In the prediction step, the initial PDF is used as the posterior PDF $p(x_{k-1}|z_{1:k-1})$ at time $k-1$, and the state estimate is propagated from time $k-1$ to time k. Considering the unknown disturbances that could affect the system, prediction generally translates, deforms, and spreads the state PDF. This involves using the process model described in Equation 3.1 to obtain the prior

[1] [73]

PDF of the state at time k via the Chapman-Kolmogorov equation:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{3.3}$$

Note that in Equation 3.3, the fact $p(x_k|x_{k-1}, z_{1:k-1}) = p(x_k|x_{k-1})$ has been used since the system model describes a first order Markov process. The probabilistic model of the state evolution $p(x_k|x_{k-1})$ is defined by the system equation (Equation 3.1), and the known statistics of v_{k-1}.

In the update step, the latest measurement z_k becomes available, and this is incorporated to modify the prior based on the Bayes' rule:

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{3.4}$$

where the normalizing constant

$$p(z_k|z_{1:k-1}) = \int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k \tag{3.5}$$

The likelihood function $p(z_k|x_k)$ is defined by the measurement model described by Equation 3.2, and the known statistics of n_k. In the update step, the measurement z_k is used to update the predicted prior $p(x_k|z_{1:k-1})$ into the posterior $p(x_k|z_{1:k})$.

Equation 3.3 and Equation 3.4 provide a conceptual solution to the tracking problem, and it cannot by generally solved analytically other than under highly restrictive assumptions. When assuming f_k and h_k are linear functions, the process model and the measurement model can be written as

$$x_k = F_k x_{k-1} + v_{k-1} \tag{3.6}$$

$$z_k = H_k x_k + n_k \tag{3.7}$$

where F_k and H_k are matrices defining the linear functions.

Assuming v_{k-1} and n_k are drawn from Gaussian distributions, it can be proved that $p(x_k|z_{1:k})$ is Gaussian if $p(x_{k-1}|z_{1:k-1})$ is Gaussian, parametrized only by a mean and a covariance [74]. For a continuous state space, the optimal solution can be derived by using Equation 3.3 and Equation 3.4, and this yields the Kalman filter [75]. It should be noted that the Kalman filter can also be applied in a least square sense when the state posterior is not strictly Gaussian.

In the case that the system model is non-linear, a local linearisation of Equation 3.1 and Equation 3.2 can be applied to approximate the original model using linear equations. The Extended Kalman Filter (EKF) [75] follows this idea, and normally takes the first term of the Taylor expansion of the non-linear equations for approximation. There are also other methods for dealing with the non-linearity, such as the Unscented Kalman Filter (UKF) [76], where the non-linearity is dealt with by propagating a set of sampled points from a Gaussian distribution. However, this class of methods still assume the PDFs to be Gaussian. If the actual distribution is not Gaussian, particularly if it is with multi-modal, a Gaussian function is not a good approximation. Therefore, for more general cases without linear and Gaussian assumptions, the Monte Carlo method can be applied to simulate the approximation, which yields the class of particle filters.

3.3 Particle Filter[1]

3.3.1 Monte Carlo and Importance Sampling[2]

"Monte Carlo" (MC) is the general name for a class of methods for stochastic simulation. It is a way to approximate expectations by random sampling. Consider a possibly multidimensional continuous random variable x having a PDF of $p(x)$ which is greater than zero on a set of values $\mathbb{X}$. The expectation of a function $f(x)$ can be calculated using the equation:

$$E(f(x)) = \int_{x \in \mathbb{X}} f(x)p(x)dx \tag{3.8}$$

If a set of N samples of x is taken according to $p(x)$, denoted as $\{x^i\}_{i=1}^N$, the Monte Carlo estimate of $E(f(x))$ can be written as:

$$\tilde{E}(f(x)) = \frac{1}{N}\sum_{i=1}^{N} f(x^i) \tag{3.9}$$

If $E(f(x))$ exists, then the law of large numbers ensures that for arbitrarily small ϵ,

$$\lim_{N \to \infty} P(|\tilde{E}(f(x)) - E(f(x))| \geq \epsilon) = 0 \tag{3.10}$$

[1] [73]
[2] [77]

This means that as N gets larger, there is small probability that $\tilde{E}(f(x))$ deviates much from $E(f(x))$. Moreover, the Monte Carlo estimate is an unbiased estimation for $E(f(x))$ since:

$$\begin{aligned} E(\tilde{E}(f(x))) &= E\left(\frac{1}{N}\sum_{i=1}^{N} f(x^i)\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} E(f(x^i)) \\ &= E(f(x)) \end{aligned} \tag{3.11}$$

When the distribution $p(\cdot)$ is not easy to be sampled directly, importance sampling can be adopted to simulate the random variable according to another distribution. Let $q(x)$ be a density for the random variable x which takes positive values in $\mathbb{X}$ so that $\int_{x\in\mathbb{X}} q(x)dx = 1$. Since $\dfrac{q(x)}{q(x)} = 1$, we have

$$\begin{aligned} E_p(f(x)) &= \int_{x\in\mathbb{X}} f(x)p(x)dx \\ &= \int_{x\in\mathbb{X}} f(x)p(x)\frac{q(x)}{q(x)}dx \\ &= \int_{x\in\mathbb{X}} f(x)\frac{p(x)}{q(x)}q(x)dx \\ &= E_q\left(f(x)\frac{p(x)}{q(x)}\right) \end{aligned} \tag{3.12}$$

The Monte Carlo estimate for this expectation can be written as:

$$\sum_{i=1}^{N} f(x^i)w^i \tag{3.13}$$

where

$$w^i \propto \frac{p(x^i)}{q(x^i)} \tag{3.14}$$

is the normalized weight for the ith sample. The sample set $\{x^i, w^i\}_{i=0}^{N}$ is also called a particle set.

3.3.2 Sequential Importance Sampling

Sequential Importance Sampling (SIS) is a technique for implementing a recursive Bayesian filter by MC simulations, also known as bootstrap filtering [78], the condensation algorithm [79], survival of the fittest [80], *etc.* This algorithm forms the basis for most of the sequential MC filters.

For a discrete time dynamic system, the posterior PDF for $x_{0:k}$ can be denoted as $p(x_{0:k}|z_{1:k})$. The posterior expectation of the state can be calculated using equation:

$$E(x_{0:k}) = \int x_{0:k} \cdot p(x_{0:k}|z_{1:k}) dx_{0:k} \tag{3.15}$$

The key idea of the SIS algorithm is to apply MC simulation to approximate this integral using a summation by random sampling. According to the law of large numbers, as the number of samples increases to be very large, this summation approaches the actual expectation value.

However, it is generally impossible to sample from $p(x_{0:k}|z_{1:k})$ directly, so importance sampling is adopted. Suppose samples $\{x_{0:k}^i\}_{i=1}^{N_s}$ are drawn independently from a normalized proposal function $q(x_{0:k}|z_{1:k})$ (importance density), which has a support including that of the state posterior. Then the posterior expectation can be written as

$$E(x_{0:k}) = \int x_{0:k} \cdot \frac{p(x_{0:k}|z_{1:k})}{q(x_{0:k}|z_{1:k})} \cdot q(x_{0:k}|z_{1:k}) dx_{0:k} \tag{3.16}$$

and it can be approximated using

$$\tilde{E}(x_{0:k}) = \sum_{i=1}^{N_s} x_{0:k}^i w_k^i \tag{3.17}$$

where

$$w_k^i \propto \frac{p(x_{0:k}^i|z_{1:k})}{q(x_{0:k}^i|z_{1:k})} \tag{3.18}$$

is the normalized weight of the ith particle at time k. The set of particles $\{x_{0:k}^i, w_k^i\}_{i=0}^{N_s}$ is a random measure that characterizes the posterior PDF $p(x_{0:k}|z_{1:k})$.

However, this method is not sequential. For the sequential filtering case, at time k, samples constituting an approximation to the posterior expectation of time $k-1$ $\{x_{0:k-1}^i, w_{k-1}^i\}_{i=0}^{N_s}$ are available. The task for the current iteration is to draw a new sample set and to calculate the sample weights based on these. To derive the weight update equation, $p(x_{0:k}|z_{1:k})$ is first expressed as:

$$\begin{aligned}
p(x_{0:k}|z_{1:k}) &= \frac{p(z_k|x_{0:k}|z_{1:k-1})p(x_{0:k}|z_{1:k-1})}{p(z_k|z_{1:k-1})} \\
&= \frac{p(z_k|x_{0:k}|z_{1:k-1})p(x_k|x_{0:k-1}|z_{1:k-1})p(x_{0:k-1}|z_{1:k-1})}{p(z_k|z_{1:k-1})} \\
&= \frac{p(z_k|x_k)p(x_k|x_{k-1})p(x_{0:k-1}|z_{1:k-1})}{p(z_k|z_{1:k-1})} \\
&\propto p(z_k|x_k)p(x_k|x_{k-1})p(x_{0:k-1}|z_{1:k-1})
\end{aligned} \tag{3.19}$$

If the importance function is chosen to factorize such that:

$$q(x_{0:k}|z_{1:k}) = q(x_{0:k}|x_{k-1}, z_{1:k})q(x_{0:k-1}|z_{1:k-1}) \tag{3.20}$$

By substituting Equation 3.19 and Equation 3.20 into Equation 3.18, the weight update equation can then be written as:

$$\begin{aligned}
w_k^i &\propto \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)p(x_{0:k-1}^i|z_{1:k-1})}{q(x_k^i|x_{0:k-1}^i, z_{1:k})q(x_{0:k-1}^i|z_{1:k-1})} \\
&= w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{0:k-1}^i, z_{1:k})}
\end{aligned} \tag{3.21}$$

In addition, if $q(x_k|x_{0:k-1}, z_{1:k}) = q(x_k|x_{k-1}, z_k)$, this means the importance density is only dependent on x_{k-1} and z_k. This is particularly useful when only a filtered estimate of the posterior expectation is required at each time step. In this scenario, only x_k need to be stored and one can discard the trajectory $x_{0:k-1}$ and the history of observations $z_{1:k-1}$. The modified weight is then

$$w_k^i \propto w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_k)} \tag{3.22}$$

The SIS algorithm thus consists of propagating the support points according to $q(x_k|x_{k-1}^i, z_k)$ and computing the weights according to Equation 3.22 as each measurement is received. A pseudo-code description of the algorithm is given in Algorithm 1.

Algorithm 1 SIS Algorithm Pseudo-code [73]

```
$[\{x_k^i, w_k^i\}_{i=1}^{N_s}] = \text{SIS}[\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$
for $i = 1 \to N_s$ do
    Draw $x_k^i \sim q(x_k|x_{k-1}^i, z_k)$
    Calculate weight $w_k^i$ using Equation 3.22
end for
```

Degeneracy Problem

The SIS particle filter usually suffers from the degeneracy problem, where after a few iterations, all but very few particles have negligible weight. In [81], Doucet shows that the variance of the importance weights can only increase over time, which means it is impossible for the original SIS filter to avoid degeneracy. When degeneracy occurs, a large computational effort is devoted to updating particles whose contribution to the estimation is almost zero. A suitable measure of the degeneracy of the algorithm is the effective sample size N_{eff} introduced in [82] and [83] and defined as:

$$N_{eff} = \frac{N_s}{1 + Var(w_k^{*i})} \tag{3.23}$$

where

$$w_k^{*i} = \frac{p(x_k^i|z_{1:k})}{q(x_k^i|x_{k-1}^i, z_k)} \tag{3.24}$$

is the "true weight", which cannot be evaluated exactly. However it can be approximated using:

$$\tilde{N}_{eff} = \frac{1}{\sum_{i=1}^{N_s}(w_k^i)^2} \tag{3.25}$$

where w_k^i is the normalized weight obtained using Equation 3.22. Note that $N_{eff} \leq N_s$, and smaller N_{eff} means more severe degeneracy. It is obvious that the degeneracy problem is undesirable for particle filters. A very large N_s can be used to reduce this effect, but this is often impractical. The two major approaches for dealing with degeneracy are the choice of good importance density and the use of resampling.

Choice of Importance Density

We can choose a good importance density $q(x_k|x^i_{k-1}, z_k)$ to minimize $Var(w^{*i}_k)$ so that N_{eff} is maximized. It has been shown in [81] that the optimal density function is:

$$\begin{aligned} q_{opt}(x_k|x^i_{k-1}, z_k) &= p(x_k|x^i_{k-1}, z_k) \\ &= \frac{p(z_k|x_k|x^i_{k-1})p(x_k|x^i_{k-1})}{p(z_k|x^i_{k-1})} \end{aligned} \tag{3.26}$$

Substituting Equation 3.26 into Equation 3.22 yields:

$$\begin{aligned} w^i_k &\propto w^i_{k-1}p(z_k|x^i_{k-1}) \\ &= w^i_{k-1}\int p(z_k|x'_k)p(x'_k|x^i_{k-1})dx'_k \end{aligned} \tag{3.27}$$

This optimal choice suffers from two major drawbacks. It requires the ability to sample from $p(x_k|x^i_{k-1}, z_k)$ and to evaluate the integral over the new state. In general, neither of these two are straightforward to do. Only when x_k is a member of a finite set, or when $p(x_k|x^i_{k-1}, z_k)$ is Gaussian, is the optimal importance density possible.

For many other models, it is often convenient to choose the importance density to be the prior:

$$q(x_k|x^i_{k-1}, z_k) = p(x_k|x^i_{k-1}) \tag{3.28}$$

Substituting Equation 3.28 into Equation 3.22 then yields:

$$w^i_k \propto w^i_{k-1}p(z_k|x^i_k) \tag{3.29}$$

This is the most common choice of importance density since it is intuitive and simple to implement. It is also the choice for the Sampling Importance Resampling algorithm described in Section 3.3.3. However, there are also many other densities that can be chosen, and the choice is a crucial design step for the filter.

Resampling

The basic idea of resampling is to remove the particles with negligible weights and replicate the particles with larger weights to concentrate the particles around the higher posterior areas. The effect of degeneracy can be reduced by applying this operation whenever a significant degeneracy is observed, *e.g.* when N_{eff} falls below some threshold N_T. In the resampling step, a new set of particles $\{x_k^{i*}\}_{i=1}^{N_s'}$ is generated by resampling with replacement according to the discrete representation of a posterior $p(x_k|z_{1:k})$ given by a particle set $\{x_k^i, w_k^i\}_{i=1}^{N_s}$:

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{3.30}$$

so that the probability of $x_k^{i*} = x_k^i$ equals w_k^i. The resulting sample in fact approximates an i.i.d. sample from the posterior $p(x_k|z_{1:k})$, so the weights for all the new particles are the same (equal to $1/N_s'$). There are four basic resampling algorithms in the literature, multinomial resampling, stratified resampling, systematic resampling and residual resampling [84]. Systematic resampling is usually preferred since it is simple to implement, takes $O(N_s)$ time, and minimizes the MC variation. A pseudo-code is shown in Algorithm 2, where $U[a, b]$ is the uniform distribution on the interval $[a, b]$ (inclusive). A general particle filter with resampling is then described in Algorithm 3.

Algorithm 2 SR Algorithm Pseudo-code [73]

```
$[\{x_k^{i*}, w_k^{i*}\}_{i=1}^{N_s'}]$ = RESAMPLE $[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$
Initialize $c_1 \leftarrow 0$
for $i = 2 \rightarrow N_s$ do
    $c_i \leftarrow c_{i-1} + w_k^i$
end for
Initialize $u_1 \sim U[0, \frac{1}{N_s'}]$
for $j = 1 \rightarrow N_s'$ do
    $u_j = u_1 + \frac{1}{N_s'}(j-1)$
    while $u_j > c_i$ do
        $i \leftarrow i + 1$
    end while
    $x_k^{j*} \leftarrow x_k^i$
    $w_k^{j*} \leftarrow \frac{1}{N_s'}$
end for
```

However, resampling also brings problems. After the resampling, the particles with high weights are statistically replicated many times and this will lead to a loss of diversity, which is known as ***Sample Impoverishment***. This effect can be very severe for the case of very small process noise. There are techniques dealing with this problem, *e.g.* the resampling-move algorithm [85] and regularization [86].

Algorithm 3 General Particle Filter Pseudo-code [73]

```
$[\{x_k^i, w_k^i\}_{i=1}^{N_s}] = \text{PF } [\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$
for $i = 1 \rightarrow N_s$ do
    Draw $x_k^i \sim q(x_k|x_{k-1}^i, z_k)$
    Calculate the weight $w_k^i$ using Equation 3.22
end for
Normalize the weights $\{w_k^i\}_{i=1}^{N_s}$
Calculate $\tilde{N}_{eff}$ using Equation 3.25
if $\tilde{N}_{eff} < N_T$ then
    Resample $[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$ using Algorithm 2
end if
```

3.3.3 Sampling Importance Resampling

The Sampling Importance Resampling (SIR) filter first proposed in [78] is a commonly used recursive Bayesian filter. It can be derived from the SIS algorithm easily by choosing the prior density as the importance density, *i.e.* $q(x_k|x_{k-1}^i, z_{1:k}) = p(x_k|x_{k-1}^i)$, and including a resampling step at the end of each recursion. This choice of the importance density implies that samples need to be drawn from the prior density $x_k^i \sim p(x_k|x_{k-1}^i)$, which can be achieved by first generating process noise $v_{k-1}^i \sim p(v_{k-1})$ and then setting $x_k^i = f_k(x_{k-1}^i, v_{k-1}^i)$. As discussed in Section 3.3.2, the weight update equation now becomes

$$w_k^i \propto w_{k-1}^i p(z_k|x_k^i) \tag{3.31}$$

However, since resampling is applied at every time step, we have $w_{k-1}^i = 1/N_s$ for $\forall i$. Therefore the weight now becomes

$$w_k^i \propto p(z_k|x_k^i) \tag{3.32}$$

Note that the weight given by the proportionality in Equation 3.32 needs to be normalized before the resampling step. The pseudo-code for one recursion of the SIR filter is described in Algorithm 4 and illustrated in Figure 3.1.

In the diagram, the $\sim$ symbol denotes resampling, $\otimes$ denotes convolving with dynamics, and $\times$ denotes multiplication by the observation density. The dynamical convolution operation $\otimes$ randomly draws samples from the conditional distribution $p(x_k|x_{k-1}^i)$. Its effect on the distribution represented by the particle set is to transform a distribution $p(x_{k-1}|z_{1:k-1})$ into $p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1}$. The multiplica-

Algorithm 4 SIR Filter Pseudo-code [73]

```
$[\{x_k^i, w_k^i\}_{i=1}^{N_s}] = \text{SIR } [\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$
for $i = 1 \to N_s$ do
    Draw $x_k^i \sim p(x_k|x_{k-1}^i)$
    Calculate the weight using $w_k^i = p(z_k|x_k^i)$
end for
Normalize the weights $\{w_k^i\}_{i=1}^{N_s}$
Resample $[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$ using Algorithm 2
```

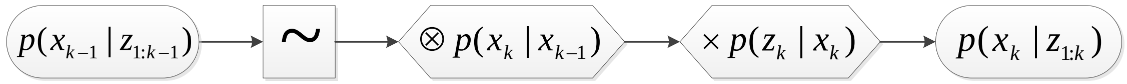

$p(x_{k-1} \mid z_{1:k-1})$
$\sim$
$\otimes\, p(x_k \mid x_{k-1})$
$\times\, p(z_k \mid x_k)$
$p(x_k \mid z_{1:k})$

Figure 3.1: Diagram for One Iteration of SIR Filter

tion operation $\times$ modifies the particle weights to $w_k^i \propto w_{k-1}^i \times p(z_k|x_k)$. Its probabilistic effect is to transform a distribution $p(x_k|z_{1:k-1})$ into the distribution proportional to $p(z_k|x_k) \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1}$. This is precisely the Bayes update rule for dynamical diffusion governed by $p(x_k|x_{k-1})$ and likelihood function $p(z_k|x_k)$.

Figure 3.2 shows an example of two recursions of an SIR algorithm. Starting with 10 particles shown as light blue circles at the top, all with the same weights. The likelihoods are evaluated for each particle using the current measurement to approximate the posterior distribution, denoted using the black curve. This results in the red circles, whose sizes indicate the weights. Resampling is applied and the particles are then diffused according to the system dynamics. This becomes the start point of the next iteration.

The assumptions of applying the SIR filter are not overly constrictive. The state dynamics function $f_k(\cdot,\cdot)$ and the measurement function $h_k(\cdot,\cdot)$ in Equation 3.1 and Equation 3.2 need to be known. Also, it should be possible to sample realizations from the process noise distribution $p(v_{k-1})$ and from the prior $p(x_k|x_{k-1}^i)$. Finally, the likelihood function $p(z_k|x_k)$ needs to be available for point-wise evaluation, at least up to proportionality.

The SIR algorithm has the advantage that the importance weights are easily evaluated and that the importance density can be easily sampled. However, since the importance density is independent of the measurement z_k, the state space is explored without any knowledge of the observations. As a result, the SIR filter can be inefficient and sensi-

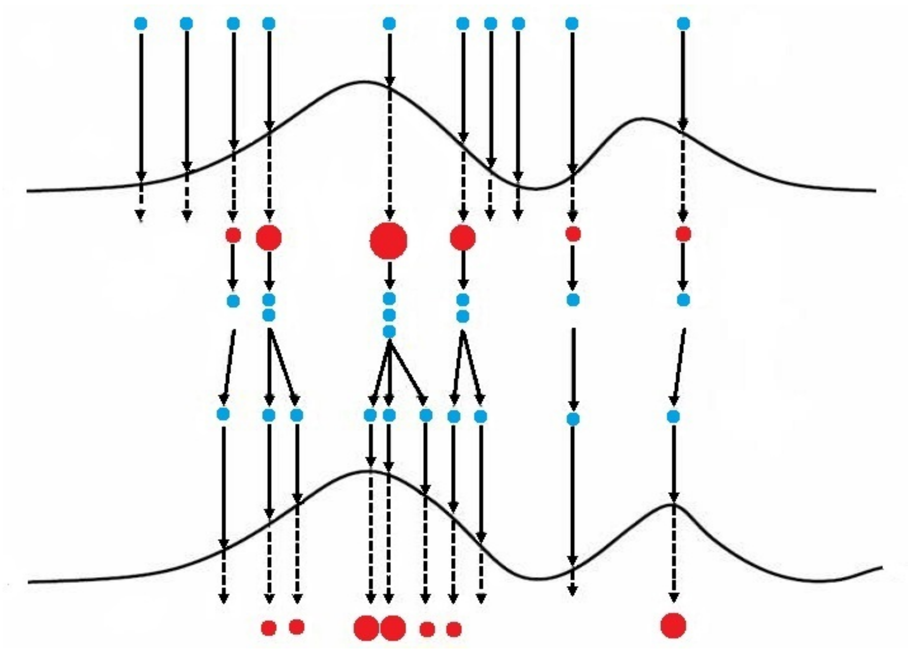


Figure 3.2: Illustration for Two Iterations of SIR Filter

tive to outliers. Furthermore, since resampling is applied for every iteration, the sample impoverishment problem can appear very rapidly.

3.3.4 Partitioned Sampling[1]

As the dimension of the tracking space increases, the SIR algorithm rapidly becomes infeasible because the number of particles required for successful tracking increases exponentially with the increase of the state dimensionality. Partitioned sampling is a way of applying particle filers to tracking problems with high dimensional configuration spaces, to avoid the large computational cost normally required. It was first introduced in [79] for tracking multiple objects, and then successfully applied to articulated object tracking in [34] based on the fact that it is a statistical analogue of a hierarchical search.

Partitioned sampling is based on the concept of ***Weighted Resampling***. Let $p(x)$ be a strictly positive, continuous function on $\mathbb{X}$, which represents the configuration space. Weighted resampling with respect to $p(\cdot)$ is an operation on a particle set which "populates"

[1] [34]

the peaks of $p(\cdot)$ with particles, without altering the distribution actually represented by the particle set. Given a particle set $\{x^i, w^i\}_{i=1}^{N_s}$, weighted resampling produces a new set $\{x^{i'}, w^{i'}\}_{i=1}^{N_s}$ as follows. First, importance weights are defined as:

$$\rho^i = \frac{p(x^i)}{\sum_{j=1}^{n} p(x^j)} \tag{3.33}$$

Next, indices $k_1, k_2, \ldots, k_n$ are selected by setting $k_i = j$ with probability ρ^j, independently for $i = 1, \ldots, N_s$. Finally, new states and weights are defined as $x^{i'} = x^{k_i}$ and $w^{i'} = w^{k_i}/\rho^{k_i}$. This choice of weights has the effect of counteracting the extent to which the particles are "biased" by the importance weights. The weighted resampling operation does not alter the underlying distribution represented by the particle set, as proven in [87]. The operation of weighted resampling with respect to $p(\cdot)$ is denoted as $\sim p$.

The strategy of partitioned sampling is to divide the state space into two or more "partitions", and sequentially apply the dynamics for each partition followed by an appropriate weighted resampling operation. Partitioned sampling can be adopted if the following conditions hold:

- The configuration space $\mathbb{X}$ can be partitioned as a Cartesian product $\mathbb{X} = \mathbb{X}_1 \times \cdots \times \mathbb{X}_{N_p}$.
- The dynamics $p(x_k|x_{k-1})$ can be decomposed as convolution products as $p(x_k|x_{k-1}) = p(x_{k,1}|x_{k-1}) \otimes p(x_{k,2}|x_{k,1}) \otimes \cdots \otimes p(x_{k,N_p}|x_{k,N_p-1})$.
- Weighting functions defined on $\mathbb{X}_i$ for $i = 1, \ldots, N_p - 1$ are available, which are peaked in the same region as the posterior restricted to $\mathbb{X}_i$.

One example of such a system is an articulated object. Assuming the state space can be divided into N_p partitions satisfying the three conditions above, a general partitioned sampling algorithm for one time step can be illustrated using the diagram shown in Figure 3.3. For each partition, the dynamics for that partition $p(x_{k,j}|x_{k,j-1})$ is first applied, and followed by a weighted resampling operation according to function p_j, which usually is a likelihood function for that partition. After the dynamics of the last partition is applied, a final weight for each particle is evaluated, resulting in an approximation of the posterior.

Partitioned sampling not only has the advantage of lowering the overall number of particles, but also makes it possible to vary the number of particles devoted to each partition. Partitions which require a large number of particles for acceptable performance can be satisfied without incurring additional effort in the other partitions. However, the question

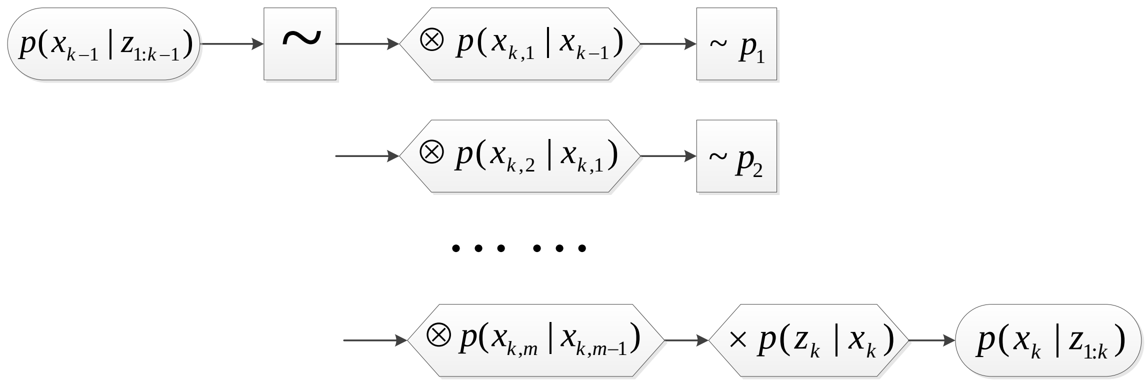


Figure 3.3: One Step of Partitioned Particle Filter

of how to distribute the particles among the partitions optimally is still an open problem. In practice, this is usually done based on intuition and experiments.

In the particular case where the overall likelihood $p(z|x)$ can be expressed as a product of the weighting functions and another easily calculated function $p_{N_p}(z_{N_p}|x_{1,\ldots,N_p})$, as

$$p(z|x) = p_1(z_1|x_1)p_2(z_2|x_{1,2}) \cdots p_{N_p}(z_{N_p}|x_{1,\ldots,N_p}) \tag{3.34}$$

the process can be simplified in terms of using standard resampling rather than weighted resampling (Figure 3.4). This is exactly equivalent to the algorithm described in Figure 3.3, but with only half the number of likelihood evaluations. This is very important for increasing the algorithm efficiency because the computational expense largely resides in evaluating the likelihood functions.

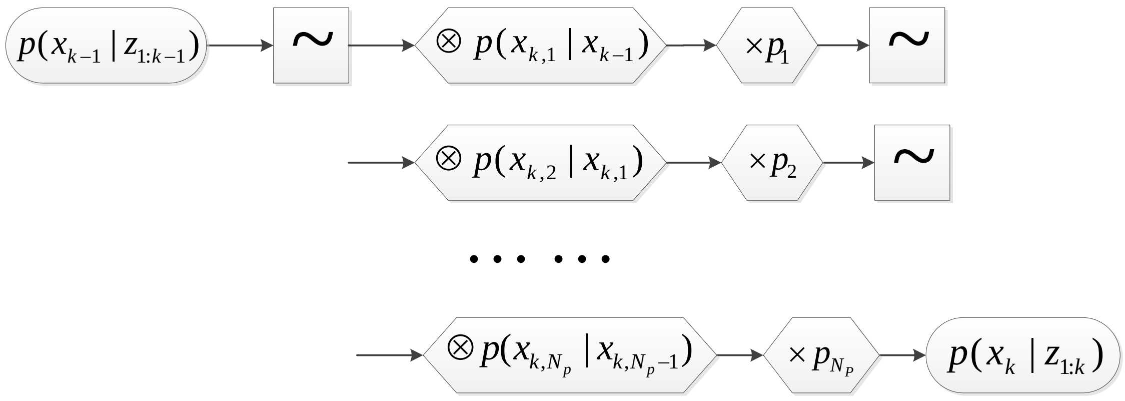

$p(x_{k-1} | z_{1:k-1})$
$\sim$
$\otimes\, p(x_{k,1} | x_{k-1})$
$\times p_1$
$\sim$
$\otimes\, p(x_{k,2} | x_{k,1})$
$\times p_2$
$\sim$
$\otimes\, p(x_{k,N_p} | x_{k,N_p-1})$
$\times p_{N_P}$
$p(x_k | z_{1:k})$

Figure 3.4: Simplified Partitioned Particle Filter

Chapter 4

Model Based Tracking

In this chapter, a model based human motion tracking framework using particle filters is proposed. The work in this chapter is validated on the Carnegie Mellon University Graphics Lab Motion Capture (CMU MOCAP) dataset [88], which includes both videos and ground truth data captured using a marker based system. First, the construction of a skeleton model compatible with the dataset is described in Section 4.1. This is followed by the description of an upper body tracking algorithm capable of tracking constrained motions in real-time (Section 4.2). Then the algorithm is improved in terms of the human model, the projection model, and the tracking algorithm, and shown to achieve near real-time tracking for more complex full body motions (Section 4.3). The two systems are tested using videos from CMU MOCAP dataset, and both tracking videos and quantitative analysis are provided.

4.1 Skeleton Model

The human body is considered as an articulated object, consisting of a skeleton model and an outer shape model. The skeleton model described in this section will be used in the tracking systems in this chapter.

To be compatible with the CMU MOCAP dataset, the skeleton model is defined according to the Acclaim Skeleton Files (ASF) convention. The skeleton model is constructed by parsing ASF files, and thus it is configurable. The model contains at most 30 bones and 64 DOF, but the actual structure and active DOF can be specified in the ASF file according to the application.

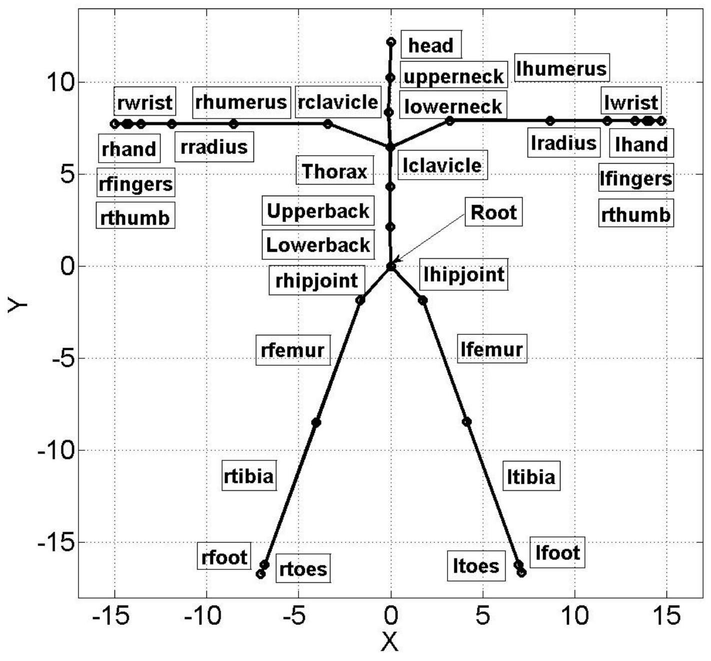
The skeleton is modelled as a hierarchical tree structure, consisting of several kinematic chains. It starts at a root node, which connects to the two hips and the lower back. Each node in the tree indicates a bone except for the root node, which describes the orientation and translation of the entire body from the world coordinate frame. Each node can have at most three children. The tree structure is illustrated in Figure 4.1, and the DOF for each bone is listed in Table 4.1.

Table 4.1: Bone DOF

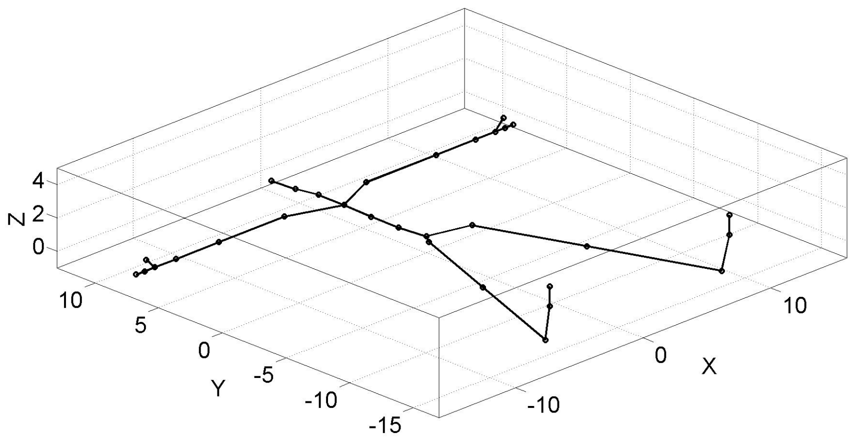
Bone	DOF	Bone	DOF	Bone	DOF
lhipjoint	0	lfemur	3	ltibia	1
lfoot	2	ltoes	1	rhipjoint	0
rfemur	3	rtibia	1	rfoot	2
rtoes	1	lowerback	3	upperback	3
thorax	3	lowerneck	3	upperneck	3
head	3	lclavicle	2	lhumerus	3
lradius	1	lwrist	1	lhand	2
lfingers	1	lthumb	2	rclavicle	2
rhumerus	3	rradius	1	rwrist	1
rhand	2	rfingers	1	rthumb	2

In the skeleton model, each bone is represented by a vector defined in the world frame. Each bone has only one connecting point for its children bones, which is its far tip. Each bone vector starts from the connecting point of its parent, and ends at its own connecting point. In addition to the world frame, for each bone there is a local coordinate frame originating at its starting point, which initially has the same orientation as the world frame. The local frames are not necessarily aligned with the corresponding bones in the default pose. So for each bone, an ***Offset Angle Rotation*** is defined, which rotates the local frame to align with the bone to reduce the number of DOF. For instance, the tibia only has one DOF after the offset angle rotation, while in its initial local frame it needs two to represent its movement.

All the rotations used in the model are Roll-Pitch-Yaw rotations by default. Roll-Pitch-Yaw is a set of Euler angles used to define the relative orientation by performing a sequence of rotations about fixed frames. Given the rotation angles for roll, pitch and yaw ϕ_x, ϕ_y, and ϕ_z respectively, the rotation matrix for each element rotation can be calculated according to Equation 4.1, Equation 4.2 and Equation 4.3.



4.1.1: 2D View with Bone Names



4.1.2: 3D View

Figure 4.1: 3D Skeleton Model

$$R_x(\phi_x) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\phi_x & -sin\phi_x & 0 \\ 0 & sin\phi_x & cos\phi_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

$$R_y(\phi_y) = \begin{pmatrix} cos\phi_y & 0 & sin\phi_y & 0 \\ 0 & 1 & 0 & 0 \\ -sin\phi_y & 0 & cos\phi_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

$$R_z(\phi_z) = \begin{pmatrix} cos\phi_z & -sin\phi_z & 0 & 0 \\ sin\phi_z & cos\phi_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

The 3D position of each bone is calculated using forward kinematics. For a given bone, the position is defined as the position of its far tip. The following example explains the bone position calculation. Consider the left humerus, which has an ***Offset Angle*** of α_x, α_y, and α_z in Roll-Pitch-Yaw order. The rotation matrix for the offset angle rotation can be calculated according to Equation 4.4.

$$L = R_z(\alpha_z) \cdot R_y(\alpha_y) \cdot R_x(\alpha_x) \tag{4.4}$$

where R_x, R_y, and R_z are obtained from Equation 4.1, Equation 4.2, and Equation 4.3.

Also, the bone has a rotation in its local frame after the offset angle rotation, which is defined as ***Joint Angle Rotation***. This joint angle rotation is given by β_x, β_y, and β_z in Roll-Pitch-Yaw order after the offset angle rotation. The rotation matrix for the joint angle rotation can be calculated using Equation 4.5.

$$R = R_z(\beta_z) \cdot R_y(\beta_y) \cdot R_x(\beta_x) \tag{4.5}$$

where Rx, Ry, Rz can also be obtained from Equation 4.1, Equation 4.2, and Equation 4.3.

Moreover, there is a translation from the origin of the local frame of the parent bone (left clavicle) to the origin of the local frame of the current bone. This translation is actually the bone vector of the left clavicle in the frame of the left clavicle. The translation matrix can be calculated by Equation 4.6.

$$T_h^c = \begin{pmatrix} 1 & 0 & 0 & x_c \\ 0 & 1 & 0 & y_c \\ 0 & 0 & 1 & z_c \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.6}$$

where $(x_c \; y_c \; z_c)$ is the bone vector for the left clavicle in its own local frame.

Given the bone vector of the left humerus p^h, we can calculate its 3D position in the local frame of the left clavicle by:

$$p^c = M_h^c \cdot p^h \tag{4.7}$$

where

$$M_h^c = T_h^c \cdot L_h \cdot R_h \cdot (L_h)^{-1} \tag{4.8}$$

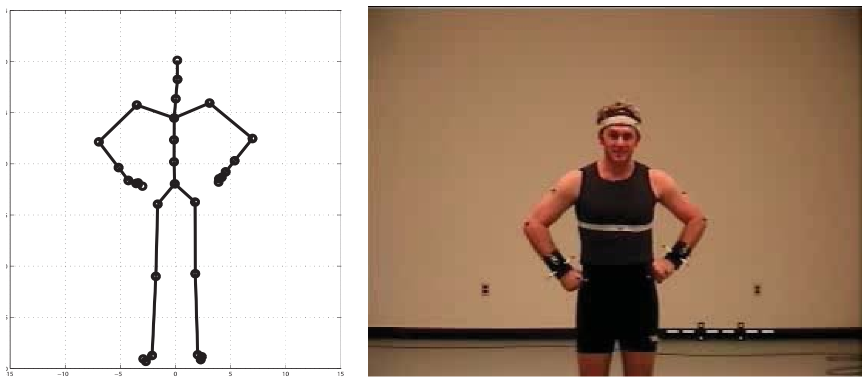
First, p^h is rotated by L_h^{-1} to compute the position in the frame after the offset angle rotation. In this frame, we perform the joint angle rotation R_h. However, we need to obtain the position in the local frame of the left humerus, so it is rotated by a L_h. After multiplying the translation matrix T_h, we get p^c.

According to the principle of the kinematic chain, the transformation matrix M is accumulative. As a result, given a bone vector p^i, its position in the world frame can be calculated by Equation 4.9.

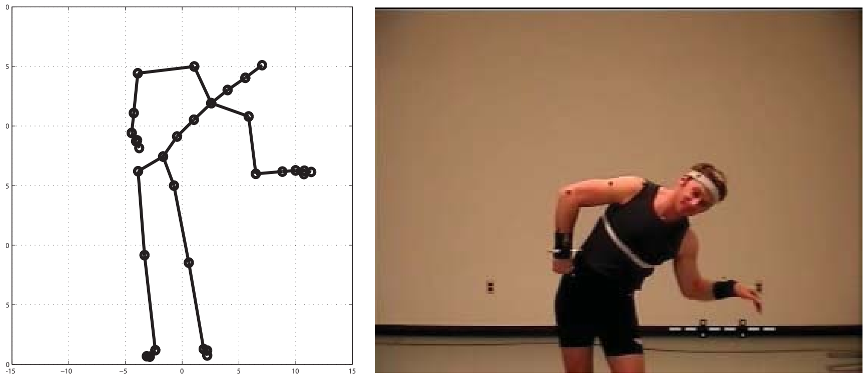
$$p^{world} = M_{root}^{world} \cdots M_i^{i-1} \cdot p^i \tag{4.9}$$

T and L are fixed matrices determined by the model, only the R matrix varies during motion based on the joint angles. By changing the joint angles, the model can represent different postures.

To validate the skeleton model, the tree construction algorithm and forward kinematics calculation were tested using several ASF files and Acclaim Motion Capture (AMC) files. The algorithm parses ASF and AMC files and outputs a Matlab ".m" file to plot the figures of human posture in 3D. By comparing the pose of the 3D model with the corresponding



4.2.1: Sample Frame 1



4.2.2: Sample Frame 2

Figure 4.2: Skeleton Model Validation

image captured from the video, the algorithm is confirmed to be correct. Figure 4.2 shows sample frames for the model validation.

4.2 Real-time Upper Body Tracking[1]

In this section, a marker-less upper body human pose tracking system is proposed. The system is based on the original form of particle filter, the Sequential Importance Resampling (SIR) filter, and is able to run in real-time. Designed for good portability, this

[1]A version of this section has been published [89].

system uses monocular image sequences as input. This reduces computational effort, but increases the difficulty of the estimation, particularly due to the missing depth. To simplify the problem, human motions are constrained to contain insignificant depth changes only. Therefore, a weak perspective model can be assumed for camera projection.

A sub-model of the skeleton model described in Section 4.1 is used. The outer shape is modelled using 2D rectangles. To enable real-time performance, only the edge cue and the distance cue are used in the likelihood calculation. The system is tested using videos from the CMU MOCAP dataset.

4.2.1 Human Model

For the upper body tracking application, a sub-model of the skeleton model described in Section 4.1 is used, which consists of 16 bones and 11 DOF. The 11 DOF are 2 DOF for the base translation (in plane), 1 DOF for the base rotation (in plane), 3 DOF for each shoulder, and 1 DOF for each elbow.

In order to represent the outer shape of the human body and facilitate contour projection, we designed a shape model as a supplement to the skeleton model. To make the projection more efficient, only 2D rectangles are used to represent the relevant body parts. By using rectangles, only the four corner points need to be projected, reducing the computation significantly. For upper body tracking, only the outer shapes of the torso and the arms are modelled.

4.2.2 Projection Model

The human model describes the posture in 3D, while the observation, which is the image sequence taken by the camera, is in 2D. Therefore, a projection model is needed to project 3D points onto the 2D image plane. The coordinate frame within which the ground truth data is captured is used as the world frame. Since neither the camera matrix nor the world frame is explicitly given by the dataset, they must first be estimated.

The camera is modelled as a pinhole camera for simplicity, and the projection is modelled as weak perspective projection. This works under the assumption that the human subject does not perform motions that cause significant depth changes. For a weak perspective projection, the projection matrix is independent of the actual depth. The projection is described by a series of coordinate transformations. Since all these transformations are linear, the transformations can be combined and the final transformation can be represented by the projection formula in homogeneous form as shown in Equation 4.10.

$$\begin{pmatrix} x_{image} \\ y_{image} \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ b_{11} & b_{12} & b_{13} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{3D} \\ y_{3D} \\ 1 \end{pmatrix} \tag{4.10}$$

To obtain the projection matrix, 35 points in 2D image and in 3D space are matched manually, and the projection matrix is estimated through least square surface fitting. The fitting results are listed in Table 4.2 and the projection result is illustrated in Figure 4.3.

Table 4.2: Fitting Results

Coefficients (with 95% confidence bounds)	Goodness of fit
$a_{11} = 7.789$ $(7.617, 7.96)$ $a_{12} = 0.0955$ $(-0.1555, 0.3465)$ $a_{13} = 159.4$ $(153.5, 165.3)$	SSE: 327.7 R-square: 0.9954 Adjusted R-square: 0.9951 RMSE: 2.899
$b_{11} = -0.2756$ $(-0.5472, -0.003997)$ $b_{12} = -8.802$ $(-9.199, -8.405)$ $b_{13} = 336$ $(326.7, 345.3)$	SSE: 818.1 R-square: 0.981 Adjusted R-square: 0.9801 RMSE: 4.58

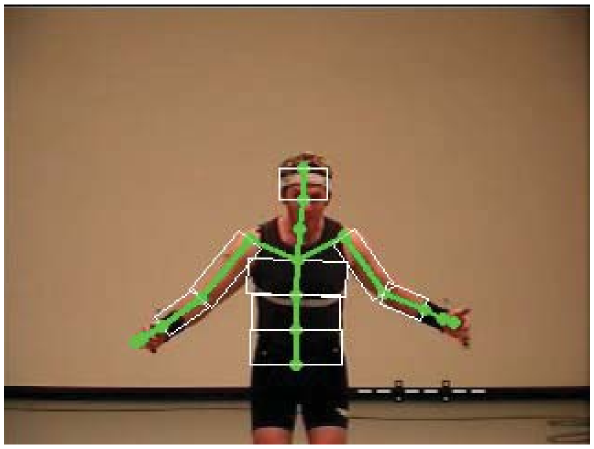


Figure 4.3: Projection Result

4.2.3 Particle Filter Implementation

To achieve real-time processing, the system is implemented in C++ using the OpenCV library [90]. The particle filter implementation consists of three key components:

Initialization and Prediction

The system is initialized using the ground truth data. After initialization, particles are generated by duplicating the initial state. A dynamic model is used to propagate the particles from the previous time step to the current time step. A zero order model is used as the dynamic model, which means for each particle, Gaussian noise is added to the previous state to generate a prediction. Joint limits and limb penetration detection are not considered.

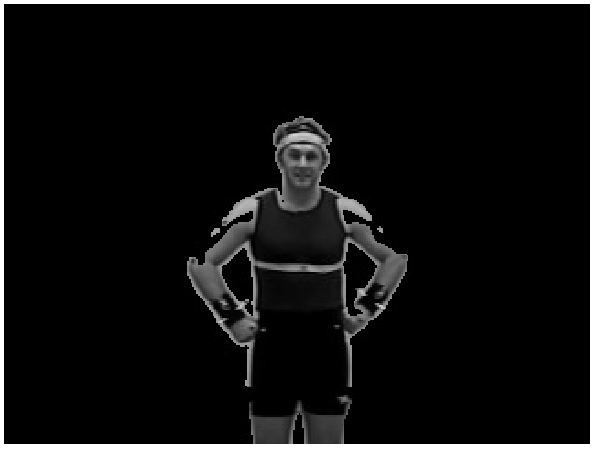
Weight Calculation

The core of a particle filter is the calculation of the particle weights, using cues from observed images. Each particle's weight reflects how well the particle matches the observation. The cues that can be considered in a human MOCAP application include colour, edge, distance, region, motion and etc. As discussed in [35], the edge cue and the distance cue are the most time efficient cues. Therefore, these two cues were selected and the final weight is computed by fusing them. For each cue, a distance is computed and the particle weight is calculated by combining the distance measures.

Edge Cue: Edges are important sources of information about the shapes of the objects in an image. Edge detectors calculate the gradients of intensity in an image, and the edge is obtained by thresholding the gradient image. Canny detector [91] is considered as the best edge detector [92], because it thresholds with hysteresis thresholds and preserves both strong edges and weak edges connected to them. The Canny detector is used in this work.

However, the edges contained in the background make the resulting edge map ambiguous. Background subtraction is applied to segment the foreground before extracting the edges. We tested the segmentation in both grey image and colour image, and a comparison shows that the background subtraction in the colour space is better (Figure 4.4). When doing segmentation in the colour space, the pixel difference is computed as the Euclidean distance in the Red-Green-Blue (RGB) space.

After foreground segmentation, the Canny Operator is applied to extract the edges within the foreground region. Dilation is applied to make the edges thicker (Figure 4.5).


4.4.1: In Grey Space

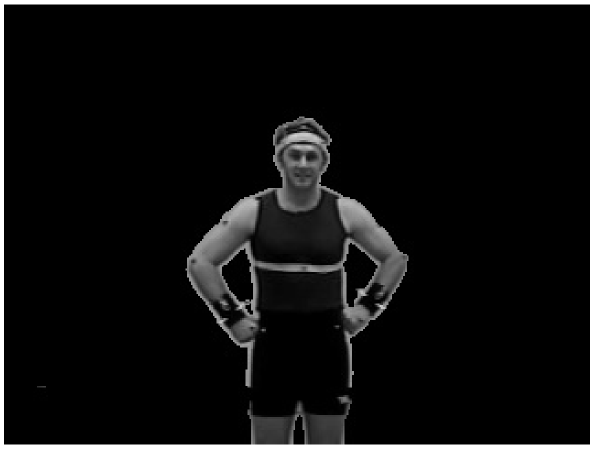
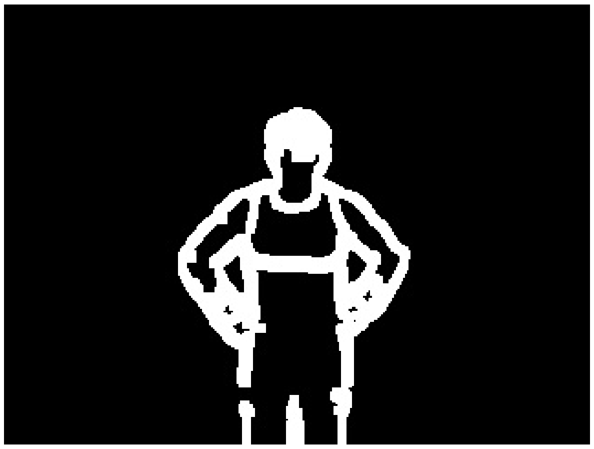

4.4.2: In RGB Space

Figure 4.4: Foreground Segmentation


4.5.1: Dilated Foreground Edges

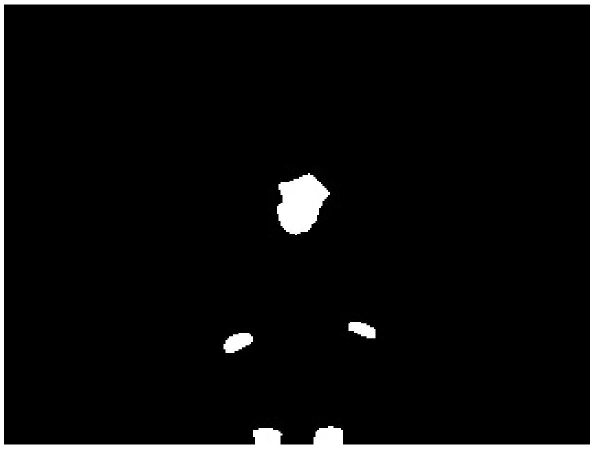

4.5.2: Skin-Color Blobs

Figure 4.5: Image Processing Results

The distance for the edge cue is calculated by comparing the projected edges from each particle with the detected edges in the observed image. For each particle, the projected edges are examined to determine whether the corresponding pixels are detected as edge pixels in the edge map. The distance is calculated according to Equation 4.11:

$$d^E = \sum_{i=0}^{N} \frac{\sum_{j=0}^{m_i}(1 - b_{ij})^2}{m_i} \tag{4.11}$$

where N is the number of all the body parts considered, m_i is the number of pixels of the projected edges in the ith body part, and b_{ij} is the binary value for the jth pixel along the projected edge in the ith body part. The distance for each body part is normalized by the length of the total edges in that body part, and summed together to form the distance for that particle.

Distance cue: In addition to the edge cue, we also make use of the three skin-colour blobs: one face and two hands. Skin colour segmentation is done in the YCrCb space, which separates the colour information from the brightness information. However, the arms and legs, together with the background also contain skin colour. We filter the blobs by area and ratio criteria, and make the assumption that the legs are always lower than the hands, and that the two hands do not cross. The extracted skin-colour blobs are shown in Figure 4.5.2.

The distance for the distance cue is calculated by summing the Euclidean distances between the predicted positions and the detected positions for all three blobs. The three skin-colour blobs are not always available, but it does not affect the result if any of them is absent for all the particles.

Integration: The distances calculated from the edge cue and the distance cue are not in the same scale. In order to integrate the cues without introducing any bias and to increase the weight resolution, the distances are rescaled into the range of $[0, 1]$ linearly. Then the weight for each cue is calculated from the distance according to the weighting function Equation4.12

$$w_i = A^{-d_i} \tag{4.12}$$

where w_i is the weight for the ith particle from either edge cue or distance cue, A is a number larger than 1 which affects the sharpness of the weighting function, and d_i is the distance for the ith particle from either the edge or distance cue.

The final weight for each particle is computed by weighted multiplication of the weights calculated from both cues (Equation 4.13), and then normalized so that the weight sum equals 1.

$$w_i = (w_i^E)^\alpha \cdot (w_i^D)^\beta \tag{4.13}$$

Re-sampling and Estimation

At each time step, re-sampling is applied to redistribute the particles in the search space while maintaining the PDF, to deal with the degeneration problem [73]. We use the weighting function as the importance function for re-sampling, *i.e.*, number of times that each particle is copied is proportional to its weight value. The systematic re-sampling approach is adopted [84]. After re-sampling, the estimation of the state is computed by calculating the expectation of the posterior PDF.

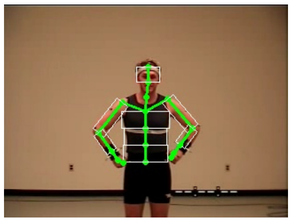
4.2.4 Experimental Results

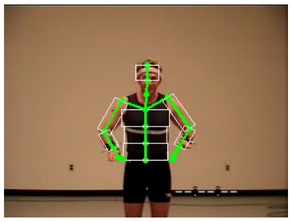
Tracking Video

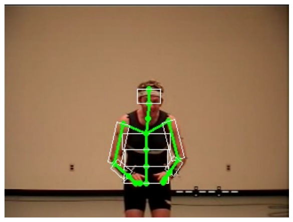
Figure 4.6 shows captured frames from a tracking video when 300 particles are used. The system runs at a speed of 20Hz, and the tracking is accurate and robust from visual inspection. In the testing video, an actor performs the "little tea pot" movement. This movement includes complex motions in every DOF of the upper body model, especially at the two shoulder joints. In the video, the green lines indicate the projected skeleton, and the white rectangles indicate the projected outer shape. From the tracking video, we can observe that for most of the time, the system captures the human motion correctly. In Figure 4.6.10, however, the tracking result of the head is slightly off from its true position, due to the insufficient DOF in the back and the neck in the skeleton model. Also, the system almost loses tracking for the left arm in Figure 4.6.13, but it completely recovers after about 30 frames. This demonstrates the robustness of the particle filter.

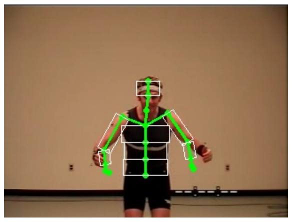
Quantitative Error Analysis

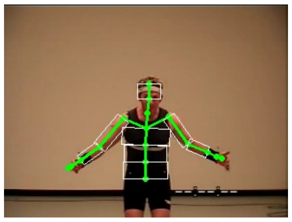
In order to perform a quantitative evaluation, error metrics must be defined. Because we are most interested in the final tracking result, our error is measured from the expected

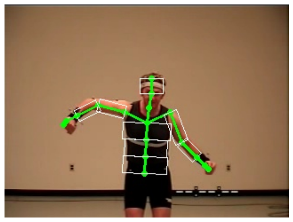

4.6.1: Frame32

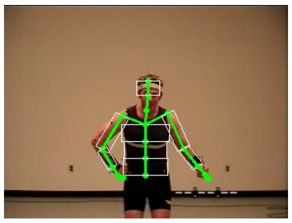

4.6.2: Frame43

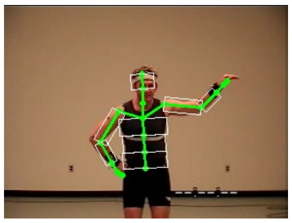

4.6.3: Frame93

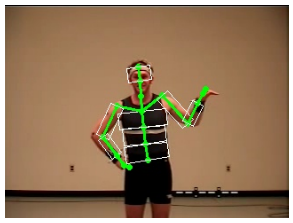

4.6.4: Frame104

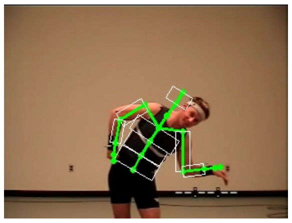

4.6.5: Frame119

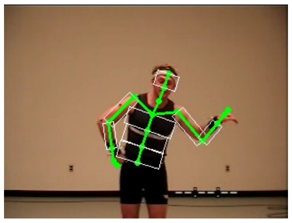

4.6.6: Frame159

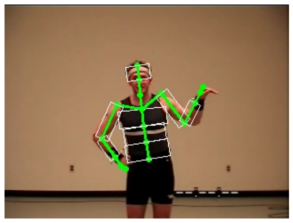

4.6.7: Frame194

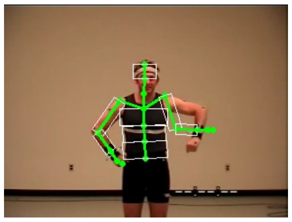

4.6.8: Frame224


4.6.9: Frame315


4.6.10: Frame416


4.6.11: Frame449


4.6.12: Frame471


4.6.13: Frame491

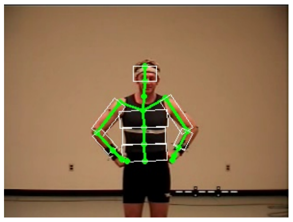

4.6.14: Frame532

Figure 4.6: Frames Extracted from Test Video

pose by calculating the distance from the expected pose to the ground truth. The error is measured in terms of the positions of 8 key joints. Three error terms are defined:

1. ***Average Error (AE)***: calculated by averaging the errors over all the joints in one frame.
2. ***Joint Average Error (JAE)***: calculated by averaging the error for each joint throughout the tracking.
3. ***Overall Average Error (OAE)***: the average error over all the joints throughout the tracking.

In addition to the averages, the error standard deviation can also be calculated to measure the fluctuation of the error. In all the following experiments, each test is run 10 times to compute the error statistics.

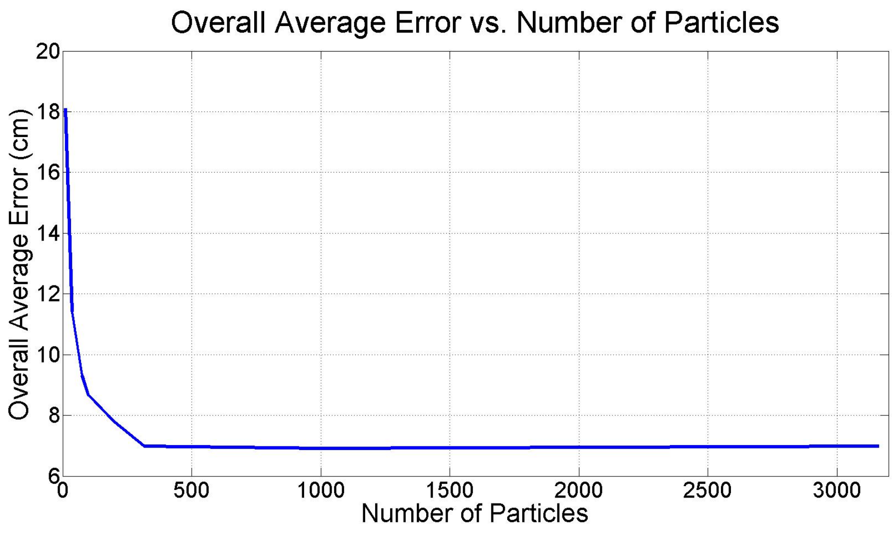


Figure 4.7: OAE for Upper Body

We first test the ***Overall Average Error*** when the number of particles increases from 10 to 3000 to find out the optimal number of particles for our system, as plotted in Figure 4.7. This figure demonstrates that when the number of particles increases, the error decreases monotonically from 18cm to 6.9cm. The error drops most significantly as the number of particles increases from 10 to 300, and then remains almost constant. This

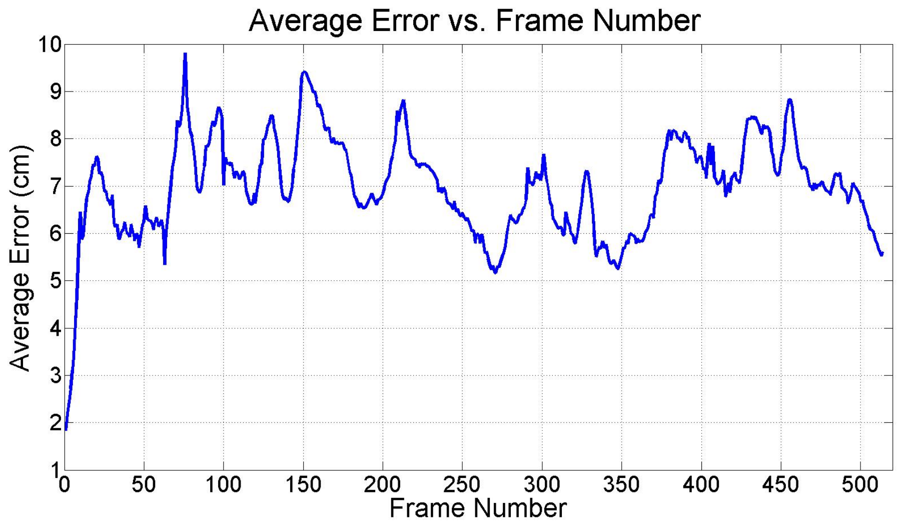


Figure 4.8: AE Over Frames for Upper Body

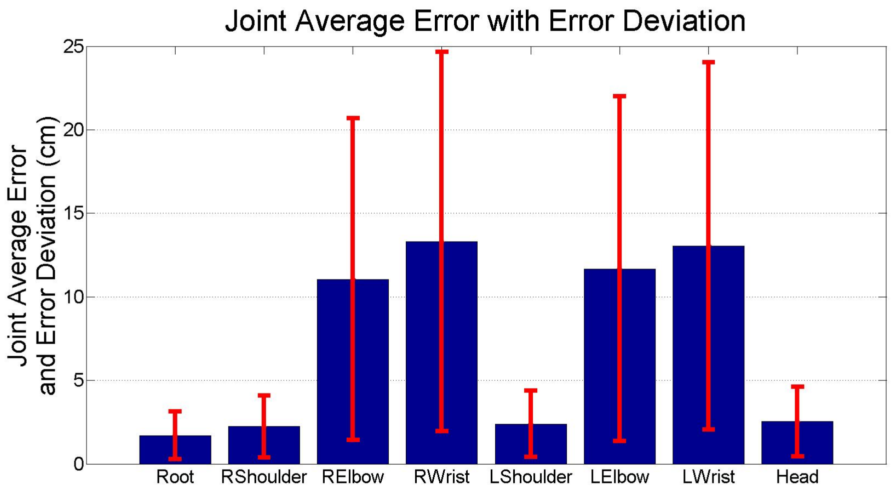


Figure 4.9: JAE with Deviation for Upper Body

indicates 300 is the optimal number. When 300 particles are used, the system runs at 20Hz with an overall average error of 7cm and an error standard deviation of 6.07cm.

In Figure 4.8, we show the ***Average Error*** throughout the tracking when 300 particles are used. The higher errors correspond to the faster motions. Figure 4.9 is the bar plot for the ***Joint Average Error*** with error standard deviation. We can see that both the error and the error standard deviation is very small for the root, head and both shoulders. The relatively larger error and error deviation for both arms indicate the arm tracking is poorer than the torso. This is because the arm joints move much faster than the torso within a larger angle range.

The error is mainly due to the missing depth information. In this test video, the torso stays at a certain depth, but the depth of the arms can still change. Without any depth information, the 3D locations of the arms become difficult to track accurately. The large increase of the error from the torso to the arms in Figure 4.9 supports this conclusion. Furthermore, the tracker loses track for some body parts occasionally, like the situation in Figure 4.6.13. In this case, the tracker loses track for the left arm, while left hand is close to its true position. Here, the edge cue becomes more informative in weighting the particles, and it is more powerful in helping the tracker to recover. If we can adjust the weights adaptively, the system would recover faster and the performance would be improved. Moreover, the limited DOF of the model also introduces error.

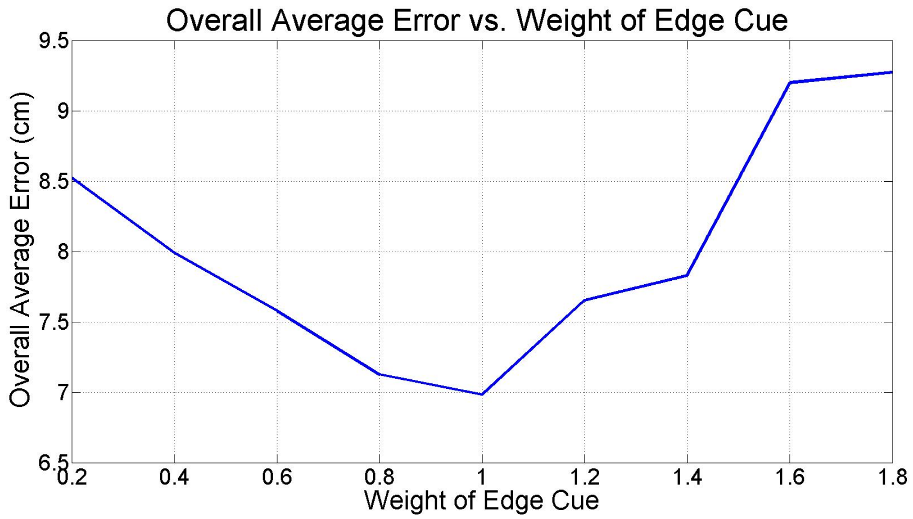


Figure 4.10: OAE with Changing Cue Weights for Upper Body

We also tested the effect of adjusting the weight for each cue. The weight for the edge cue (α) is increased from 0.2 to 1.8 with a step of 0.2, while the sum of the weights remains at 2. This is because after rescaling, the range for the distances calculated from each cue is $[0, 1]$, so the range of the total distance is $[0, 2]$. From Figure 4.10, we can see that when the weight for each cue is 1, the error is smallest. This implies both cues are equally important in our system and provide independent sources of information to the tracker. Note that the relative cue importance may change, *e.g.* if the background also contains many skin-colour objects that are difficult to distinguish, the distance cue will become less salient.

4.2.5 Summary

In this section, we proposed a real-time marker-less upper body MOCAP system within the particle filter framework. Designed for a humanoid application, this system uses monocular image sequences as the input. Successful tracking is achieved through using a particle filter, despite the lack of depth information. To enable real-time human position estimation, the time efficient edge cue and the distance cue are used during the likelihood calculation. The system is tested using videos from the CMU MOCAP database, which includes both the videos and the ground truth motion data captured using a Vicon system. By using this publicly available dataset, we are able to perform quantitative error analysis which is essential in system evaluation and algorithm comparison. The size of each frame in the test videos is 320 by 240, and our system is capable of tracking successfully at a speed of 20Hz on a 2.67GHz Intel Core2 Quad CPU, with an average error of 7cm for each key joint, and an error standard deviation of 6.07cm. Considering the missing depth information, this result demonstrates good accuracy.

4.3 Near Real-time Full Body Tracking[1]

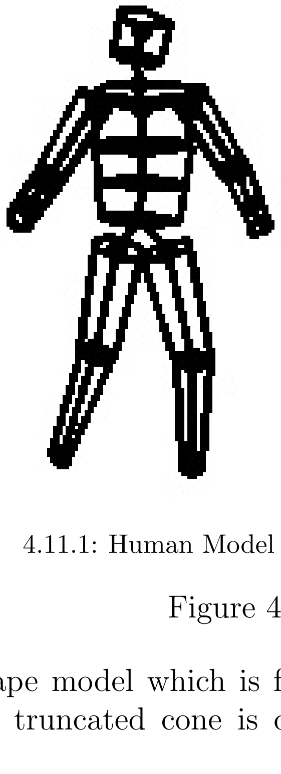
In this section, we propose a new articulated model-based full body human motion capture system intended for a humanoid application, which is capable of running in near real-time. The system is designed for stereo input, and the human motion is tracked by a particle filter with partitioned sampling in order to solve the high dimensionality problem for full body tracking. The tracker makes use of multiple cues, including a newly developed depth cue. To enable a quantitative error analysis of the algorithm performance, rather than

[1]A version of this section has been published [93].

using video obtained from a stereo camera, we use videos from the publicly available CMU MOCAP dataset [88], which also includes ground truth data obtained from a marker-based motion capture system. Since the dataset does not provide access to stereo camera images, we generate virtual depth images to simulate the true depth images. The system is tested with challenging videos, and the results demonstrate that our system is capable of tracking through random, fast and complex motions with high speed and good accuracy.

4.3.1 Human Model

The human model is composed of a skeleton model and an outer shape model (Figure 4.11.1). The skeleton model is as described in Section 4.1, with 25 DOF activated: 3 DOF for the root translation, 3 DOF for the root rotation, 3 DOF for the neck, 3 DOF for each shoulder, 1 DOF for each elbow, 3 DOF for each hip and 1 DOF for each knee. For each joint angle, an angle limit is specified to reduce the search space and to avoid impossible configurations. The angle limit is implemented as a reflective surface, when a predicted angle goes beyond the limit, it is simply reflected back into the allowed region with the same magnitude.


4.11.1: Human Model

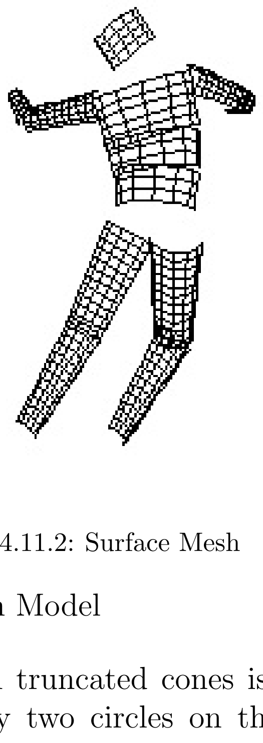

4.11.2: Surface Mesh

Figure 4.11: Human Model

A 3D outer shape model which is formed with truncated cones is added to describe the surface. Each truncated cone is described by two circles on the two ends, and a mesh formed by points sampled from the cone surface (Figure 4.11.2). As more points are sampled, more computation will be required during tracking. Therefore the sample density is made adjustable, so that we can balance the model accuracy and the computation time.

4.3.2 Projection Model

When weighting the particles, we need to compare the predicted configuration with the image data. The projection model, which describes how the camera will project points from 3D to 2D, is therefore required. Using the videos and the corresponding ground truth data of the human motions, we obtained the projection model through non-linear fitting. Through the use of the non-linear projection model, lens distortion is also implicitly rectified.

First, the projection model is derived theoretically to determine the parameters that need to be estimated. Knowing the approximate relationship between the world frame and the camera frame, we model the projection through a series of transformations. In this case, the theoretical model is non-linear (Equation 4.14). The second step is to find a large set of points in the 3D space and their corresponding positions in the 2D image. 100 matches covering the tracking volume are found manually. The last step is to obtain the parameters through non-linear least square fitting.

$$\begin{cases} x^I = \frac{a_1 x^W + a_2 y^W + a_3 z^W + a_4}{a_5 - z^W} \\ y^I = \frac{b_1 x^W + b_2 y^W + b_3 z^W + b_4}{b_5 - z^W} \end{cases} \tag{4.14}$$

This projection model is validated by comparing the projection results of the human model with different poses against the corresponding images, and it is shown to perform well overall by visual inspection. Since the sampled points are not uniformly distributed throughout the space, the model accuracy depends on where the point to be projected is located.

4.3.3 Computation Overview

To achieve faster processing speed, the entire system is implemented in C++, using the OpenCV library. The system is initialized using ground truth. Since we are using a partitioned particle filter, the prediction, likelihood calculation and re-sampling are performed in each partition for each frame. After the evaluation for the last partition is completed, all the particles are re-evaluated considering the full body to calculate the final weights. At the end of the processing for each frame, the estimated pose of the actor is generated by calculating the expectation over all the particles.

For simplicity, we use a zero order model for prediction, which means the state in the next frame is predicted by adding a Gaussian noise around the previous estimated state. Joint angle limits are considered, and work as reflective surfaces. The predicted joint angles that exceed the limits are reflected back into the allowed region with the same distance from the limits.

The weight calculation is the most important component of a particle filter. Assuming the use of a stereo camera system as input, we develop an explicit depth cue for evaluating the particles. Without having actual data from a stereo camera, we simulate the stereo images by generating virtual depth images off-line.

The virtual depth images are generated using our human model driven by the ground truth MOCAP data (Figure 4.12). The human model is then projected onto the image plane according to the projection model. The depth map is formed by the front surface of the human model, which is calculated by weighted averaging of the depth of the nearest vertices. Occlusion is considered by always updating the depth of a pixel with the smallest depth value (Z-buffering). Finally, the image is smoothed and Gaussian noise is added to simulate the noise which would occur during actual sensing. The precision of the virtual depth image is about 2cm, which is comparable to commercially available stereo cameras.

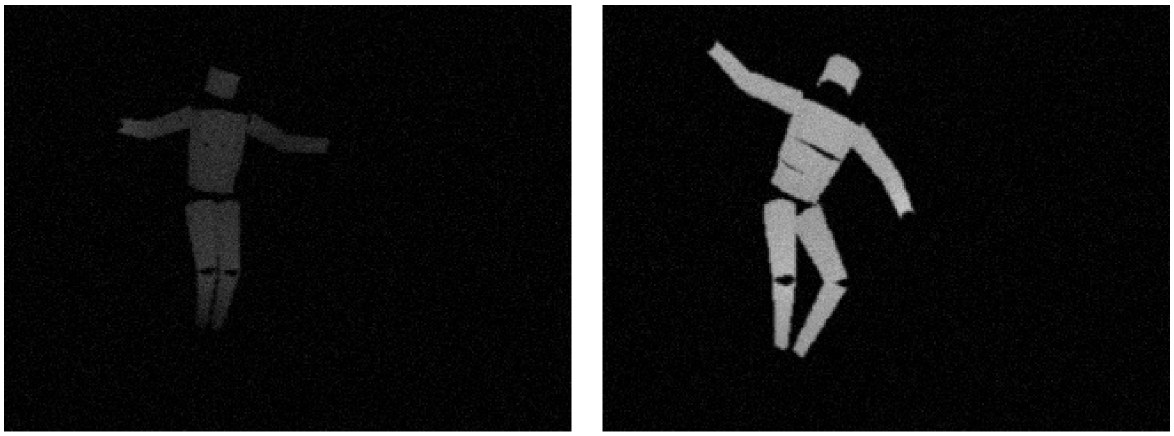


Figure 4.12: Virtual Depth Images

For each time step, the particle weights are calculated from the depth cue, the edge cue, the colour cue, and the distance cue, described in Section 4.3.4 below. After the weight calculation, some particles will have large weights while many others will have very small weights, leading to the degeneration problem [73]. Re-sampling is used to redistribute the particles in the search space while maintaining the PDF, so that more particles are concentrated in the peaks. In the partitioned particle filter, re-sampling is performed after the likelihood calculation for each partition and after the final likelihood calculation by considering all the partitions. The systematic re-sampling approach is adopted [84, 94]. At the end of the processing for each frame, the estimation is generated by calculating the expectation of the configuration distribution.

4.3.4 Weight Calculation

The weight for each particle is calculated by comparing the predicted human model contained in that particle with the observed image data using different cues. For each cue, we need to extract the corresponding information from the image, and calculate the distance between the predicted features and the observed features. Since in different cues, the scales of the distances are different, we rescale all the distances into unit scale ranging from 0 to 1 linearly. Then the weight is calculated from the distance using Equation 4.15:

$$w = A^{-d} \tag{4.15}$$

In Equation 4.15, A determines the survival rate of the particle filter in this cue [34]. In our experiment, A is set to 100 and the average survival rate is around 0.5.

Image Pre-processing

First, foreground segmentation is performed to reduce background noise. Assuming the background image is available and the camera is static, a standard background subtraction technique [95] is applied on the colour image in the RGB space. The subtraction and thresholding are performed separately for R, G and B dimensions. The final foreground mask is obtained by combining the foreground masks obtained from all three colours through Exclusive Or operation. However, because of the complexity of the background, there are many holes in the foreground mask after doing the subtraction. Dilation is performed to fill those holes.

If the camera is moving, for instance in the case of being mounted on a robot head, this background subtraction technique will not be suitable. In this case, if we have good depth information, foreground segmentation can be conducted completely based on depth, and this method will tolerate ego-motion. Since we are using virtual depth images, our implementation here is still based on background subtraction in the colour space.

The Depth Cue

Different from the implicit stereo used in [35], here we propose an explicit depth cue, in which we use the depth image generated from a stereo system directly. Using the foreground segmentation result, we extract the foreground depth map. The distance for the depth cue is calculated by comparing the projected front surface of the predicted configuration with the foreground depth map. For each sampled point, if it is projected within the foreground mask, the distance is the absolute value of the depth difference. If it is not, the distance is set to a fixed large value as a penalty, which is 2.8m in our experiment. The penalty value should not be too large, otherwise the depth cue will degenerate into a region cue and lose its advantage in providing depth information. The distance for a partition is the summation of the normalized distances for every body part in that partition (Equation 4.16).

$$d^{De} = \sum_{i=0}^{N} \frac{\sum_{j=0}^{M_i} d_{ij}}{M_i} \tag{4.16}$$

In Equation 4.16, d_{ij} is the depth difference for the jth sampled point of the ith body part in this partition, M_i is the total number of sampled points in body part i, and N is the number of body parts in this partition.

Because this depth cue needs to compare surfaces, it is not very time efficient. However, the depth cue is very helpful for full body tracking in the single view camera setting since it employs the important depth information. This will be demonstrated in the Section 4.3.6.

The Edge Cue

The distance from the edge cue is calculated in the same way as described in Section 4.2.3 ***Weight Calculation***, except that only the body parts in the current partition are considered. All the body parts are considered in the final particle weight re-evaluation for each frame.

The Colour Cue

The colour cue is typically based on template matching, using the learned colour histogram of each body part. However, we argue that generating the colour histogram for each particle takes too much time, because each pixel covered by the predicted silhouette must be considered for every particle.

Instead, based on the sampled points we already have in the human model, we directly use the colours of these sampled points in the initial frame as the template. The Euclidean distance is calculated from the colour of the predicted configuration to the colour template in the RGB space Equation 4.17.

$$d^C = \sum_{i=0}^{N} \frac{\sum_{j=0}^{M_i} \sqrt{(r_{ij} - r'_{ij})^2 + (g_{ij} - g'_{ij})^2 + (b_{ij} - b'_{ij})^2}}{M_i} \tag{4.17}$$

In Equation 4.17, r_{ij}, g_{ij} and b_{ij} are the intensity of the red, green and blue colour of the jth sampled point in the ith body part in the predicted human model, and r'_{ij}, g'_{ij} and b'_{ij} are the values in the template, M_i is the number of sampled points in the ith body part, and N is the number of body parts in this partition.

The Distance Cue

During full body tracking, small body parts such as the forearms and the lower legs are the most difficult to track. To improve tracking performance, we use an additional cue, the distance cue, which is composed of two parts, the ***Blob Distance*** and the ***Ground Distance***. For the Blob Distance, we make use of the two black wrist bands worn by the actor to indicate the hand locations. This does not only help to localize the arms, but also provides a clue for where the torso is from the bottom-up [69]. For each arm, the ***Blob Distance*** is calculated as the Euclidean distance between the predicted position of the wrist and the corresponding detected band position. For the torso, we consider both wrists at the same time. In addition, we make the assumption that for most of the time, the feet of the actor are on the ground. The distance is calculated from the feet to the ground. Similar to the ***Blob Distance***, the ***Ground Distance*** not only helps to localize the legs, but also helps to localize the torso. For the cases when the feet are below the ground, which is obviously impossible, we set a large value as a penalty. For the ***Ground Distance***, we make no specific assumption about the appearance of the feet, so the actor's footwear will not affect the result.

Integration

The final weighting function for the lth particle in the ith partition is obtained by integrating these cues using Equation 4.18.

$$w_{li} = (w^E)^\alpha \cdot (w^{De})^\beta \cdot (w^C)^\gamma \cdot (w_B)^\delta \cdot (w^G)^\theta \tag{4.18}$$

In Equation 4.18, w_{li} is the final weight for the lth particle in the ith partition, w^E, w_{De}, w^C, w^B, and w_G are the weights calculated from the distance from the edge cue, the depth cue, the colour cue and the distance cue respectively, and α, β, γ, δ and θ which are within $[0, 1]$ are their weights.

4.3.5 Experimental Results

The system is tested using challenging videos, in which the actor performs random and fast motions. The system is shown to be capable of tracking accurately and robustly with good processing speed.

Tracking Video

Figure 4.13 shows the images captured when tracking a video of 1123 frames. In the video, the actor performs a variety of dancing movements, including 11 jumps accompanied by raising arms (Figure 4.13.10), half squatting down (Figure 4.13.20), and turning (Figure 4.13.30). In the experiment shown, 1000 particles are used for the torso, 200 for the head, 500 for each upper arm, 200 for each forearm, 500 for each thigh and 200 for each calf leg. For each cone, the number of sampled points on the front surface ranges from 15 to 75, proportional to the area of the front surface of the cone.

As can be seen from Figure 4.13, the tracker performs well through challenging tracking scenarios, where the actor's movements are random and fast. In the video, the actor moves his body parts through a large range without any constraints. The actor is not restricted to be always directly facing the camera. In the last part of the video, the actor rotates his body about the vertical axis almost 120 degrees. The tracker maintains tracking, although the accuracy decreases due to the occlusion. The accuracy is also impacted when there are ambiguous motions. For example in Figure 4.13.3, the forearm is perpendicular to the image plane, and this is difficult to determine from a single view. An additional difficulty is the fact that most of the motions are very fast. For example, there are only 17 frames for the jumping motion between Figure 4.13.6 to Figure 4.13.8, during which the position and the posture of the actor changes significantly. The tracker is able to successfully track through most of these situations smoothly. Even when the tracker loses track of some body parts, it can recover from the incorrect configuration quickly. This can be seen in Figure 4.13.12 to Figure 4.13.15, where the tracker momentarily loses track of both arms because of the fast movement, but recovers again after 80 frames, demonstrating the robustness of the system.

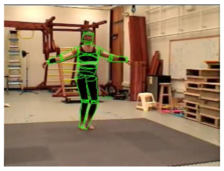
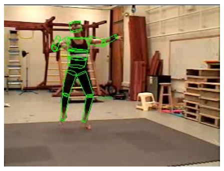
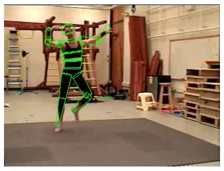
Quantitative Error Analysis

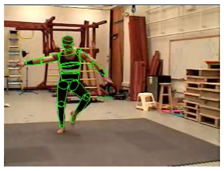
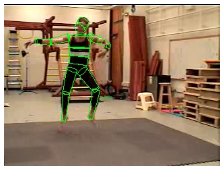
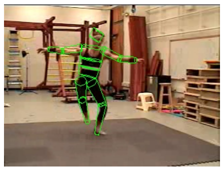
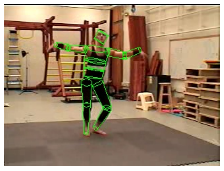
The error metrics used in the following analysis are defined in Section 4.2.4 ***Quantitative Error Analysis***. In all the following experiments, each test is run for five times.

One of the most basic problems when using a particle filter is how to determine the number of particles. For a partitioned particle filter, the problem is exacerbated because there are several partitions and each can have a different number of particles. Intuitively, the partition with more DOF and closer to the root of the hierarchical structure should have more particles. Empirically, we found that using 1000 particles for the torso, 200 for the head, 500 for each upper arm and 200 for each forearm, 500 for each thigh, 200 for each calf leg, produced good results with an ***Overall Average Error*** of 19.1cm and standard error deviation of 12.9cm. These values are used in the following experiments. By using this number of particles, it takes 1.1 second to process each frame. However, the processing speed can be improved by reducing the number of particles. When half the particles are used, it takes 0.67 second for each frame, with an ***Overall Average Error*** of 20.7cm and standard error deviation of 13.3cm. If using one fourth of this number, the frame rate can achieve 4Hz with an ***Overall Average Error*** of 24.3cm and standard error deviation of 15.9cm.

Then, we obtained the ***Joint Average Error*** and the corresponding standard deviation, as plotted in Figure 4.14. The joints are the root (R), the thorax (T), the head (H), the two shoulders (RS and LS), the two elbows (RE and LE), the two wrists (RW and LW), the two hips (RH and LH), the two knees (RK and LK), and the two ankles (RA and LA). From Figure 4.14, we can conclude that the error increases along each kinematic chain. This is because we used more particles for the partitions that are closer to the root, and also the body parts become smaller along the kinematic chain, which makes them more difficult to track. Also, the errors for the ankle joints are larger than the wrists. The reason is that the blob distance in the distance cue is more powerful than the ground distance.

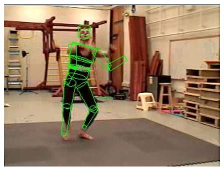
Next, the ***Average Error*** over frames is generated to help us to determine how the

4.13.1: Frame5

4.13.2: Frame110

4.13.3: Frame327

4.13.4: Frame332

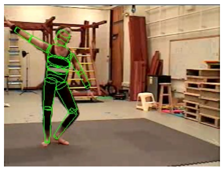
4.13.5: Frame352

4.13.6: Frame396

4.13.7: Frame403

4.13.8: Frame413

4.13.9: Frame499

4.13.10: Frame510

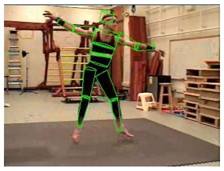
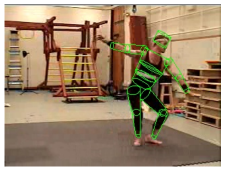
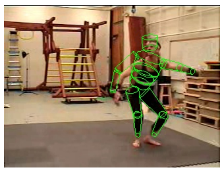
4.13.11: Frame571

4.13.12: Frame579

4.13.13: Frame584

4.13.14: Frame652

4.13.15: Frame660

4.13.16: Frame676

4.13.17: Frame686

4.13.18: Frame720

4.13.19: Frame729

4.13.20: Frame792

4.13.21: Frame851

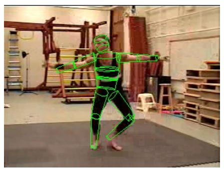
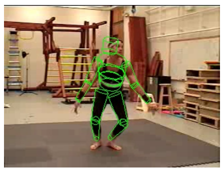
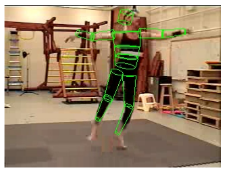
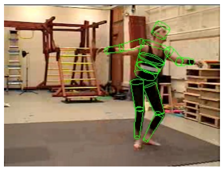
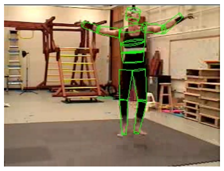
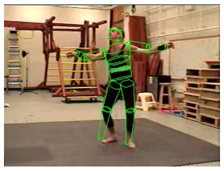
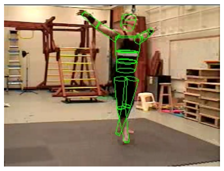
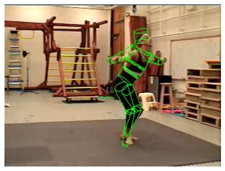
4.13.22: Frame874

4.13.23: Frame882

4.13.24: Frame931

4.13.25: Frame940

4.13.26: Frame967

4.13.27: Frame1022

4.13.28: Frame1045

4.13.29: Frame1073

4.13.30: Frame1097

Figure 4.13: Frames Extracted from Video

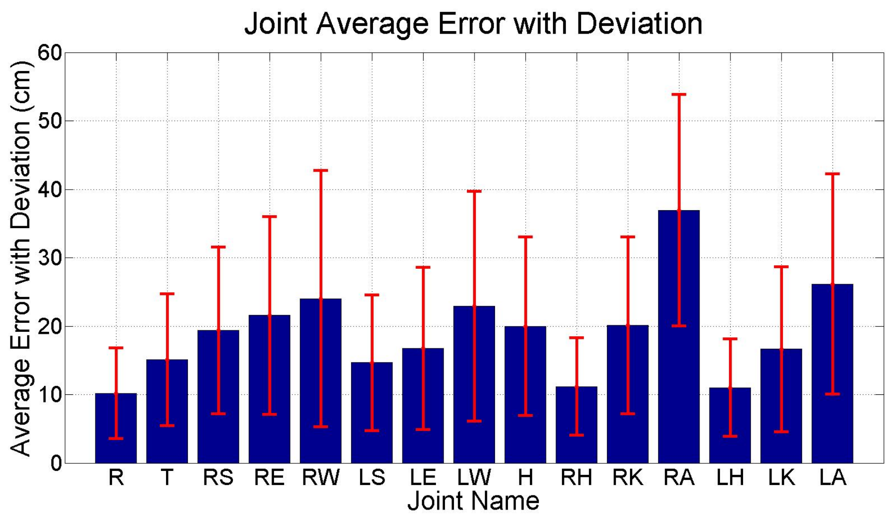


Figure 4.14: JAE with Deviation for Full Body

Table 4.3: Cue Comparison (Errors [cm])

	All Cues	No Edge	No Depth	No Colour	No Dist.
Error	16.4	14.6	34.8	14.1	19.0
Std Dev	11.7	9.6	23.0	10.7	11.7

tracker performs as time proceeds. The Average Error is plotted in Figure 4.15. From this figure, we can see that as time proceeds, the Average Error increases, with several peaks. In the video, the actor performs very small movements in the first 300 frames, followed by 11 sequential jumps which approximately correspond to one peak each in the figure. For example, there is a jump at around frame 400, and another at around frame 590. Starting from frame 850, the actor performs several jumps with body rotation, and the error also increases. From these observations, we can conclude that the error is greater when the actor is performing fast, strenuous, and complex movements.

We also tested the tracking performance when using different cues, to determine the relative importance of each cue. In this experiment, we first test the system using all the four cues, and then test excluding one cue to see how this cue affects the tracking. The system is tested using a shorter clip from the video used above, and the ***Overall Average Error*** with standard deviation is shown in Table. 4.3 (Errors reported in [cm]).

Surprisingly, the average error is low when the edge cue is excluded. However, we cannot judge the tracking based on the quantitative error alone, but should also consider

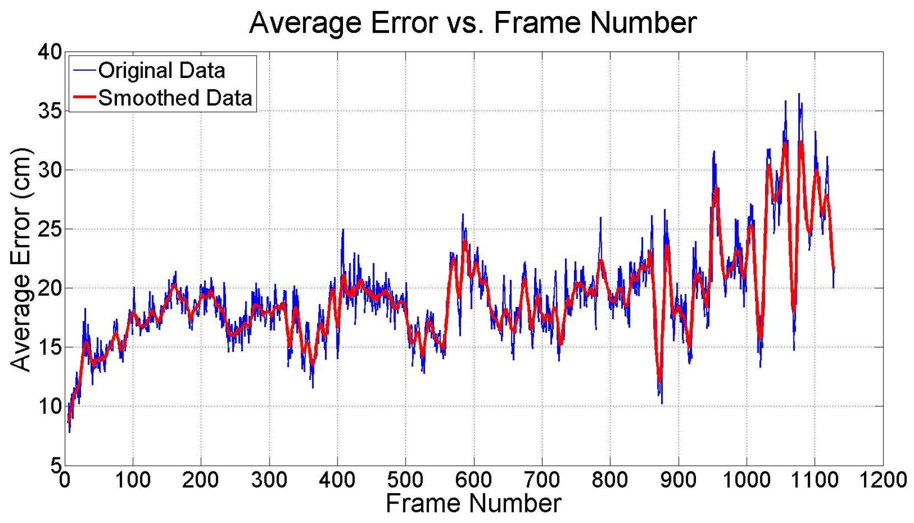


Figure 4.15: AE Over Frames for Full Body

the correctness from visual inspection. When the edge cue is not used, the two legs usually get overlapped when they are close to each other, and the tracking of the legs is incorrect (Figure 4.16), introducing large errors in the leg position. However, tracking of the upper body appears improved because the unusual position of the legs introduces an additional constraint at the waist. This is an artefact of the particular video clip. In order to get the correct tracking result, the edge cue is important to avoid leg overlapping. We can also see that the newly proposed depth cue is the most important cue. Without using the depth cue, the performance decreases significantly, since no other cue incorporates depth information. Adding the colour cue also decreases performance, due to the fact that regions in the background are similar to the actor's skin colour, and this confuses the tracker.

4.3.6 Discussion

The implementation of the tracking system reveals several key problems with using a partitioned particle filter for human tracking.

Full body tracking: Using the partitioned particle filter, full body human motion could be tracked in near real-time. However, partitioning also introduces additional problems, such as how to determine the number of particles and how to control the survival rate for each partition.

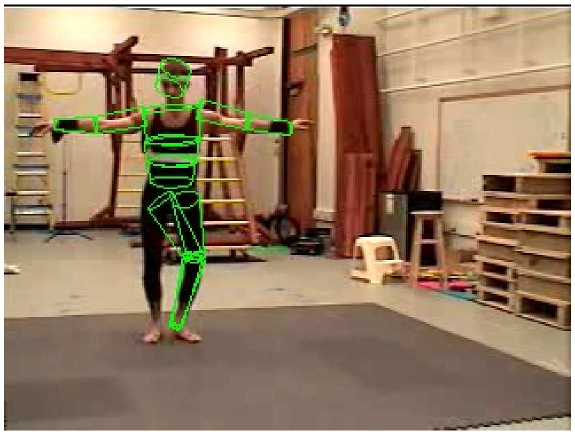


Figure 4.16: Tracking without Edge Cue

Tracking based with depth camera: Considering the application to portable tracking, the system was designed in a stereo context. Depth information is incorporated into monocular tracking by using the proposed depth cue, and the system achieves successful tracking that would be very difficult when using monocular video only. However, the system could benefit further from the stereo system, such as in foreground segmentation. On the other hand, the depth cue could become significantly degraded if the actor is operating in a cluttered environment, where objects in the foreground would affect the depth image.

Cue comparison: We also evaluated the commonly used cues and the newly proposed depth cue. From our experiment, we show that the depth cue is a strong cue which can compensate for the missing depth information in monocular tracking. However, the depth cue cannot work well alone, especially when only a limited number of sampled points is used. The edge cue and distance cue are also strong cues which require less computation.

Processing speed: To speed up the system, we applied partitioned sampling to reduce the required number of particles. For the weight calculation, only sampled points are used to describe the front surface of the human model to reduce computation. To further improve the speed, the most essential problem is how to improve the particle filter to further reduce the required number of particles. Over a half of the time is consumed in configuration and projection calculations, so further improvements could be realized by optimizing these computations. In addition, developing more computationally efficient cues would also help to speed up the processing.

Sources of error: There are several factors that lead to the tracking error.

1. The particle filter is based on a sampling method, so the tracking result cannot avoid

being jittery when a limited number of particles is used. Especially when the goal is higher processing speed, accuracy is sacrificed.

2. The cues for weighting the particles cannot distinguish certain configurations. For example, when the forearm is perpendicular to the image plane, it is almost impossible for the current system to track correctly.

3. The projection model is not always accurate. In modelling the projection, we made the assumption that the z axis of the world frame is perpendicular to the image plane, although there is a perceptible angle deviation. In estimating the parameters, the sampled points are not uniformly distributed in the work space. So the projection model accuracy also depends on area of the workspace in which the actor is currently located.

4. The image processing introduces some error. The background is complex and contains regions of similar colour with the foreground. When doing background subtraction using colour, the foreground is corrupted. This directly affects the edge extraction by losing some edges.

Moreover, we did not consider changes to the environment, for instance lighting brightness. If the brightness changes, the background subtraction and the colour cue will be affected, because both of them are done in the RGB space. Also, if a significant part of the human body is not seen by the camera for a long time, the tracking result would be degraded, because the configuration of the unseen body parts will be impossible to estimate and the resulted distribution will be random, and this will affect the configuration estimation of the whole body. If only the upper body is visible, tracking based on a partial model of the body can also be applied, as described in Section 4.2.

4.3.7 Summary

In this section, we propose a new marker-less full body human MOCAP system based on stereo input. Human motion is tracked by a particle filter with partitioned sampling to deal with the high dimensionality problem for full body tracking. In addition to the edge cue, the colour cue and the distance cue, we propose a new explicit depth cue to utilize the stereo information. To enable a quantitative error analysis for the algorithm performance, we use videos from the publicly available CMU MOCAP dataset, and generate virtual depth images off-line using the 3D human model driven by the ground truth data. The system is tested with challenging videos, and the results demonstrate that our system is capable of tracking random and fast motions accurately and robustly in near real-time. From the comparative cue study, the importance of using good depth information and higher-level image features is revealed. In next chapter, the problem of detecting body part from depth data will be explored.

Chapter 5

Endpoint Body Part Detection[1]

5.1 Chapter Introduction

The previous chapter demonstrates that the localization of the extremity body parts, *i.e.* the head, the hands and the feet, provides a strong cue for the inference of the full body configuration. Knowledge of these locations reduces the search space significantly, helping to deal with the biggest difficulty in full body tracking: the high dimensionality problem. When tracking of the target is lost, the locations of these body parts also provide bottom-up information in helping the tracker escape from local minima and to recover from incorrect configurations [59, 25, 69]. In addition to being used as cues for whole body tracking, the detected locations can be used directly for gesture recognition to enable gesture based human-machine interaction. Because this class of body parts are the endpoints of a human body and have special importance, we name them as ***Endpoint Body Parts***.

In detecting the locations of the endpoint body parts, processing speed is crucial. Whether the end goal is motion capture, gesture recognition or other applications, the endpoint locations are usually used as input information for higher-level tracking or recognition, and this higher-level processing is also often required to be at frame rate. In this chapter, we propose an accurate and very efficient method to detect and identify these body parts from depth data, as well as calculating their 3D orientations.

A key insight of the proposed method is that the endpoint body parts have very distinguishable shape features. On the one hand, a common feature shared by all the endpoint body parts is that they are connected only to one other body part, and this makes their

[1]A version of this chapter has been published [96].

interior contour shapes open only to one side. On the other hand, different endpoint body part categories (head, hand and foot) have their own characteristic shape features that are distinguishable from the others. By applying a good shape descriptor on an image with clear body part shapes, we can unify the detection task and identification task within one framework.

The first question is how to obtain images with high quality body part shapes, described by the contour edges of the body parts. These edges can be both on the boundary of the human silhouette and inside the silhouette. When using traditional colour or monochrome cameras, the shape edges are often cluttered by other edges, such as the edges due to clothing pattern. Even high quality segmentation of human silhouette is a very difficult task. Moreover, the environment conditions can affect the quality of the images significantly, for instance lighting changes. To avoid these issues, we propose an approach that uses the depth images generated from a Time-of-flight (TOF) sensor, which preserves most of the shape edges, while removing most of the cluttering edges.

The next question is what a good shape descriptor is for this application. Inspired by the Shape Context concept proposed by Belongie et. al. [44], we propose a novel ***Local Shape Context*** (LSC) descriptor specifically for describing the shape features of body parts. This descriptor describes the local shape of different body parts with respect to a given reference point on the human silhouette, and is shown to be very effective at discriminating and classifying endpoint body parts, as well as being computationally efficient. A new type of interest point is defined based on the LSC descriptor, which is called Interest Reference Point (IR Point). A hierarchical IR Point selection algorithm is designed to further conserve computational resources. The detected endpoint body parts are then classified according to learned models of each class based on the LSC feature. The proposed technique is tested using a publicly available dataset [97] and achieves around 85 percent precision and recall for each class at a speed of 100Hz on a Quad Core PC.

This chapter is structured as follows: in Section 5.2 we propose our LSC descriptor based approach for localizing the body parts, and present the algorithm design. We show the experimental results in Section 5.3, and provide both a quantitative and qualitative analysis of the system performance. Further discussion can be found in Section 5.4, and the chapter is concluded in Section 5.5.

5.2 Algorithm Design

5.2.1 Local Shape Context

In [44], Belongie *et al.* propose a Shape Context descriptor for measuring shape similarity. The idea is that among the points on the shape contour, the vectors from a reference point to all the other points contain rich global shape information with respect to that point. Assuming the edges that describe the shape have already been found and sampled into a number of points, a histogram that statistically summarizes the distribution of all these vectors is defined as the Shape Context for the reference point. The shape of the entire contour is described by the Shape Contexts of all the sample points. This descriptor is shown to be very effective for object recognition, but it is computationally expensive. It is inherently invariable to translation and small deformation, and it can also be made invariant to rotation and scaling. Later, Mori *et al.* applied the shape context descriptor to estimate human body configurations [14], but the need for exemplars severely limits its application.

When considering body part detection and identification, however, the original Shape Context descriptor is not suitable. The biggest problem is that this descriptor encapsulates the global shape information at each point, but since a human body is a highly articulated object, the global shape and the location of the endpoint body parts can vary considerably. Even if a localized, window-based version of the shape context descriptor is used, it is difficult to exclude extraneous edges arising from adjacent body parts in a window. Furthermore, although the calculation of the Shape Context is not time consuming, the matching of the points can be very slow. The total cost of the matching for every point to every point must be minimized by comparing 2D histograms, and the computation cost can be high.

Inspired by the Shape Context concept, we propose a new shape descriptor for body part detection and identification, which we name the Local Shape Context (LSC) descriptor. This descriptor is defined with respect to a reference point within the human silhouette, and it describes the local shape around that point. A LSC descriptor consists of a set of vectors from the reference point to the points on the nearest edges in radial sampled directions (Figure 5.1). If the edge is not found within a certain range in one direction, the distance will be made equal to that range. A corresponding LSC feature is defined as one vector consisting of all the lengths from the reference point to the nearest edge along each radial line in clockwise order. It does not matter which direction is used as the starting point.

This descriptor contains rich shape information about the local area, while excluding the extraneous edges of other body parts effectively. More importantly, this descriptor is concise. Compared to the previous Shape Context, the LSC Descriptor of one single point is sufficient to describe a local shape, if the reference point is well selected and the directions are sampled densely enough. Obviously, this descriptor is invariant to translation. In the following, we will show the effectiveness of applying this descriptor for detecting and identifying endpoint body parts, and discuss the ways to make it invariant to rotation, scaling and deformation.

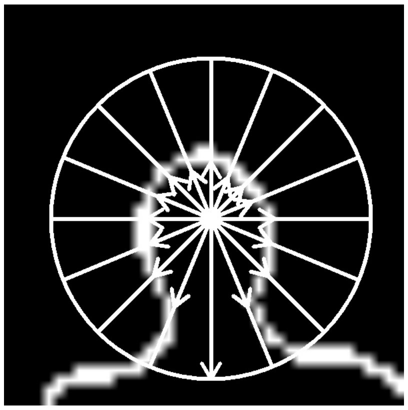


Figure 5.1: LSC Descriptor Illustrated on a Head

5.2.2 Hierarchical Interest Reference Point Selection

As described in the previous section, an LSC Descriptor is defined with respect to a reference point. A naive approach would be to sample through the entire human silhouette and compute the LSC Descriptor for each sampled point, and classify them into different categories. However, in this case, in order to ensure a good detection performance, the sampling would have to be dense globally. In addition, the computation cost of the classification would also be high, because every sampled point would need to be evaluated. To avoid these problems, a two-layer hierarchical Interest Reference Point (IR Point) selection algorithm is used to improve efficiency.

Based on the LSC concept, we define two tiers of IR Points. Tier 1 IR Points are selected based on the fact that endpoint body parts have common shape features distinguishing them from most of the non-endpoint body parts. At a point close to an endpoint body part inside the body, there is a convex edge contour which is open only to one side with a certain opening angle range (Figure 5.2). This means that, when we try to detect edges in a set of sampled directions, edges will be detected in most directions, but will not be detected in several continuous directions within a narrow range. We develop an algorithm to identify such points, these are selected as Tier 1 IR Points. Tier 2 IR Points are then formed by clustering Tier 1 Points based on Euclidean distances.

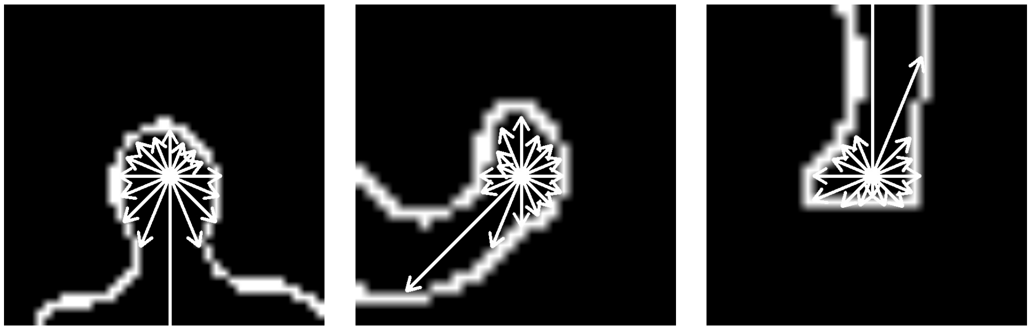
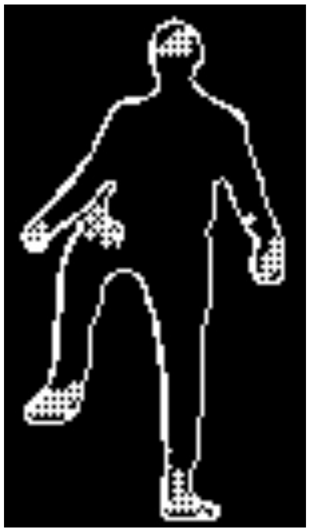

5.2.1: Head 5.2.2: Hand 5.2.3: Foot

Figure 5.2: LSC Descriptors at Endpoint Body Parts

Tier 1 Interest Reference Point Selection

We first sample over the human silhouette uniformly to identify Tier 1 IR Points. Because we just need to know whether there is an edge in a sampled direction, instead of using distance, we only use a binary value to indicate whether an edge is found within the search range. To further simplify the calculation, the radial sampling density is sparse, and the search range is relatively small. Note that 4 connection should be used if the edges could be thin. The LSC feature is thus a vector of binary values, which is actually a ring rather than a line. This means the last element in this array is connected to the first one. On this ring, we detect and count the falling edges, which are the changes from 1 to 0. Reference points with more than one falling edge are more likely not a candidate for an endpoint body part, so they are removed. Then, for the points with only one falling edge, we count the continuous 0s between the falling edge and the raising edge. The points with more than 3 continuous 0s are discarded. The remaining sampling points are the Tier 1 IR Points, shown in Figure 5.3.1. Notice that they are located mainly at the endpoint body parts, but some are also located at other locations which meet the criterion. In the figure, the human subject lifts his right leg, so a convex contour is formed near the hip, resulting in Tier 1 IR Points. Tier 1 IR Points are also found on the left forearm, because the special


5.3.1: Tier1

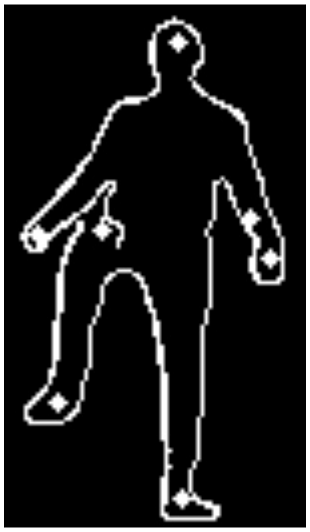

5.3.2: Tier2

Figure 5.3: Result for Tier 1 IR Points and Tier 2 IR Points

orientation of the forearm, the slightly bent wrist and the sparse radial sampling of the descriptor.

Tier 2 Interest Reference Point Formation

Once Tier 1 IR Points are identified, clustering is performed to form Tier 2 IR Points. Since some endpoint body parts may not be detectable, and some other body parts may appear to be similar to endpoint body parts, the number of clusters is unknown. Thus the commonly used K means clustering is not suitable [98]. We use a hierarchical clustering algorithm in our system, agglomerative clustering to be specific [99]. The clustering starts by assigning each point to an individual cluster. At each step, the two closest clusters are merged to form a new one. This continues until all the points are in one cluster. For every merging, the newly generated cluster is the parent of two previous ones, leading to the formation of a binary tree. At each merging, the distance between the two merged clusters is recorded as the precision level of the new cluster. After setting up the tree, given a certain precision, we can traverse the tree to search for the clustering at the required level. The distance measure between two clusters must be defined, and we use the median average-linkage, for the purpose of discarding outliers.

After clustering is performed according to a given precision, which is a distance of 10 pixels in our case, the centre of the cluster is calculated by averaging the locations of the points in the cluster. This is a Tier 2 IR Point (Figure 5.3.2). For each Tier 2 IR Point, the LSC feature is calculated at a finer level, with more radially sampled directions and a larger search range. The distance in each direction from the reference point to the nearest edge is calculated to form the LSC feature. From the LSC feature, the orientation is computed as the direction of the largest distance value, or the direction of the middle one if there are multiple continuous maximum distances. To illustrate this concept, in Figure 5.2, the orientation of the head is straight down, that of the hand is towards bottom left, and that of the foot is straight up. This orientation is very useful in aligning body parts with the model when classifying. By performing the alignment, the variance caused by rotation can be removed.

Search Range Adaptation

Note that the search range should not be a constant, but should be adaptive to the depth changes. That means when a reference point is closer to the TOF sensor, the search range should be larger, and vice versa. So before calculating the LSC feature for a given reference point, the depth value is first read and an adapted search range is calculated according to the projection model.

5.2.3 Endpoint Body Part Identification

Identification Using a Template Model

The next task is to classify the Tier 2 IR Points into the four categories, namely head, hand, foot and other. A learning based classification approach is used. The simplest model is a deterministic template model. Assuming that we know the initial pose of the human subject in each clip, then we can extract a template for each endpoint body part at the beginning of the detection, as shown in Figure 5.4. After the Tier 2 IR Points are identified in a given frame, we first rotate the IR points to the template orientation by aligning the orientation element of the template and of the IR Point, and then compute the distances from the LSC features of the Tier 2 IR Points to the templates. Based on these distances, a distance matrix is computed between the templates and the IR Points. Note that we have two other templates for each hand and foot, which are symmetric. Because the radial sampling density for generating the template is the same as for Tier 2 IR Points, the dimensions are the same. Thus Euclidean distance is used to calculate the distance. The best match is found by a greedy algorithm, which always looks for

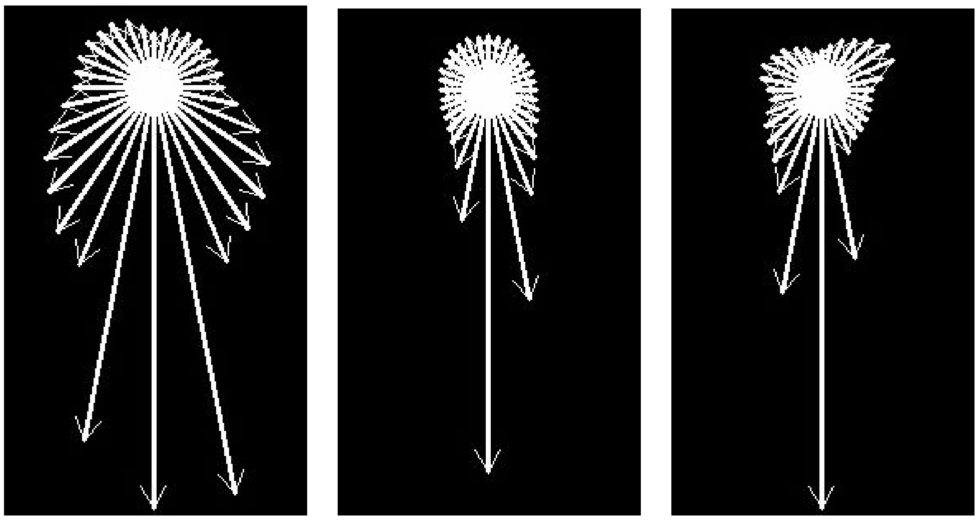

5.4.1: Head 5.4.2: Hand 5.4.3: Foot

Figure 5.4: Deterministic Template for Each Endpoint Body Part

the smallest distance for the next matching. More complicated algorithms to find better matches can also be considered, such as using the Hungary Algorithm to minimize the total distance of the matches [100]. In [27], the problem is formed as a Bipartite Perfect Matching graph problem and solved by integer programming.

Identification using a Multivariate Gaussian Model

The deterministic template model does not account for the deformation of the 2D shapes of the body parts. However, the shapes may vary significantly due to two reasons: the change of view angle, and the actual change of the endpoint shape, such as hand closing and opening. The former case can apply to all the categories, but the latter is only obvious for the hands and feet. A straightforward improvement of the identification is to use a probabilistic model instead of a deterministic template. We model the LSC feature for each category using a multivariate Gaussian model, by specifying a mean and a covariance matrix among the directions of the feature. The mean and the covariance matrix are calculated using the LSC features obtained from 27 manually selected frames which cover a variety of the possible shapes of each of the endpoint body parts. Given a Tier 2 IR Point, the probability that describes the likelihood that the IR Point corresponds to one of the categories is calculated. The same greedy algorithm is applied for finding the best match.

Using the probabilistic model significantly improves the robustness to shape variations in the hands and feet.

3D Position and Orientation Computation

After identification, the 2D locations of the endpoint body parts are obtained. Also, the orientations are described in 2D. However, it is not difficult to calculate the corresponding values in 3D. For locations, this is the inverse problem of recovering the depth images. Combined with the sensed depth value and the given projection model, we are able to transform the locations to 3D. For the orientation, we compute the depth at two points along the 2D orientation line and calculate the vector between them.

Local Sliding Window

For calculating the LSC feature for a given body part, the location of the reference point matters. The classification result is improved if the reference point for evaluating an endpoint body part candidate can be close to that of the training model. When generating the models, the reference points are approximately at the centre of each endpoint body part. Because Tier 2 IR Points are calculated by averaging Tier 1 IR Points, the location should already be close to the centre of each body part, but there is no guarantee. To improve the IR Point positioning, we apply a Local Sliding Window (LSW) around each Tier 2 IR Point, and extend the LSC descriptor of one Tier 2 IR Point as the LSC descriptors of a set of sampling points around that IR Point.

5.2.4 Processing Steps

Depth Data Pre-processing

The depth dataset used for validation is publicly available, originally generated by Ganapathi *et al.* using a Swissranger SR4000 TOF camera by MESA Imaging AG, with a resolution of 144 by 176 [25]. The dataset contains 28 clips of a variety of different motions, along with the ground truth data obtained from a marker based MOCAP system. The depth is represented by a 3D point cloud with coordinates described in the eye coordinate frame. Our first task is to recover the depth image using the projection matrix which is provided along with the dataset. A typical recovered depth image is shown in Figure 5.5.1.

After the depth images are obtained, the depth values are thresholded to generate the silhouette (Figure 5.5.2). To denoise the sensing data, we remove the background and zero depth points, and smooth the resulting image using a median filter. Next, the image is thresholded again, and a clean binary human silhouette image is obtained. We then mask the smoothed depth image using the silhouette image, to obtain the smoothed foreground depth image (Figure 5.5.3).

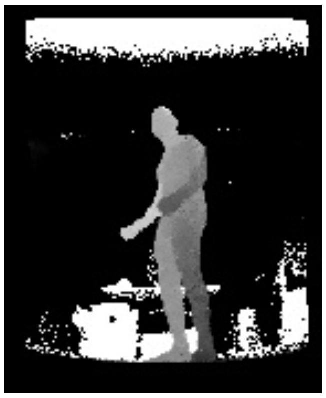
The next task is to extract the shape of the human body, which is described by the shape edges. The shape edges contain the entire contour edge, as well as the edges inside the contour due to the depth discontinuity. The Canny edge detector [91] is applied twice, to both the binary silhouette image (Figure 5.5.4) and the smoothed foreground depth image (Figure 5.5.5), but with different thresholds. The results from the two pipelines are then integrated by an "OR" operation and generate the final edge image as shown in Figure 5.5.6. We do this because we want to ensure the contour edge is continuous, and the inner shape edges are controllable by adjusting the Canny thresholds separately.

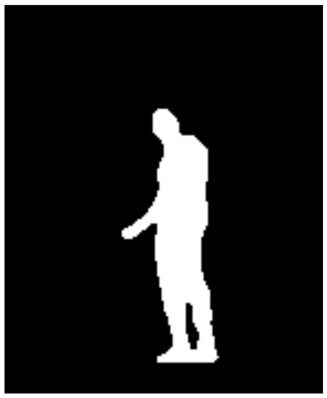
Algorithm Pseudo-code

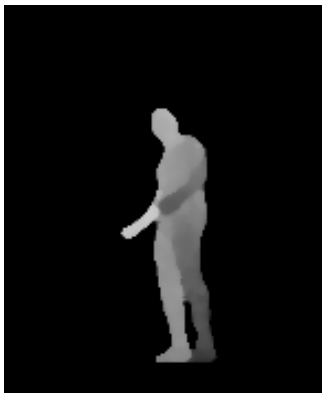
After depth data pre-processing, the approach described in Section 5.2.2 and Section 5.2.3 is implemented to achieve the endpoint body part detection and identification task. The complete algorithm is summarized in Algorithm 5.

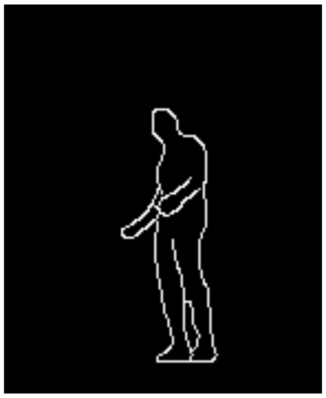
Algorithm 5 LSC Algorithm Pseudo-code

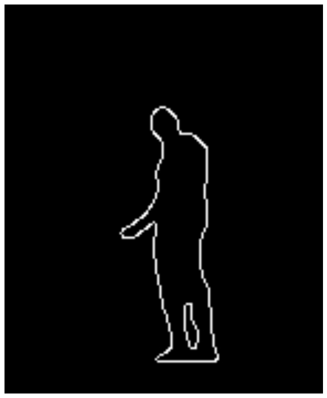
```
for frame number = first to last do
    Step1: read and pre-process the depth image (Section 5.2.4)
    Step2: sample and select Tier 1 IR Points (Section 5.2.2)
    Step3: cluster Tier 1 IR Points to form Tier 2 IR points (Section 5.2.2)
    Step4: classify Tier 2 IR Points (Section 5.2.3)
    Step5: evaluate the result using ground truth data (Section 5.3.2)
end for
```


5.5.1: Original


5.5.2: Silhouette


5.5.3: Foreground


5.5.4: Edge1


5.5.5: Edge2

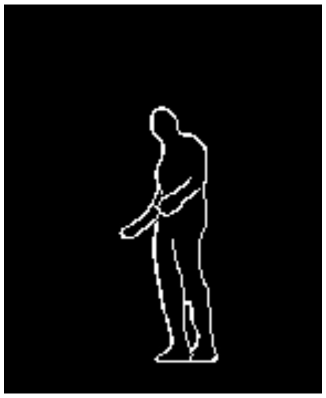

5.5.6: Edge

Figure 5.5: Preprocessing Result

5.3 Experimental Results

5.3.1 Parameter Settings

When selecting Tier 1 IR Points, the initial point sampling is carried out over a uniform grid with sampling at every 4 column by 4 row point. The radial sampling is carried out in 16 directions and the search range is set to 9 pixels before adaptation. For the LSC feature calculation of Tier 2 IR Points, the radial sampling density is set to double that of the Tier 1 IR Point density (*i.e.*, 32 directions), and the search range is also doubled. The range of the local sliding window is set to be within a 9 by 9 pixel square around each Tier 2 IR Point. Parameter selection is discussed further in Section 5.3.2.

5.3.2 Quantitative Results

The system is tested using all 28 clips contained in the dataset [97], including simple and complicated human motions. The motions include arm waving, leg raising, squatting, turning around and baseball motions. Furthermore, in some clips, when the human subject approaches too close to the sensor, the silhouette becomes incomplete in the depth image. Using the parameter settings specified above, the system requires 10 ms on average to detect and identify endpoint body parts through all 7900 frames.

We evaluate the obtained results quantitatively, using the ground truth data obtained from a marker-based motion capture system. For each endpoint body part category, we identify ***true positives***, ***false positives*** and ***false negatives***. Given a classified Tier 2 IR Point, we calculate the distance from this point to the ground truth position of the corresponding body part in 3D. If the distance is within 20cm, we identify this as a true positive, otherwise it is a false positive. When a body part is not detected in one frame, this is identified as a false negative. This procedure assumes that all the five endpoint body parts are detectable in every frame, but there are actually a few exceptions. As a result, the identified false negative rate will be slightly higher than its actual value.

We then compute the ***precision*** and ***recall*** for each classifier, which are widely used measures for evaluating classifier performance. The precision is calculated as the number of true positives divided by the sum of true positives and false positives, and the recall is calculated as the number of true positives divided by the sum of true positives and false negatives. The results using the parameter settings described in Section 5.3.1 are shown in Table 5.1.

Table 5.1: Precision and Recall

	Head	Hand	Foot
Precision	0.87	0.80	0.99
Recall	0.91	0.82	0.91

A confusion matrix is also computed to further characterize the classifier performance. Using the calculation of the false positives, if a Tier 2 IR Point A is identified as an endpoint body part, say head, but the identification is incorrect, we compute the distances from A to the ground truth of all the other body parts, *i.e.* hands and feet. If any of these distances are smaller than 20cm, *e.g.* from A to right hand, we identify this as a confusion between head and hand. The confusion matrix is shown in Table 5.2, where columns are actual, rows are predicted, normalized by the actual number. These quantitative errors show that the accuracy of the proposed approach is comparable with the state-of-the-art method, however, the processing speed is much faster (10ms for each frame compared to 60ms in [25]).

Table 5.2: Confusion Matrix (col: actual, row: predict)

	Head	Hand	Foot
Head	6343(99.09%)	7(0.07%)	1(0.01%)
Hand	58(0.91%)	10630(99.13%)	801(5.38%)
Foot	0(0%)	86(0.8%)	14089(94.61%)

We also varied the parameters to see how the performance is affected. First we altered the search range for Tier 1 IR Points, while maintaining the search range for Tier 2 IR Points as double the Tier 1 range. The results show that as the search range decreases, head detection suffers most. This is because typically the head is the largest of the three body part categories. When the range is increased to 11, the performance also worsens, because during the detection of Tier 1 IR Points, fewer sampled points have directions along which edges are not detected. Next we altered the sampling density of the points used to generate Tier 1 IR candidates. We found that as the density drops from every 2 points to every 4 points, the performance does not decrease appreciably, but the speed becomes much faster. This is because much of the processing time for each frame is used for clustering when the number of Tier 1 IR Points is large, and denser sampling increases the number of Tier 1 IR Points. Finally, we tested the effect of adding the Local Sliding Window (LSW). Before adding the sliding window, the precision and recall are shown in Table 5.3. We can see that the precision is not affected much, but the recall is much lower than when a local sliding window is used.

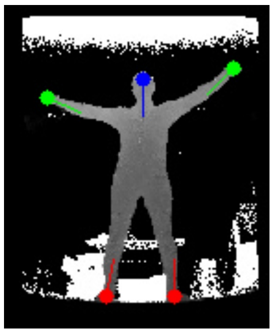
Table 5.3: Precision and Recall without LSW

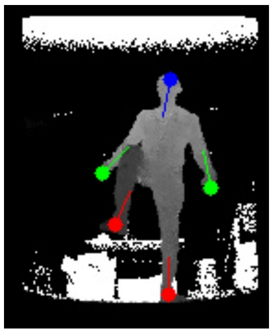
	Head	Hand	Foot
Precision	0.86	0.83	0.98
Recall	0.58	0.75	0.43

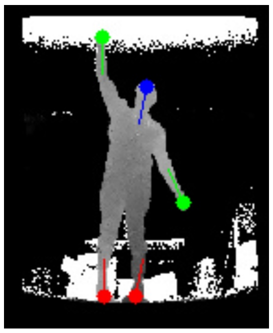
5.3.3 Qualitative Analysis

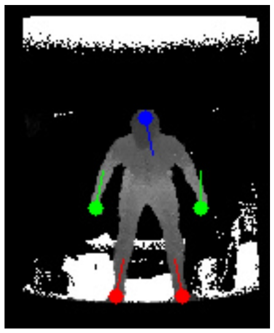
Figure 5.6 shows examples of correct and incorrect detections. Comparing Figure 5.6.2 with Figure 5.3.2, we can see that although Tier 2 IR Points are also detected on the right thigh and the left forearm, after identification using the LSC feature, those two reference points are successfully discarded. Figure 5.6.3 shows that although the 2D shape for the feet changes significantly, the system can still successfully detect and recognize them because of the probabilistic model. Figure 5.6.4 demonstrates the effect of the depth-adapted searching range; even though the head is closer to the sensor and appears to be bigger, the search range is adapted so that the head is still correctly detected. Figure 5.6.5 and Figure 5.6.6 demonstrate that even when the shape edges for endpoint body parts are inside the contour, such as the left hand in both frames, due to our edge detection method, they can still be correctly identified in these frames.

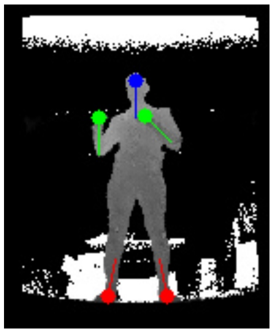
There are also some typical incorrect situations. In Figure 5.6.7, the feet cannot be identified, due to the fact that they are too close to the sensor and thus after foreground segmentation, this portion of the human body is lost. Even though the feet cannot be detected, the detection of the hands and head is unaffected. In Figure 5.6.9, the right foot is not detected, because the view angle relative to this foot has changed significantly, and this foot appearance is not included in the limited training data. This can be easily improved by using a more comprehensive data set to train the system. A similar problem is seen in Figure 5.6.11, where the left foot looks very unlike any of the training data. But surprisingly, in Figure 5.6.11 the right foot is still detected although the person turns around and the right foot is partially occluded by the left foot. Figure 5.6.9 shows the situation when the right hand is too close to the head and appears to be connected in the depth image, in this case both of these two body parts cannot be detected. This difficulty also exists in other depth image based methods, such as [68]. Although Figure 5.6.10 is a similar situation to Figure 5.6.5, the shape edges for the hands cannot be detected because the arms are too close to the torso and thus no depth edge is found. If we use a more accurate depth sensor with better resolution, this problem can be alleviated. Making the thresholding of the Canny detector adaptive can also improve performance. In Figure 5.6.12, the system confuses the left hand and the head, because of the shape changes due to occlusion and deformation.

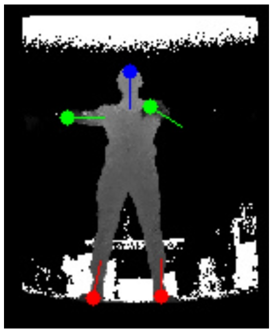

5.6.1: Correct1

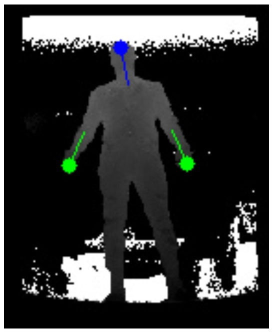

5.6.2: Correct2

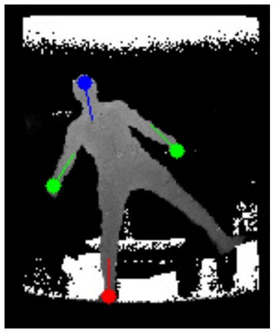

5.6.3: Correct3

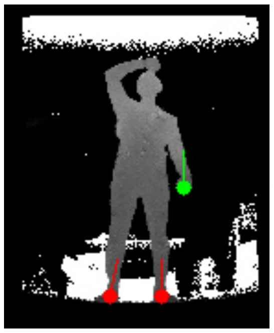

5.6.4: Correct4

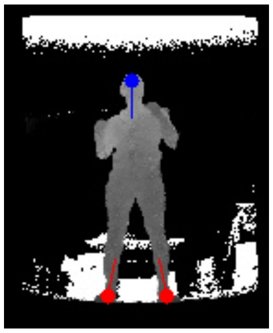

5.6.5: Correct5

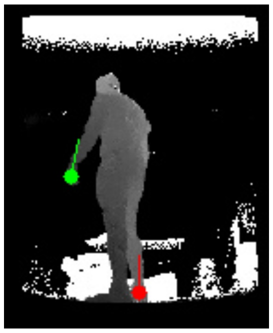

5.6.6: Correct6


5.6.7: Incorrect1


5.6.8: Incorrect2


5.6.9: Incorrect3


5.6.10: Incorrect4


5.6.11: Incorrect5

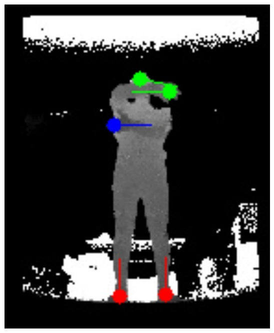

5.6.12: Incorrect6

Figure 5.6: Selected Correct and Incorrect Frames

5.4 Discussion

As seen from the experimental results, the proposed approach is effective and efficient at detecting and identifying endpoint body parts from depth images. The newly proposed LSC descriptor contains rich shape information about body parts, and is very easy to compute. It is invariant to translation, and through the orientation computation, adapted searching range, use of probabilistic models, it can be made at least partially invariant to rotation, scaling and deformation. Combined with the hierarchical IR Point selection algorithm, the system is able to localize the endpoint body parts at a very fast rate with high precision and recall rates. Furthermore, the system does not take any temporal information into account, so the output of our system can be seen as raw sensing data, to be used as input for solving tracking problems.

However, there are also several factors which degrade performance:

1. Our detection and identification is shape based, so when the shape of one body part looks very similar to another, this approach may fail to distinguish between them.

2. The quality of the depth images affects the performance significantly. Especially when the endpoint body parts are inside the body silhouette contour, the detection is completely based on the depth discontinuity. If the sensor is very noisy or the resolution is too low, the shape edges inside the contour cannot be found.

3. Depth image based body part detection suffers when the body part to be detected is too close to other body parts. A potential solution to this problem is to also incorporate other cues to compensate for the edge losses. For example, if the shape edges are very clear in colour image, they could also be combined into the final shape edge image.

4. Since our approach is completely shape based, it does not incorporate any other knowledge, but other sources of information can be very helpful. For example, if temporal information is used to filter the detection result, many confusion errors will be eliminated. Also the knowledge of the location of the entire human body and of the human motion can also help to eliminate mislabelling.

5.5 Summary

For both motion capture and gesture recognition applications, localizing the endpoint body parts is very important. In this chapter, a new method for fast detection and classification of endpoint body parts is proposed, based on a novel image feature describing the local shape near the body parts of interest. An efficient algorithm was developed for locating and classifying LSC features, using edge images extracted from depth data. The proposed algorithm is computationally efficient and shows excellent classification performance at 100Hz, with a high precision and recall.

Chapter 6

Full Body Tracking based on Body Part Detection

In this chapter, a human pose tracking system using monocular depth sequences is proposed, integrating the body part recognition algorithm (Chapter 5) into the partitioned particle filter framework (Section 4.3). The Local Shape Context (LSC) descriptor is generalized to detect not only the endpoint body parts, but also the torso and the limbs. A simplified skeleton model is designed to improve computational efficiency of the forward kinematics. To validate the final system, a new dataset is collected from a synchronized Kinect sensor and a commercial marker based Motion Capture (MOCAP) system. Thorough experimental results, analysis and discussion are provided in Section 6.5.

6.1 Body Part Detection

In Chapter 5, a method is proposed for detecting and identifying endpoint body parts using a Time-of-Flight (TOF) depth sensor. In this chapter, we use the Kinect sensor to generate depth data. A depth map from the Kinect sensor is different from that of a TOF sensor in terms of higher resolution (640 by 480 compared to 144 by 176), lower Salt-and-Pepper noise, blurrier object contour, and missing depth in dark coloured or reflective regions.

To deal with the new sensor data characteristics and to further improve efficiency, several improvements and adaptations are made to the original Local Shape Context (LSC) descriptor proposed in Chapter 5.

6.1.1 Using Depth Difference Directly

In the original algorithm, edges are first extracted from depth maps, and the interest reference points are selected based on the LSC descriptor calculated from the edge map. The edges are detected by applying the Canny algorithm, which requires the setting of two thresholding parameters. However,the selection of these two thresholds is not intuitive. Also, the Canny algorithm is based on the magnitude of image gradients, without directional information. However, directional gradient information is important, because a sharp depth increase usually indicates an object boundary, while a depth decrease is very likely being caused by an occlusion. Therefore, we propose to use depth difference directly in detecting body parts. If the depth increase is greater than a threshold value along a direction, we say a sudden depth change is detected. This makes the setting of the thresholds more intuitive, and can contain directional information.

6.1.2 Statistical Features

In the original algorithm, the selection of Tier 1 IR Points was based on continuous radially sampled directions where edges are detected within a certain range. This assumption is too strong when a Kinect sensor is used, since the object shapes are more blurred. Therefore we propose to detect Tier 1 IR points in a statistical way. First, a feature vector $V1$ is formed by stacking the binary values for all the radially sampled directions (clockwise), indicating whether a sharp depth increase is detected or not in each direction. Then a mean value $m1$ is calculated over all the dimensions in the vector; it encapsulates the possibility of hitting sharp depth increases in all sampled directions. Since it is a mean value, it is invariant to orientation. This operation can be seen as a per-pixel transformation on a depth map, which responds to convex shapes. Within a certain scale, the more convex a shape is, the stronger the response will be. A dense $m1$ map for the initial pose is illustrated in Figure 6.1.

To generate this result, 36 directions are sampled radially, with a search range of 0 to 180mm in each direction. The depth differences are calculated every 4 pixels. In Figure 6.1, we can observe that the $m1$ values at the endpoint body parts, i.e. the head, the hands and the feet, are highest. The limbs have lower $m1$ values, and the torso has the lowest. There is a dark region in the torso with 0 $m1$ values. This is because the scale of the torso is larger than the 180mm search range.

Although $m1$ captures some shape information around one pixel, the distribution of 0s and 1s in $V1$ is not described by just a mean value. For example, for the same $m1$ value, the

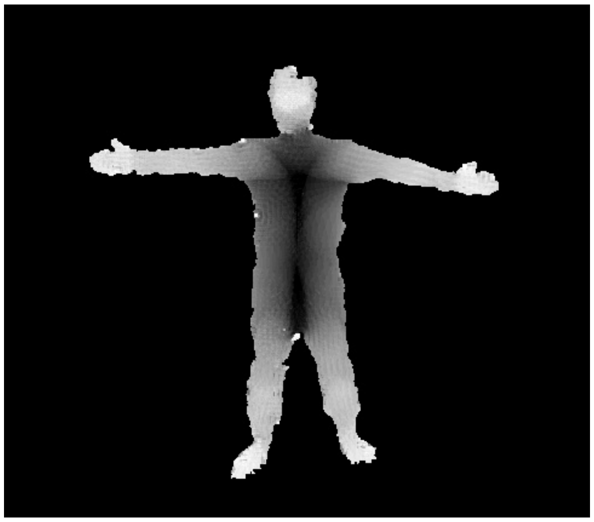
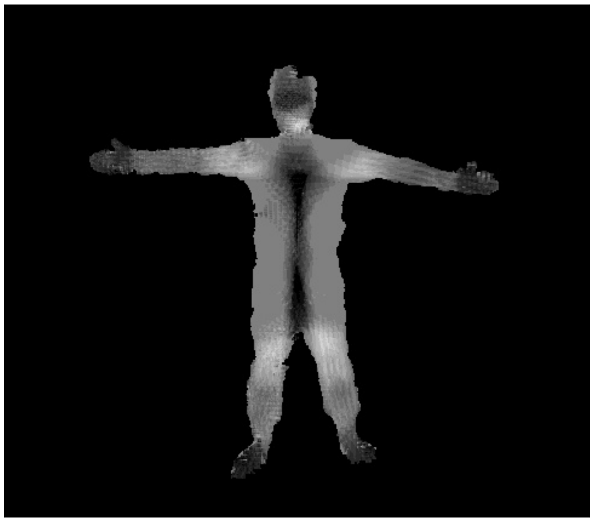


Figure 6.1: Dense $m1$ Map

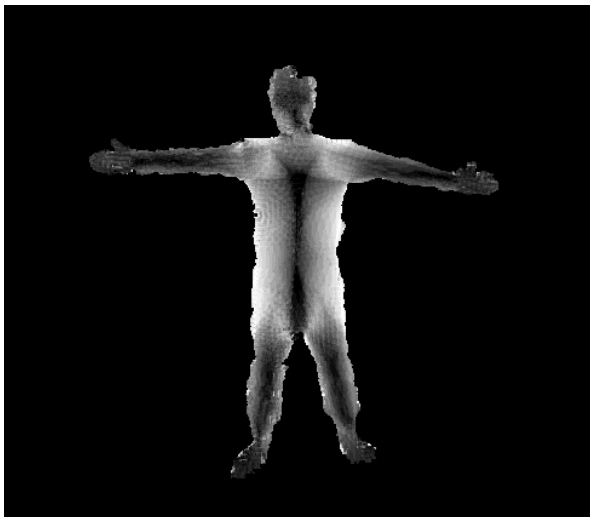
1s could be either continuously distributed or separated by 0s. Different distributions of 0s and 1s are also discriminative between different local shapes. To describe the distribution, other feature vectors can be computed based on directional correlations. Exclusive Or Operator can be applied to each slot with the slot with a certain offset, and a mean value can be further computed. We calculated a $V2$ vector using a quarter circle offset, and a $V3$ vector with a half circle offset to demonstrate this idea. $m2$ and $m3$ are computed as average values of $V2$ and $V3$ respectively. Dense $m2$ and $m3$ value maps are shown in the Figure 6.2.

We can observe that different body parts respond differently to each of these two transformations. For instance, in Figure 6.2.1, the limbs and the neck respond strongly, while the endpoint body parts appear darker. In Figure 6.2.2, the torso responds strongly, while the limbs appear darker. Theoretically, more vectors and mean values can be calculated using different offsets, and even with different search ranges. The mean values can form a new feature vector, which will be discriminative among different body parts. But in our case, a thresholding on $m1$ already provides good segmentation of body parts, not only for endpoint body parts, but also for the limbs. Segmentation results for the initial pose are shown in Figure 6.3, using 0.5 as the threshold for the limbs and 0.85 as the threshold for the Endpoint Body Parts (EPBP). Additional results are shown in Figure 6.4.

From Figure 6.3 and Figure 6.4, we can see that generally speaking, the limbs are well segmented, even when the arms are in front of the torso (Figure 6.4.1, Figure 6.4.3 and

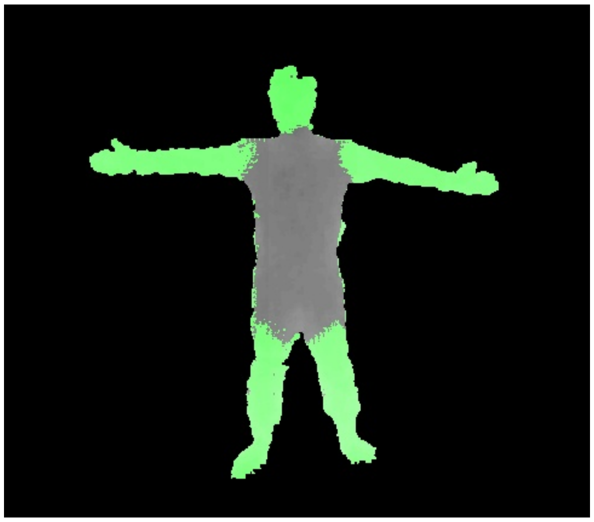


6.2.1: Dense $m2$ Map

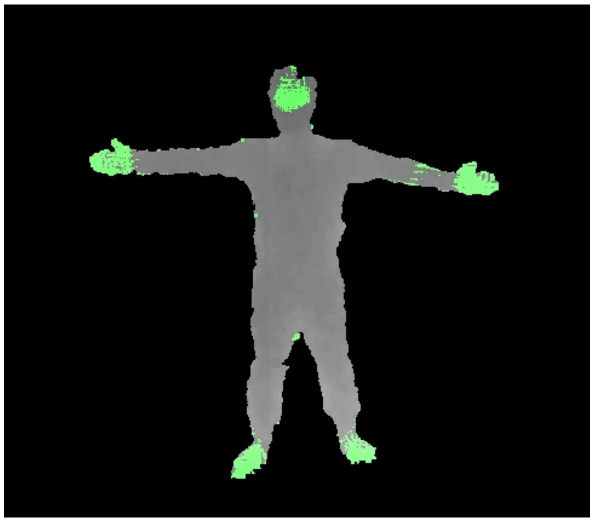


6.2.2: Dense $m3$ Map

Figure 6.2: Dense $m2$ and $m3$ Maps

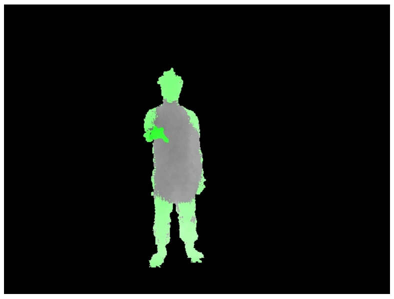


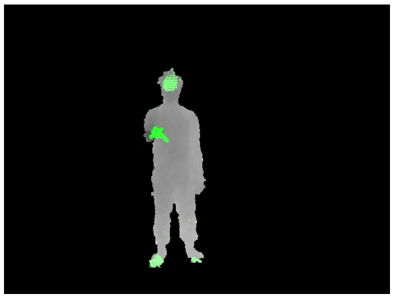
6.3.1: Limbs

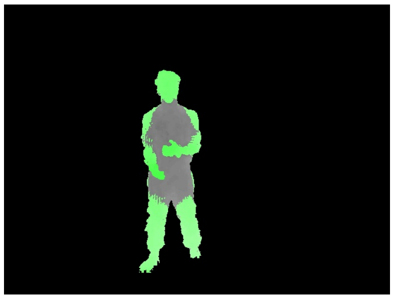


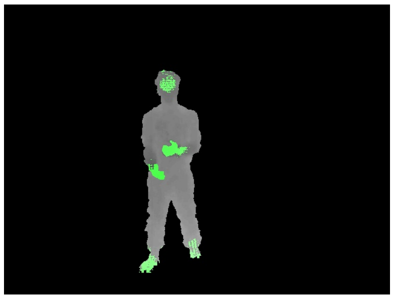
6.3.2: EPBP

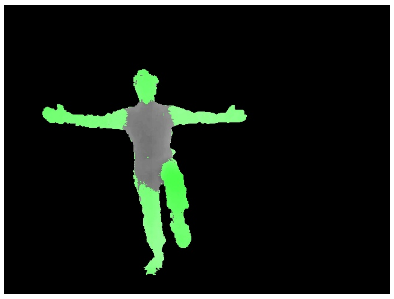
Figure 6.3: Body Part Segmentation for Initial Pose

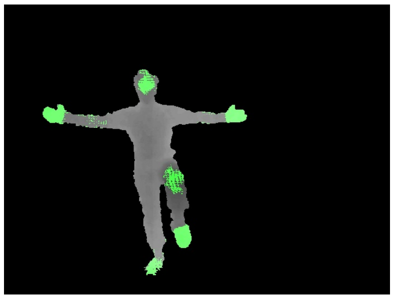

6.4.1: Example1: Limbs

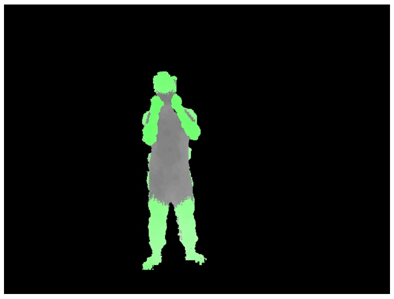

6.4.2: Example1: EPBP


6.4.3: Example2: Limbs


6.4.4: Example2: EPBP


6.4.5: Example3: Limbs


6.4.6: Example3: EPBP


6.4.7: Example4: Limbs

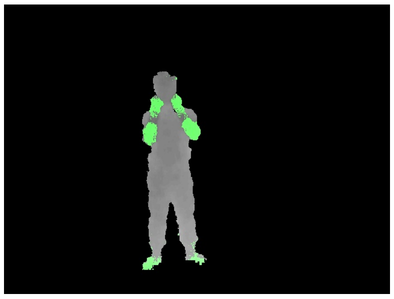

6.4.8: Example4: EPBP

Figure 6.4: Body Part Segmentation Results for Various Poses

Figure 6.4.7). However, in Figure 6.4.1, the left arm is only partially recognized, since its too close to the torso and partially occluded by the torso. The segmentation of the endpoint body parts is relatively trickier. In Figure 6.3.2, some pixels on the left arm are recognized as EPBP pixels. Pixels at the left knee in Figure 6.4.6 and pixels at the elbows in Figure 6.4.8 are segmented as EPBP pixels, because in these special poses, their local shapes are similar to that of EPBP pixels. In Figure 6.4.8, the head is not segmented since it is partially occluded by the two hands.

Two things should be noted during actual implementation. First, the feature calculation for each pixel is independent. Therefore a parallel implemented can be designed and realized. Second, it may not be necessary to compute the features for every pixel. In our implementation, to speed up the computation, the original depth map is down sampled at every 5 pixels.

6.1.3 More Efficient Clustering

In the original algorithm, Tier 2 IR Points are generated by clustering the Tier 1 IR Points using an agglomerative algorithm. A clustering tree is fully constructed with two clusters merged at each level. Given a precision value, the tree is traversed using a breath-first search and the clusters are determined. This process can be time consuming when the number of Tier 1 Points is large. Instead of constructing the entire tree and then searching from top down, the cluster merging stops when the given precision is reached from the bottom up.

6.1.4 Tier 2 IR Points Identification

In the original algorithm, the identification of the Tier 2 IR Points is based on trained models using the LSC descriptor. This works well when the contours of the body parts are clear from a depth map, which does not hold for a Kinect depth map. The stereo and interpolation based depth recovery causes the boundary of the objects to be blurry. Instead of using learned LSC feature models to identify Tier 2 IR Points, we propose to use the following two cues:

1. Temporal: Utilize the tracking results from the previous frame and compute distances between the detected Tier 2 IR Points and the tracked endpoint body part locations.

2. Heuristic: When 5 Tier 2 IR Points are detected, but less than 5 are identified based on temporal continuity, a heuristic identification is applied. If the spatial distribution of the 5 detected Tier 2 IR Points is close to that of the initial pose, identities are assigned. Otherwise, temporal based identification is still used. This enables the capability of tracking recovery.

6.2 Simplified Skeleton Model

The skeleton model described in Chapter 4 is constructed according to the Acclaim Skeleton File (ASF) convention to be compliant with the CMU MOCAP dataset. The drawback of this model is that there is a set of offset angles for each bone, resulting in two additional matrix multiplications for each bone during the forward kinematics computation. To make the calculation more efficient, a new skeleton model is designed, consisting of 18 bones and a maximum of 28 DOF. The tree structure is rooted at the centre of the two hip joints. The default pose is shown in Figure 6.5. Each bone has a Local Frame with its origin point at the starting joint, with an initial X axis to the right, Y axis pointing downward, and Z axis pointing into the plane (right hand rule system). Each bone is aligned with one of the axes of its Local Frame to avoid the offset angles. The bone lengths can be set using the real values of the person to be tracked. Angle limits are set based on ergonomics considerations. No outer shape model is used.

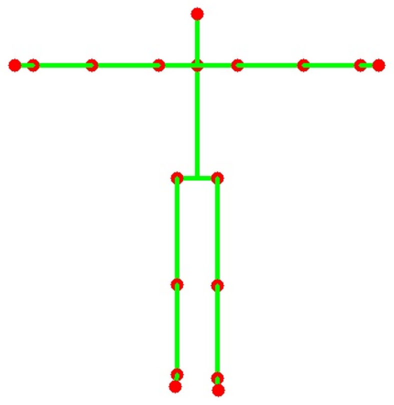


Figure 6.5: Default Pose for the Simplified Skeleton Model

Table 6.1: Partitions for Particle Filter Tracking

Torso	9 DOF
Right Upper Arm	3 DOF
Left Upper Arm	3 DOF
Right Forearm	1 DOF
Left Forearm	1 DOF
Right Thigh	3 DOF
Left Thigh	3 DOF
Right Calf Leg	1 DOF
Left Calf Leg	1 DOF

6.3 Particle Filter Implementation

Successful tracking is achieved by using a new implementation of the particle filter with partitioned sampling. The skeleton-only model is used for tracking. Particle likelihood is evaluated using the identified endpoint body part locations, the depth map, and other detected body part information.

The full tracking space (25 DOF) is partitioned into 9 partitions, as listed in Table 6.1. The 9 DOF of the Torso Partition includes 6 DOF for the root rotation and translation, 1 DOF for back rotation about Z axis, and 2 DOF for head rotation about X and Z axes.

The particles are first propagated, evaluated, and resampled in each of these partitions, and then each particle is evaluated one more time by considering all the partitions to estimate the final distribution.

The particles are weighted using the following cues:

Explicit Distance Cue: Tier 2 IR Points are detected in each frame and endpoint body parts are identified either using the previous frame tracking result or the heuristic cue. The distance is computed as the sum of the Euclidean distances between the predicted and detected endpoint body part locations. For the Torso Partition, only the head distance is computed. For the Right/Left Forearm Partition, only the right/left hand distance is computed. For the Right/Left Calf Leg Partition, only the right/left foot distance is computed. For the weight evaluation for all the other partitions, endpoint body parts are not considered. All the available endpoint body parts are used in the final full body evaluation.

Depth Cue: Each bone in a partition is sampled using a fixed step length. The distance is calculated as the average of the depth differences for all these sampled points.

Note that the depth sensed by the depth camera is the surface depth, so a certain amount of depth suppression needs to be considered. The depth suppression values used in our experiments are generated from ground truth.

Implicit Distance Cue: The $m1$ feature map is used to provide more body part information. The distance function is formed so that the torso prefers the darker region, while limbs prefer the lighter. The distances are computed similar to the depth cue, based on sampled points on the bones. This cue can complement the Explicit Distance Cue when not all the endpoint body parts are detected.

6.4 Test Dataset

To test the full body human pose tracking system, a new dataset is designed and collected using a synchronized Kinect sensor and a commercial marker based MOCAP system.

6.4.1 Hardware

Kinect Sensor

An brief introduction to the sensor can be found in Section 2.5. The open source platform OpenNI is adopted for driving the Kinect hardware and collecting the sensor data, including the depth maps and the colour images. The depth camera and the colour camera are assumed to be synchronized. The NITE middleware is used to generate human silhouettes from depth maps at run time. The skeleton tracking results from NITE are also recorded for future reference.

MOCAP System

The MOCAP system used for collecting ground truth data is an 8-camera Motion Analysis system located in the Adaptive System Laboratory at the University of Waterloo. The MOCAP system is based on reflective markers, which reflect the infrared light emitted from the cameras. Prior to dataset collection, tests were conducted to confirm that this system does not interfere with the Kinect sensor when operating simultaneously.

A calibration step needs to be performed before each capture, consisting of two steps:

1. Set the world coordinate frame using an L frame, and obtain the locations and orientations of some of the cameras with respect to the L frame.
2. Wave a calibration wand to cover the capture volume, estimate and refine the poses of all the cameras with respect to the world frame.

The world frame set by the L frame is also defined as the World Frame for the Kinect system.

System Integration

To collect synchronized data, the Kinect sensor and the MOCAP system need to be integrated, in terms of spatial calibration and temporal synchronization.

Spatial Calibration: Depth data captured by the Kinect sensor is described in the depth camera frame, which has its origin at the centre of the infra-red camera lens, with X pointing to the right, Y pointing down and Z pointing out (right hand rule system) when the Kinect is upright. The depth value for each pixel is the Z axis projection of the distance to the closest surface along the projection line. Using the intrinsic parameters of the depth camera, the 3D point cloud can be recovered in the depth camera frame. However, the estimation of the intrinsic parameters of a depth camera is not straightforward. OpenNI provides a function that transforms the original depth map into a new depth map as if taken from the colour camera with exactly the same intrinsic and extrinsic parameters. This function is used and only the parameters of the colour camera need to be estimated. The colour camera frame, which has its origin at the centre of the colour camera lens, with X pointing to the right, Y pointing down and Z pointing out (right hand rule system) when the sensor is upright, is defined as the Camera Frame.

The marker locations captured by the MOCAP system are described in the World Frame. A transformation between the World Frame and the Camera Frame needs to be determined. A calibration board is designed based on a known chess board pattern, with the Board Frame located at the top-left corner when facing the camera. 32 reflective markers are attached at the outside-most corners, and an additional one at a place away from the board centre to make the marker distribution asymmetric. The intrinsic parameters of the colour camera are first calibrated using the functions in the OpenCV library. The intrinsic parameters include the focal lengths and scaling in the X and Y directions, the image centre location, and the distortion coefficients. Then the board is placed at the centre of the MOCAP volume so that it can be seen by the Kinect camera and captured by the MOCAP system. The transformation between the Board Frame and the Camera Frame is estimated through extrinsic parameter calibration using the functions from the OpenCV library. The coordinates of the markers can then be transformed into the Camera Frame. Keeping the board unmoved, the marker locations in the World Frame can be determined using the MOCAP system. Then, the transformation between the World Frame and the Camera Frame can be estimated in a least square sense using the two sets of coordinates of the same markers.

To achieve an accurate estimation, a sufficient number of markers needs to be used (34 in our case). Moreover, since all the markers are coplanar, i.e. in the calibration board plane, the estimated matrix may degenerate and be invalid as a transformation matrix. To rectify this problem, two sets of virtual markers are added. The coordinates of the virtual markers are calculated by adding a positive or a negative displacement to each real marker in the direction perpendicular to the board plane.

Temporal Synchronization: The frame rates of the Kinect sensor and the MOCAP system are different. The Kinect sensor works at 30Hz maximum, while the MOCAP system is usually operated at 200Hz. In addition, the two systems run in different threads with different starting times. A temporal synchronization needs to be performed to enable a direct comparison of the data.

For the Kinect sensor, the data is collected at its highest frame rate with the best resolution. The best resolution for the cameras is 640 by 480. Since the depth map has a float data type which is 4 bytes per pixel, one depth map image has a size of about 1.17M. For a standard PC, the hard disc usually is not fast enough for saving a data block of this size at 30Hz. To deal with this problem, the depth map for each frame is first buffered into memory, and then saved to hard disc in another thread. This allows recording for up to one and half minutes for a single trial before running out of memory. The MOCAP system runs in a third thread once it is triggered.

The time for each frame of the MOCAP data is recorded automatically in the MOCAP file, in terms of the MOCAP Clock. The MOCAP Clock does not start immediately after it is triggered from the main thread. To synchronize the two systems, the starting time of the MOCAP Clock in the System Clock needs to be obtained, and time stamps for Kinect data need to be recorded. After starting the System Clock, the time stamp corresponding to each Kinect frame is saved. At the same time, the main thread tries to retrieve frames from the MOCAP thread until it gets the first one. Usually, the first frame obtained by the main thread is not the actual first frame. The time of this frame in the System Clock is recorded, and the time in the MOCAP Clock can be looked up using its frame number. In this way, the two clocks can be aligned.

6.4.2 Dataset Design

To create a comprehensive dataset for evaluating human pose tracking algorithms, colour images, depth maps, human silhouettes, and MOCAP ground truth are collected.

For ground truth collection, 28 markers are placed at 14 key joints, 2 at each joint. The marker locations are: head front, head back, left/right shoulder front, left/right shoulder back, left/right elbow medial, left/right elbow lateral, left/right wrist medial, left/right wrist lateral, left/right hip front, left/right hip back, left/right knee medial, left/right knee lateral, left/right ankle medial, and left/right ankle lateral.

The subjects should include both males and females and cover various body shapes. 22 motions are designed to actuate all the DOFs of all the major joints, starting with an initial pose close to the initial pose of the skeleton model (Figure 6.5). The 22 motions are listed in Table 6.2.

6.4.3 Data Collection and Post-processing

We collected the 22 designed motions from 5 human subjects, including 3 males (S1, S2 and S4) and 2 females (S3 and S5). Each motion for each actor is collected in three trials. Calibration is performed before the first capture and the positions of the MOCAP cameras and the Kinect sensor are kept unmoved for the rest of the capture sessions. The transformation between the Camera Frame and the World Frame is estimated and validated by inspecting the projection result of the MOCAP markers onto the colour images.

From the three trials for each motion, the best one is selected based on MOCAP quality. Then post-processing is performed on the MOCAP data by removing phantom markers and interpolating for the missing ones. Although the System Clock and the MOCAP Clock are aligned using the method described previously, it is still possible that a small time discrepancy to remain for some trials. This discrepancy is then manually adjusted. Since the MOCAP system runs at a much higher frame rate than the Kinect sensor, only the frames corresponding to Kinect frames are extracted to form a more compact MOCAP file.

Bone lengths and the joint suppression depth values are calculated using the MOCAP data for each human subject. The final result is obtained by averaging over all the frames for each subject. These bone lengths are used for constructing the skeleton models.

Table 6.2: 22 Motions

M1	Raise arms sideways without touching hands above head
M2	Raise arms sideways touching hands above head
M3	Raise arms to the front
M4	Punch using two hands, one after the other
M5	Cross two arms in front of chest
M6	Bend waist side to side, keeping arms in initial pose
M7	Put hands on hips, and bend waist side to side
M8	Bend waist forward, keeping arms in initial pose
M9	Put hands on hips, and bend waist forward
M10	Lift legs one after the other
M11	Kick with legs straight, one after the other
M12	Kick with legs bent first and then kick out, one after the other
M13	Squat, keeping arms in initial pose
M14	Jump and raise arms simultaneously
M15	Walk in place
M16	Run in place
M17	Little teapot motion (similar to motion used in [88])
M18	Boxing motions with punching and dodging
M19	Point sideways (up, middle and down), one hand after the other
M20	Sit down on a chair and stand up, keeping arms in initial pose
M21	Put hands down, sit down on a chair and stand up
M22	Reach out to front (up, middle and down), one hand after the other

6.5 Experimental Results and Discussion

The system is tested using all the videos included in the new dataset and demonstrates high accuracy and efficiency. The average ***Overall Average Error (OAE)*** (defined in Section 4.2.4) for all the 110 videos is 10.7cm, with a standard deviation of 3.1cm. The experiments run on a laptop with a Quad Core i7-2620 CPU, and achieve a processing speed of 3Hz. Since the implementation has only one thread, just a quarter of the CUP is used. The ***Overall Average Error***s for all the 110 test videos can be found in Table 6.3.

A bar plot for the table content is shown in Figure 6.6, grouped by motion type. It can be observed that the system performs quite inconsistently for different actors on some motions, such as M5, M14 and M18. The main reason for this is the inconsistency in the accuracy of the endpoint body part detection. In the current implementation, only the $m1$ feature is calculated with a fixed search range and thresholding parameters, so the algorithm can not generalize well to different actors with significantly different body shapes. Using more features and learned parameters will improve this problem.

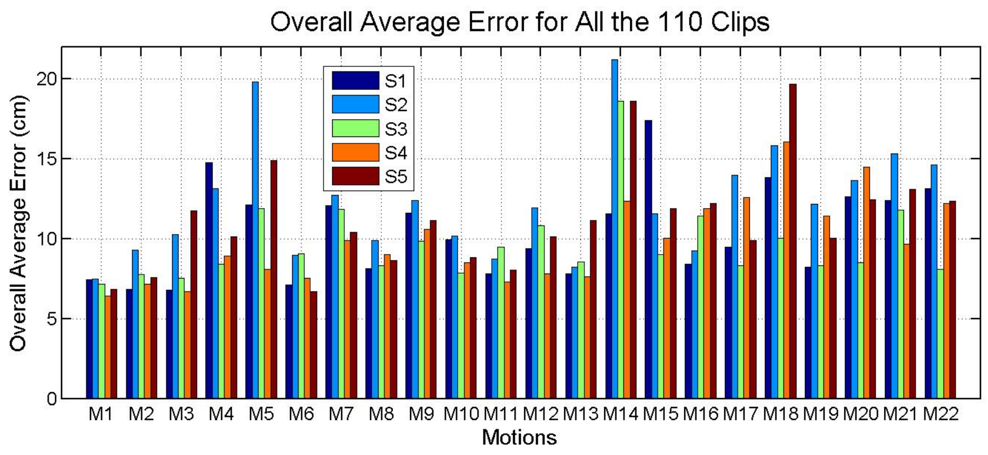


Figure 6.6: Overall Average Errors for All the Clips

In Figure 6.7, average OAE for each motion is computed by averaging over different actors. This figure demonstrates the tracking accuracy on different motions. Consistent with our intuition, simple motions (e.g. M1, M2, M6 and M11) have smaller errors than more complex motions (e.g. M4, M18 and M22). Also, motions with in-plane movements only usually have smaller errors than the motions containing out-of-plane movements, such as M1, M2 compared to M3, M4 and M5. Moreover, the motions with hands reaching out tend to have smaller errors than the motions with hands touching each other or other body

Table 6.3: Overall Average Errors for All Testing Videos

Err(cm)	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	M22	Mean	Std
S1	7.4	6.8	6.8	14.8	12.1	7.1	12.1	8.1	11.6	9.9	7.8	9.4	7.8	11.6	17.4	8.4	9.4	13.8	8.2	12.6	12.4	13.1	10.4	2.9
S2	7.5	9.3	10.2	13.1	19.8	9.0	12.7	9.9	12.4	10.2	8.7	11.9	8.2	21.2	11.6	9.2	14.0	15.8	12.1	13.7	15.3	14.6	12.3	3.5
S3	7.2	7.8	7.5	8.4	11.9	9.1	11.8	8.3	9.9	7.9	9.4	10.8	8.5	18.6	9.0	11.4	8.3	10.0	8.3	8.5	11.8	8.1	9.7	2.5
S4	6.4	7.2	6.7	8.9	8.1	7.5	9.9	9.0	10.6	8.5	7.3	7.8	7.6	12.4	10.0	11.9	12.6	16.1	11.4	14.5	9.7	12.2	9.8	2.6
S5	6.8	7.6	11.8	10.1	14.9	6.7	10.4	8.6	11.1	8.8	8.0	10.1	11.1	18.6	11.9	12.2	9.9	19.7	10.0	12.4	13.1	12.3	11.2	3.3
Mean	7.1	7.7	8.6	11.1	13.3	7.9	11.4	8.8	11.1	9.1	8.3	10.0	8.7	16.5	12.0	10.6	10.8	15.1	10.0	12.3	12.4	12.1	10.7	
Std	0.5	0.9	2.3	2.8	4.3	1.1	1.2	0.7	1.0	1.0	0.8	1.5	1.4	4.2	3.2	1.7	2.3	3.5	1.8	2.3	2.0	2.4		3.1

parts, such as M1 compared to M2, M6 compared to M7, and M8 compared to M9.

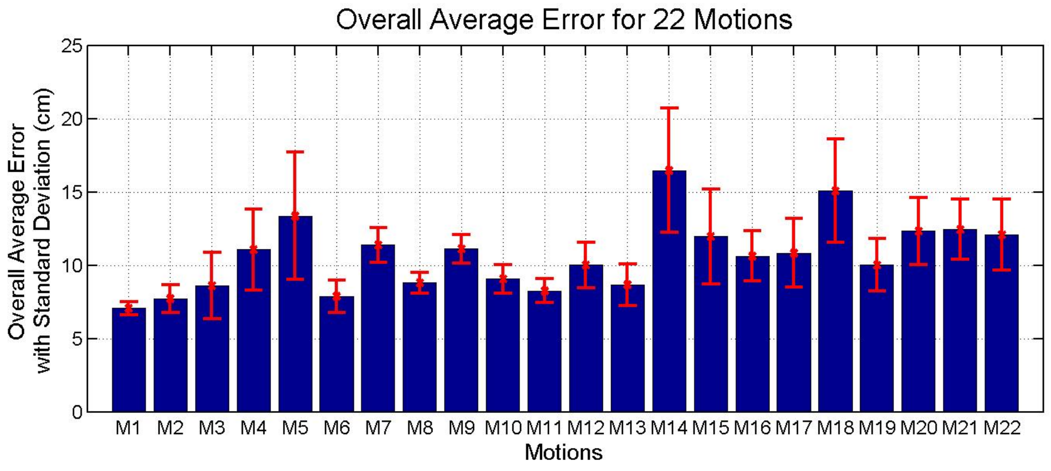


Figure 6.7: Overall Average Errors for Each Motion

In the following two sections, some typical tracking clips will be selected to demonstrate the tracking results in more details.

6.5.1 Selected Successful Clips

Clip 1: Motion M2 by Actor S1

In this clip, actor S1 performs the M2 motion, which consists of raising the arms sideways and touching the hands at the highest point. 12 key frames are sampled from the video to demonstrate the tracking results, shown in Figure 6.10. The green sticks represent the tracked skeleton, the small red dots represent the tracked joint locations, the small light blue dots represent the ground truth joint locations, and the big dots indicate the detected endpoint body part locations (blue for head, green for right hand, dark green for left hand, red for right foot and dark red for left foot). The same drawing rules apply to all the clips.

From the sampled frames, we can see that the endpoint body part detection and the skeleton tracking is accurate most of the time. Challenges occur when the two hands touch each other when they reach the top most position above the head. In Figure 6.10.8, although the right hand is still correctly located, the left hand is mistakenly located at the right elbow. However, this does not confuse the tracker due to the integration of the cues including the ***Depth Cue*** and the ***Implicit Distance Cue***. As soon as the two hands separate as shown in Figure 6.10.9, the two hands are immediately correctly identified.

Numerical errors are also plotted, in terms of the ***Joint Average Error (JAE)*** shown in Figure 6.8 and the ***Average Error (AE)*** over frames shown in Figure 6.9 (both defined in Section 4.2.4). In Figure 6.8, JAE is plotted for the 14 key joints with their standard deviations. The meaning of the abbreviations on the X axis are listed in Table 6.4. From this figure, we can see that the errors of the head and the left foot are the biggest. The head is the largest endpoint body part among the five, and the detected location is usually around the centre of the head with some uncertainty. Another reason is that in order to stabilize the tracking of the torso, the stiffness of the neck joint is set high. So the accuracy of the head is compromised to ensure a more stable torso, which is important since it affects the tracking of the limbs connecting to it. The feet are also hard to detect accurately. This is because for most of the time they are on the ground, and the segmentation is not always satisfying. Poor segmentation means deformed shapes, which directly affects shape based detection. The two shoulders also have relatively bigger errors, this is because the shoulder is the widest part on a human body, and since no outer shape model is used, the back bone movement is not well constrained. Since the hands are detected well throughout the video, the arms are tracked accurately.

In Figure 6.9, we can observe that when the actor starts to move, there is a sharp increase in the error. The major peak occurs at around Frame 250, when the two hands of the actor touch together. As the two hands separate and the actor puts down the arms, the error quickly decreases.

Table 6.4: Joint Name Abbreviations

H	Head	N	Neck
RS	Right Shoulder	LS	Left Shoulder
RE	Right Elbow	LE	Left Elbow
RW	Right Wrist	LW	Left Wrist
RH	Right Hip	LH	Left Hip
RK	Right Knee	LK	Left Knee
RA	Right Ankle	LA	Left Ankle

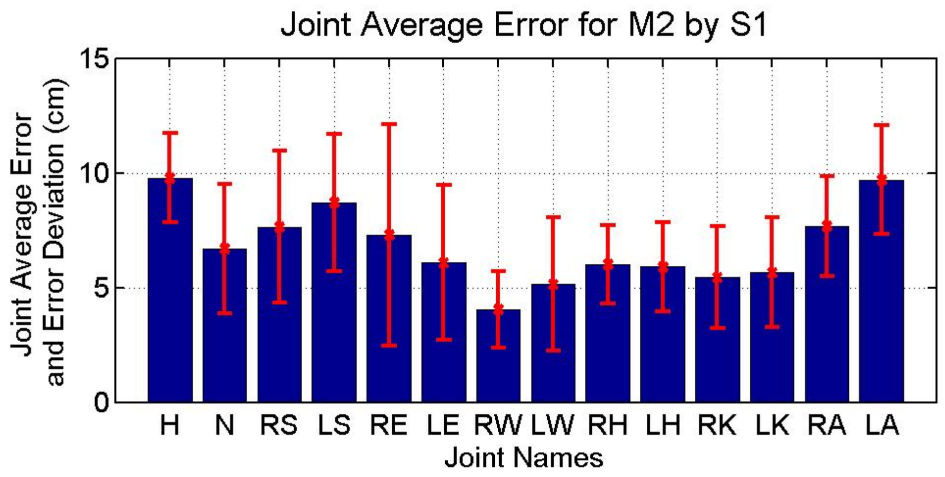


Figure 6.8: JAE with Deviation for M2 by S1

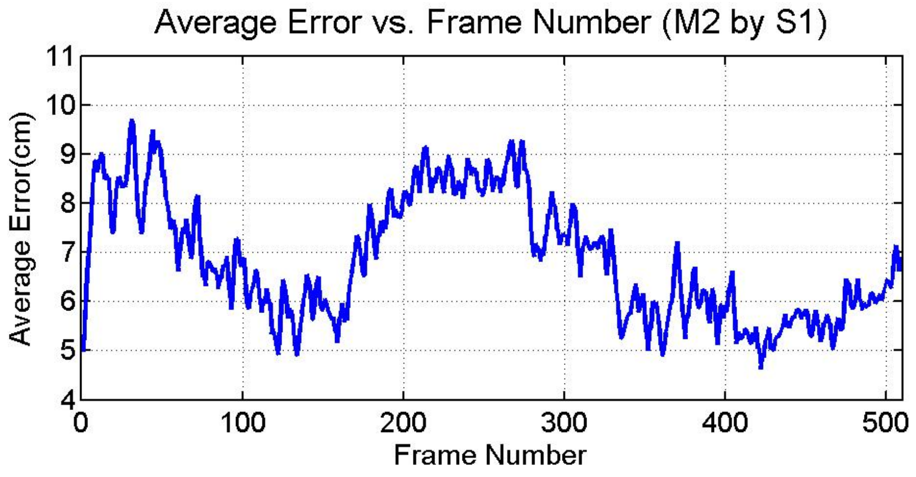
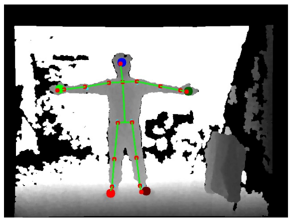
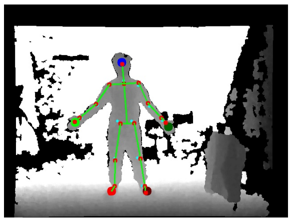
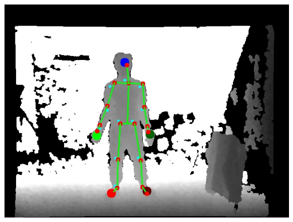
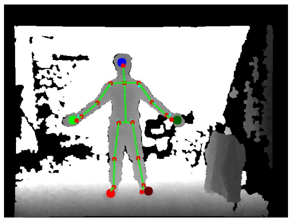


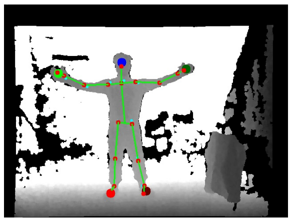
Figure 6.9: AE over Frames for M2 by S1

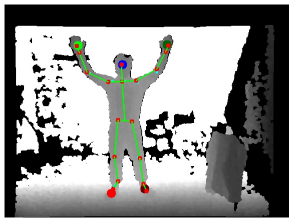

6.10.1: Frame0

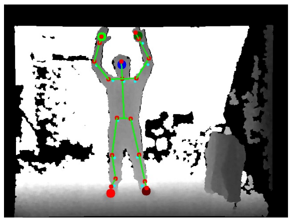

6.10.2: Frame50

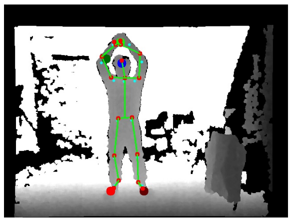

6.10.3: Frame82

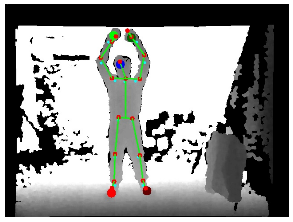

6.10.4: Frame111

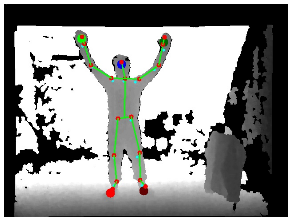

6.10.5: Frame139

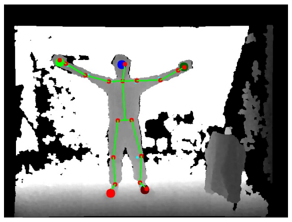

6.10.6: Frame169


6.10.7: Frame191


6.10.8: Frame218


6.10.9: Frame264


6.10.10: Frame299


6.10.11: Frame333

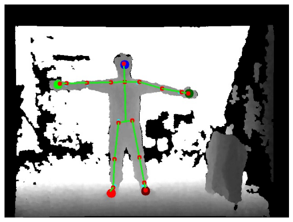

6.10.12: Frame382

Figure 6.10: Extracted Frames for M2 by S1

Clip 2: Motion M12 by Actor S4

In this clip, actor S4 performs the kicking motion with each leg bent first and then kicked out, one after the other. 12 key frames are sampled from the tracking video, with results and ground truth drawn on them, as shown in Figure 6.13. JAE is plotted in Figure 6.11 and AE over frames is plotted in Figure 6.12.

From Figure 6.13, we can see that the endpoint body parts are accurately detected throughout the clip, even when the feet are lifted high off the ground and their shapes change significantly (Figure 6.13.4 and Figure 6.13.9). The skeleton model successfully follows the lower body motions. However, if we look closely, we can find that the knees are sometimes off their real location comparing to the ground truth (Figure 6.13.3, Figure 6.13.5, Figure 6.13.7 and Figure 6.13.8). This observation is confirmed by the JAE plot in Figure 6.11, where the knees have the biggest errors, as well as standard deviations. This error is mainly caused by the out-of-plane motion of the legs, in which part of the thighs are occluded. Since the ***Depth Cue*** is only applied to the sampled points on the bones, it is not strong enough to deal with such out-of-plane motions. Besides, the shoulders also have big errors, with similar reasons as explained in the previous motion.

The AE as time proceeds is plotted in Figure 6.12. The centres of the two major peaks are located at around Frame 75 and Frame 225, which correspond to the moments when each of the two legs is lifted to the highest point before the actor kicks out.

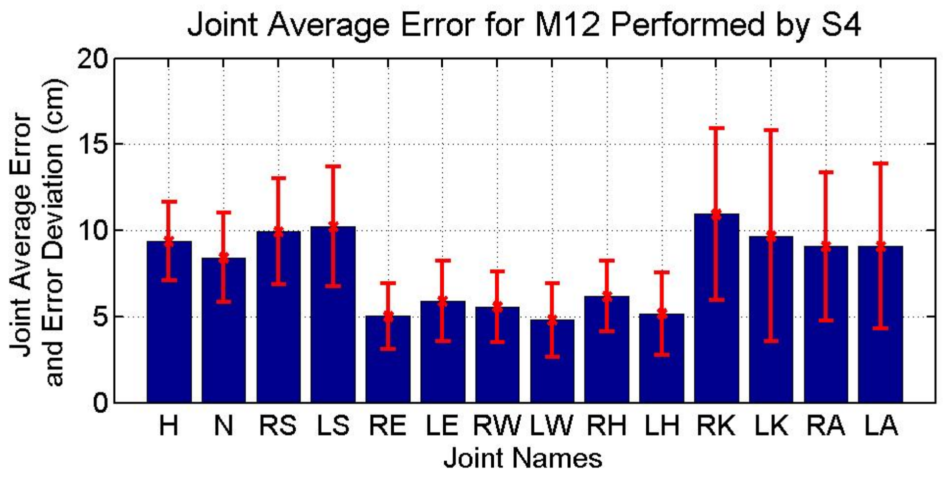


Figure 6.11: JAE with Deviation for M12 by S4

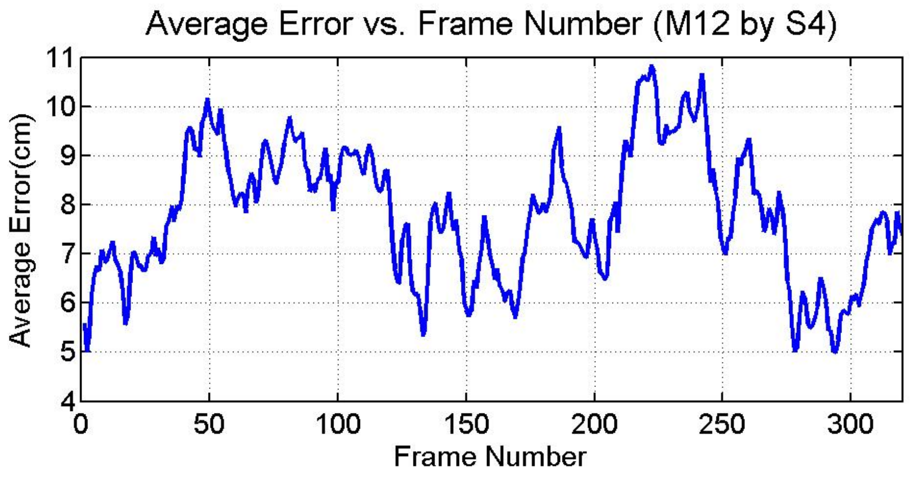
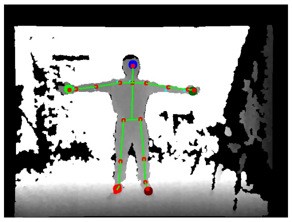
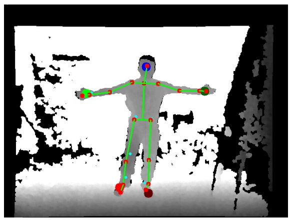
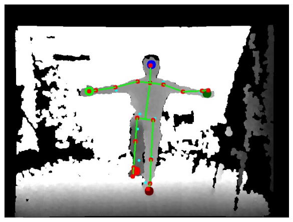
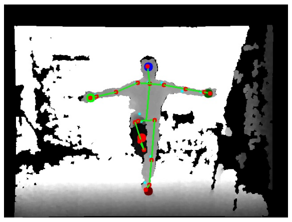


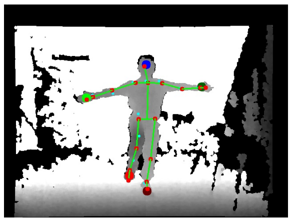
Figure 6.12: AE over Frames for M12 by S4

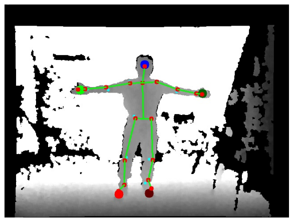

6.13.1: Frame0

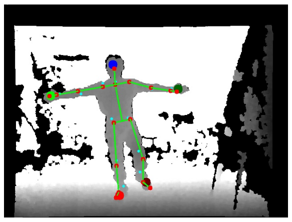

6.13.2: Frame41

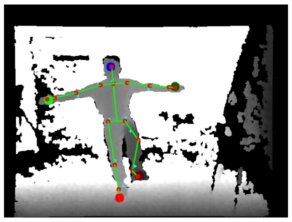

6.13.3: Frame62

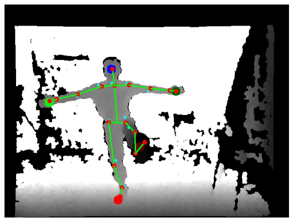

6.13.4: Frame83

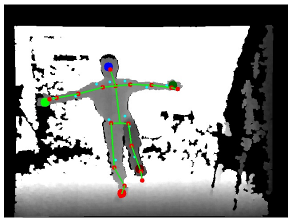

6.13.5: Frame114

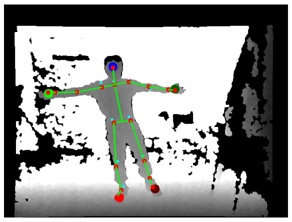

6.13.6: Frame134


6.13.7: Frame175


6.13.8: Frame195


6.13.9: Frame207


6.13.10: Frame226


6.13.11: Frame250

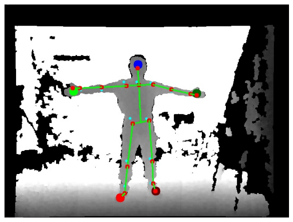

6.13.12: Frame306

Figure 6.13: Extracted Frames for M12 by S4

Clip 3: Motion M15 by Actor S3

In this clip, actor S3 performs the walking in place motion with 8 continuous strides. 15 frames are sampled from the tracking video, as shown in Figure 6.16. JAE with standard deviation is plotted in Figure 6.14 and AE over time is plotted in Figure 6.15.

For this walking motion, there are two major challenges: occlusion and fast movement. While swinging the arms, the rear arm may be occluded by the torso (Figure 6.16.3), and the front arm may fold (Figure 6.16.4). These occlusions make the detection of the hands more difficult, and also cause problems for skeleton tracking due to the unseen parts. But Figure 6.16.3 demonstrates the benefit of including the ***Implicit Distance Cue***. Although the left hand is not detected, the left arm is still correctly located. When the arm moves forward, the forearm may occlude the upper arm as shown in Figure 6.16.6. This confuses the tracker and causes the tracking result of the right arm to become inaccurate. Fast movement can also make the system suffer, due to the simple dynamical model. Since only Gaussian noise with constant variance is applied for making configuration prediction, the model is unable to follow when the motion is fast. This is demonstrated in Figure 6.16.10, where the actor lifts the right leg rapidly and the model falls behind. The out-of-plane motion of the arms and the legs is another source of error.

In Figure 6.14, the head still has a big error. The reason is similar to the previous motions. The left arm has significantly larger error and variance compared to the right arm, due to the actor's habit of moving the left arm closer to the torso.Although the feet are well detected, the legs still have big errors. This is mainly due to the fast out-of-plane movements of the legs. There are several peaks in the AE plot in Figure 6.15, corresponding to the eight strides performed by the actor.

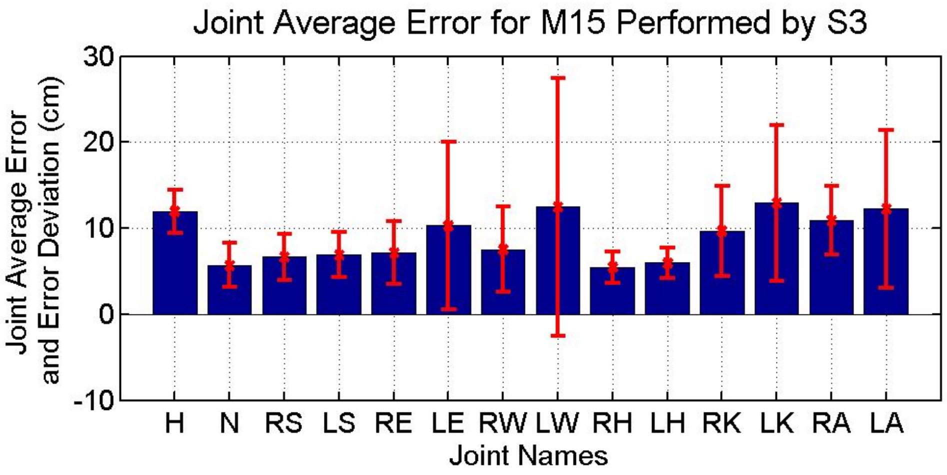


Figure 6.14: JAE with Deviation for M15 by S3

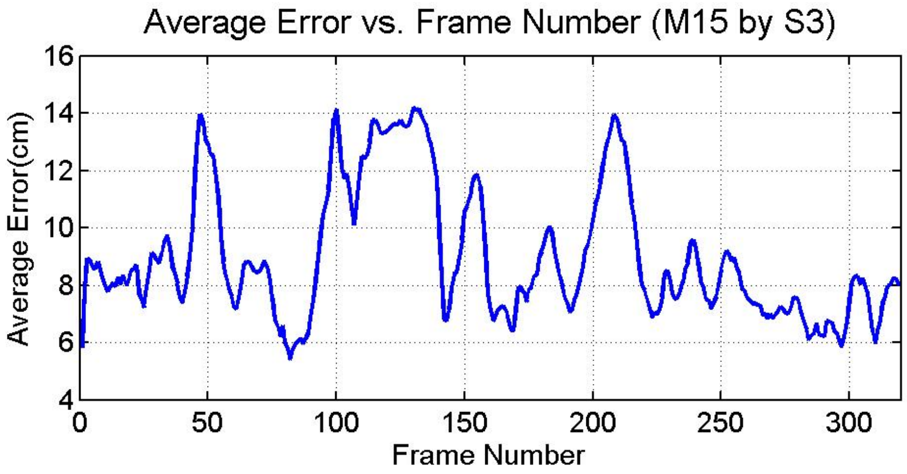
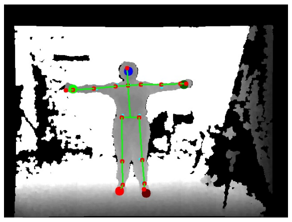
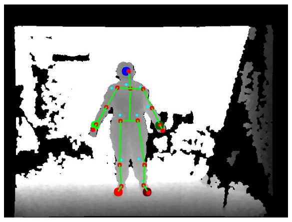
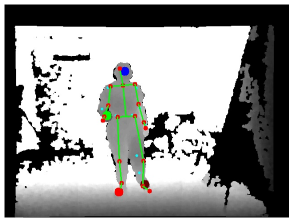
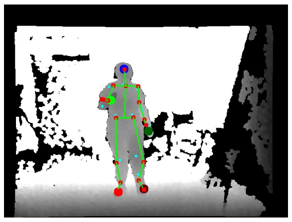


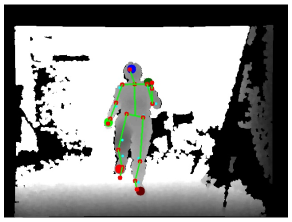
Figure 6.15: AE over Frames for M15 by S3

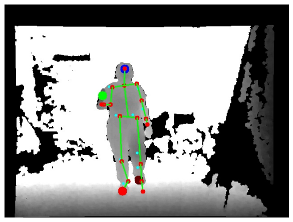

6.16.1: Frame0

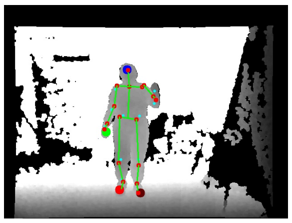

6.16.2: Frame33

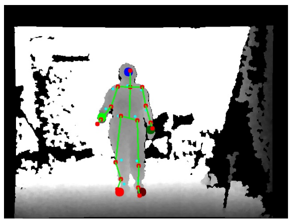

6.16.3: Frame41

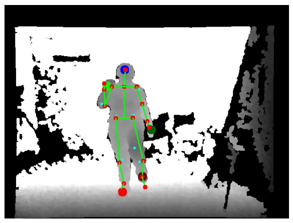

6.16.4: Frame54

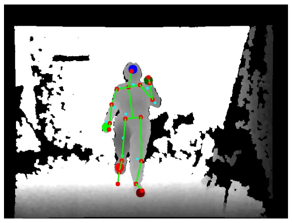

6.16.5: Frame74

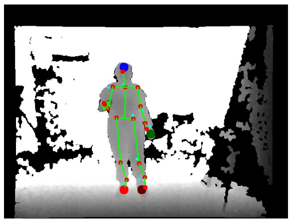

6.16.6: Frame94

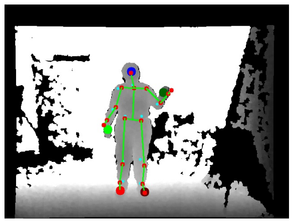

6.16.7: Frame118

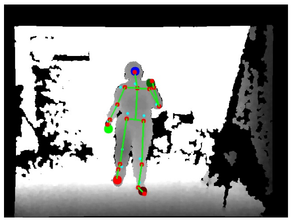

6.16.8: Frame144

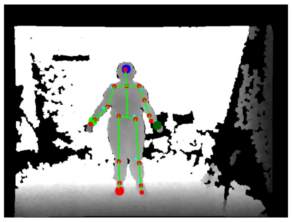

6.16.9: Frame158


6.16.10: Frame182


6.16.11: Frame218


6.16.12: Frame227


6.16.13: Frame243


6.16.14: Frame258

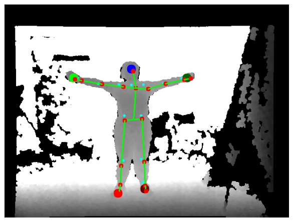

6.16.15: Frame318

Figure 6.16: Extracted Frames for M15 by S3

Clip 4: Motion M17 by Actor S3

In this clip, actor S3 performs the Little Teapot motion, similar to the motion used for testing the monocular upper body tracking system in Section 4.2. 15 frames are sampled from the tracking video, as shown in Figure 6.19. JAE with standard deviation is plotted in Figure 6.17, and AE over frames is plotted in Figure 6.18.

The major challenge of this motion is that it activates all the upper body DOFs, including the complicated movement at the right shoulder, and the difficulty in detecting the hands when they touch the torso. In Figure 6.19, we can see the right arm is accurately tracked although it performs a complicated motion. Compared to this, the left forearm is not well tracked when the left hand is touching the torso, since the left hand is not detected. But the left upper arm still remains roughly in its place. This is confirmed by Figure 6.17, which indicates the errors for the right shoulder and the right arm are smaller than that of the left shoulder and the left arm. This comparison demonstrates the importance of the endpoint body part detection in skeleton tracking. Although the two hands are correctly detected when they first touch the torso shown in Figure 6.19.3, the fast movement of the two arms from Frame 56 to Frame 71 makes the model fail to follow. This indicates that the dynamical model is not fast enough, and that there is an insufficient number of particles at the shoulders. As the hands touch closer onto the torso, the hands on the torso are not detected from Frame 87 to Frame 259. Although the torso bends to one side, the two hips are always located accurately, shown in Figure 6.17.

In Figure 6.18, the first error peak happens at around Frame 80, when the hands start to touch the torso. From Frame 170 to Frame 250, the actor bends to the right side. From Frame 300 to 350, the two arms recover to their initial poses.

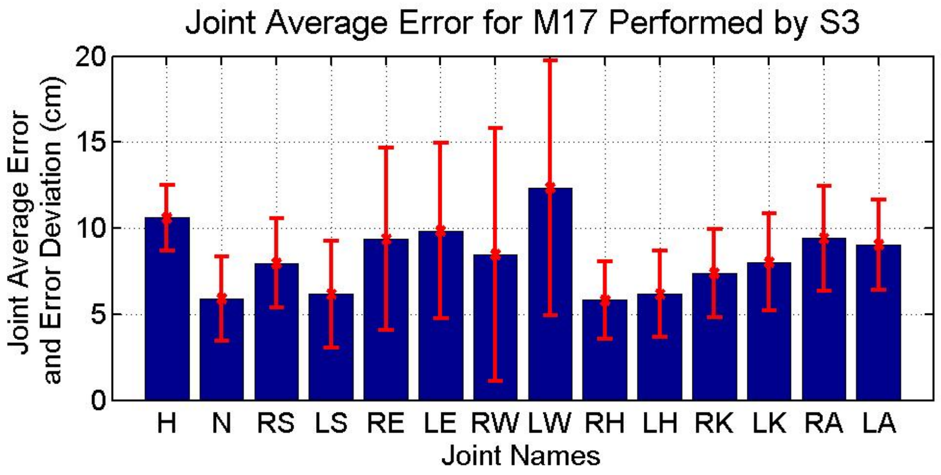


Figure 6.17: JAE with Deviation for M17 by S3

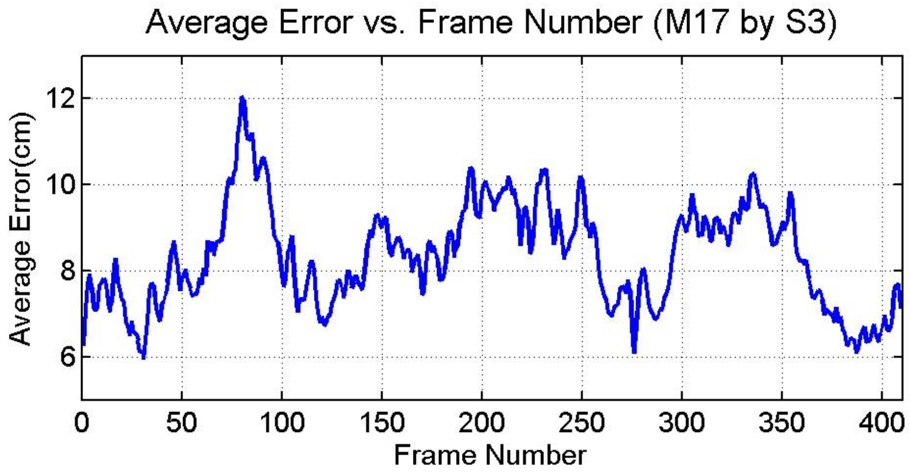
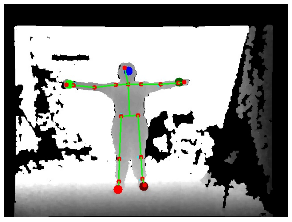
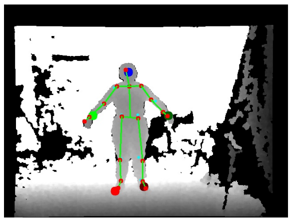
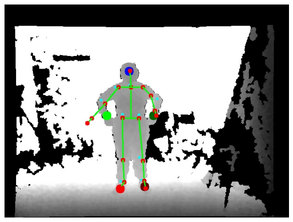
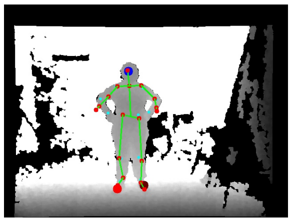


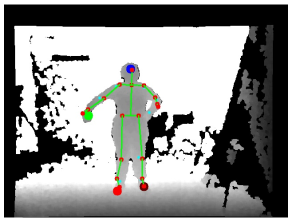
Figure 6.18: AE over Frames for M17 by S3

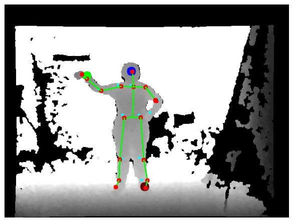

6.19.1: Frame1

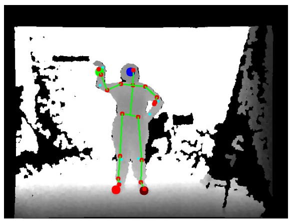

6.19.2: Frame56

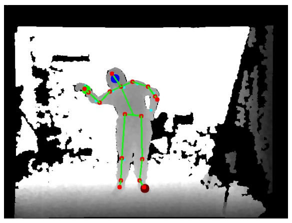

6.19.3: Frame71

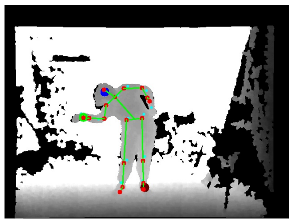

6.19.4: Frame87

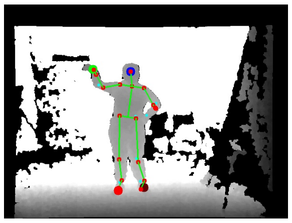

6.19.5: Frame107

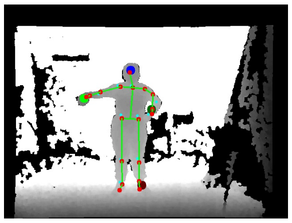

6.19.6: Frame128

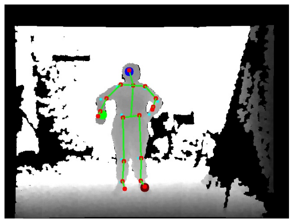

6.19.7: Frame150

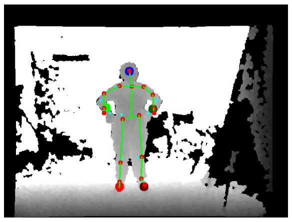

6.19.8: Frame179

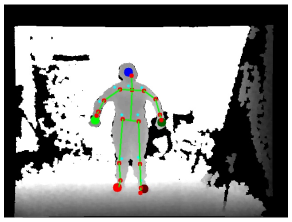

6.19.9: Frame206


6.19.10: Frame259


6.19.11: Frame278


6.19.12: Frame288


6.19.13: Frame297


6.19.14: Frame326

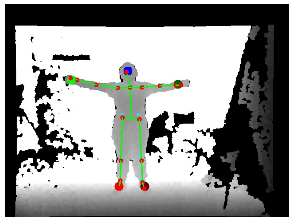

6.19.15: Frame362

Figure 6.19: Extracted Frames for M17 by S3

Clip 5: Motion M19 by Actor S1

In this clip, actor S1 performs the pointing sideways motion (up, middle and down), one hand after the other. 15 frames are sampled from the tracking video, as shown in Figure 6.22. JAE with standard deviation is plotted in Figure 6.20, and AE over frames is plotted in Figure 6.21.

The sampled frames in Figure 6.22 show that the tracking is very accurate overall. The skeleton is tracked correctly even when the endpoint body parts are not detected, which clearly demonstrates the benefit of using the ***Implicit Distance Cue***. From Frame 61 to Frame 247, the left hand is not detected because it is too close to the torso. In this case, the ***Depth Cue*** will not help much in locating the left arm in the plane, since the depth of the left arm and the torso is similar. However, in a $m1$ map, the arm is still a convex shape with higher $m1$ values than the edge between the arm and the torso, and the torso itself. This ensures that the left arm does not move across the edge. The same thing happens to the right arm from Frame 274 to Frame 391.

In Figure 6.20, we can see the JAEs for all the joints are below 10cm, which demonstrates good tracking accuracy. The JAE for different joints are similar, but the two wrist joints have significantly bigger variance compared to the other joints. This is reasonable since they move the most in this clip.

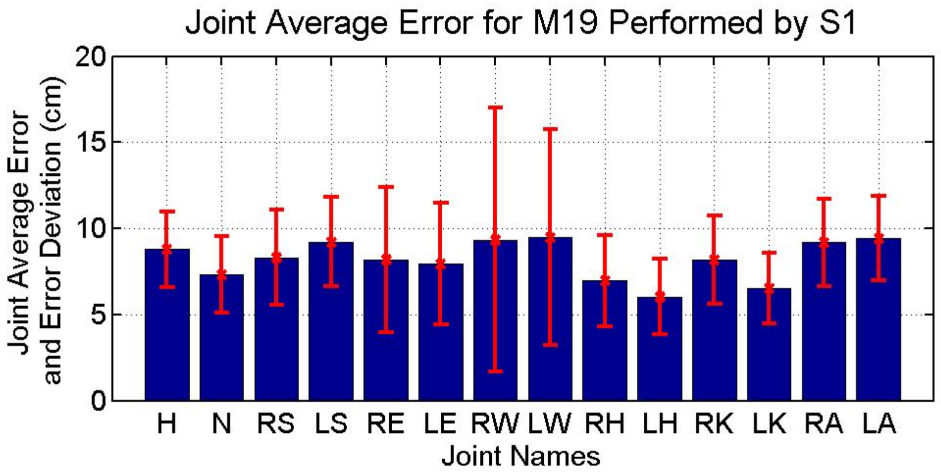


Figure 6.20: JAE with Deviation for M19 by S1

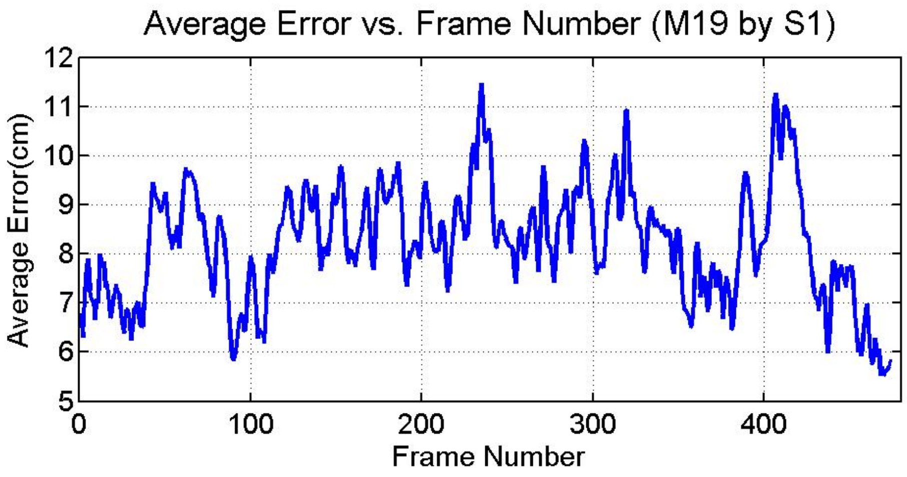
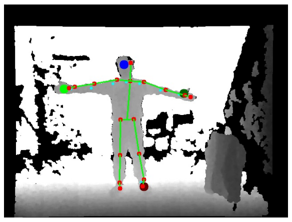
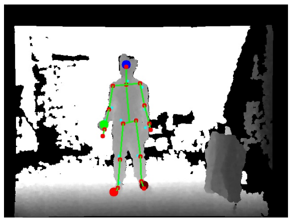
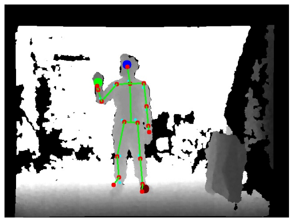
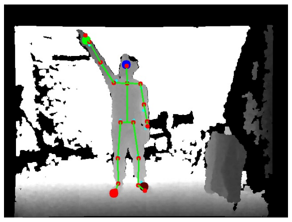


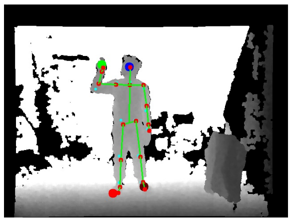
Figure 6.21: AE over Frames for M19 by S1

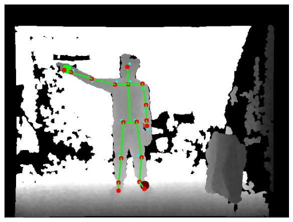

6.22.1: Frame0

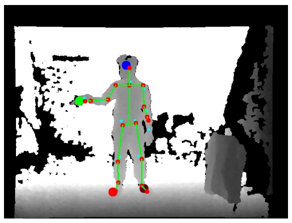

6.22.2: Frame61

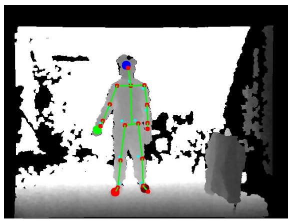

6.22.3: Frame73

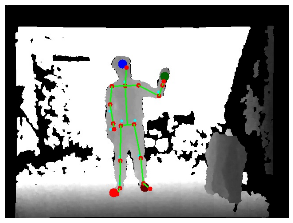

6.22.4: Frame101

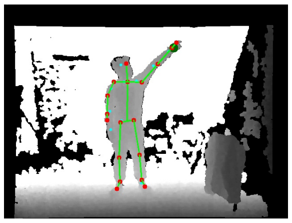

6.22.5: Frame129

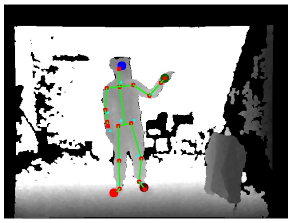

6.22.6: Frame170

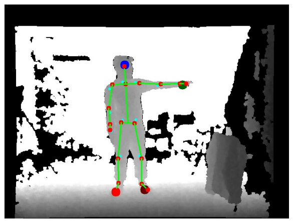

6.22.7: Frame195

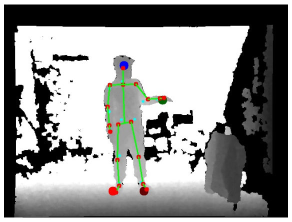

6.22.8: Frame247

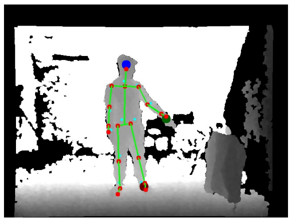

6.22.9: Frame274


6.22.10: Frame294


6.22.11: Frame330


6.22.12: Frame359


6.22.13: Frame379


6.22.14: Frame391

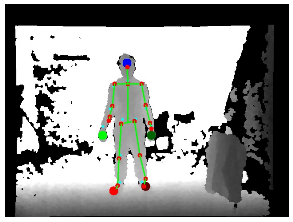

6.22.15: Frame423

Figure 6.22: Extracted Frames for M19 by S1

6.5.2 Selected Failed Clips

Clip 1: Motion M5 by Actor S3

In this clip, actor S3 performs the motion of arms crossing in front of the chest. 12 key frames are sampled from the tracking video, as shown in Figure 6.22. JAE with standard deviation is plotted in Figure 6.20, and AE over frames is plotted in Figure 6.21.

This is a very challenging clip because the arm crossing motion happens in a small space and causes strong occlusions. In Figure 6.22, we can see that the system starts to lose track of the two arms from Frame 82, when the two arms move to the front of the torso. Although the two hands are correctly detected in Frame 82, due to the fast movement speed and insufficient number of particles at the shoulders, the tracker fails to follow. Starting from Frame 110, the two arms cross in front of the chest, and the two hands are close to each other, and to the head. This makes the head partially occluded, and therefore, the head and the hands are difficult to detect. In Frame 110, the right elbow is detected as the right hand, and the right hand is detected as the left hand. In Frame 130, the two elbows are recognized as the two hands, and the right hand is recognized as the head. The strict angle limits are another possible factor that prevent the arms from folding closely in front of the chest. However, when the actor recovers to her initial pose, the endpoint body parts are correctly detected from Frame 182, and the tracker starts to recover after Frame 182. In the recovery process, the left arm struggles to reach its correct pose because it is forced to find a path inside the contour. The particles that drive the skeleton to move outside the contour are penalized heavily in the ***Depth Cue***. This issue can be solved by solving inverse kinematics directly using the detected hand locations.

The observations are confirmed by the numerical error plots. Figure 6.23 shows that the left wrist has the largest error and largest deviation. In Figure 6.24, there is a clear peak from Frame 100 to Frame 180, which corresponds to the period when the two arms are crossed in front of the chest.

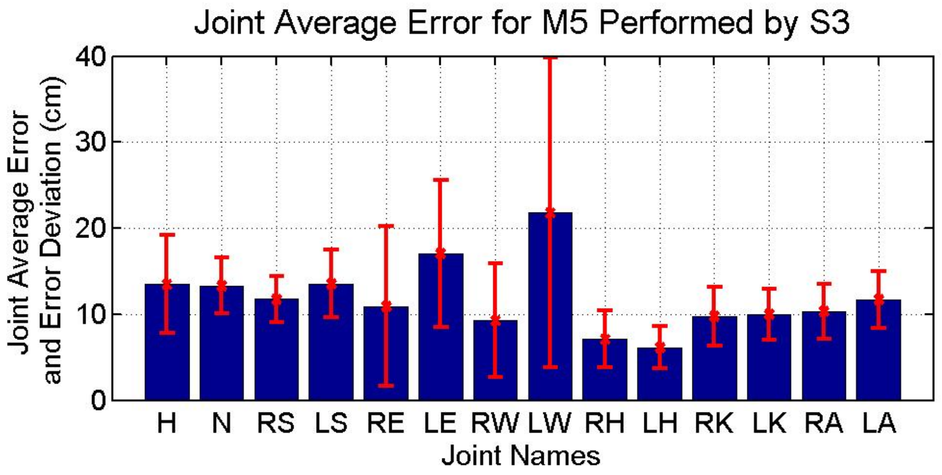


Figure 6.23: JAE with Deviation for M5 by S3

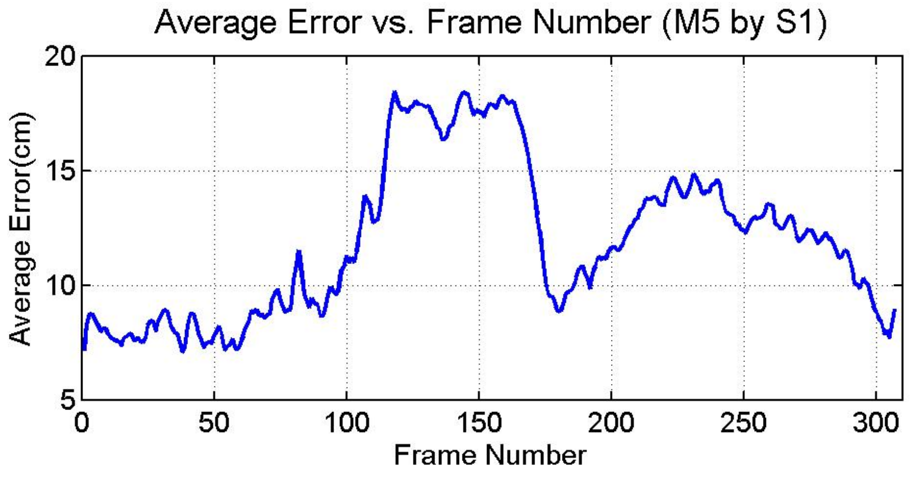
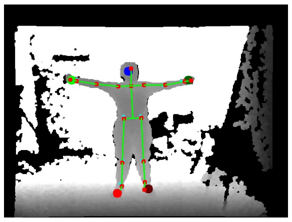
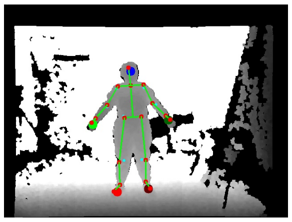
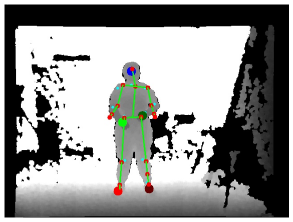
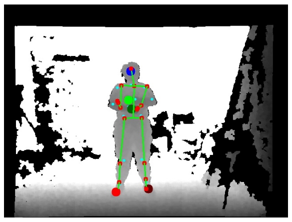


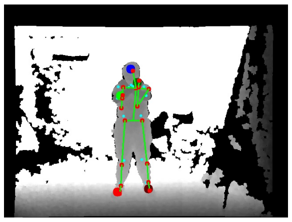
Figure 6.24: AE over Frames for M5 by S3

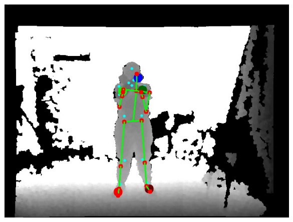

6.25.1: Frame0

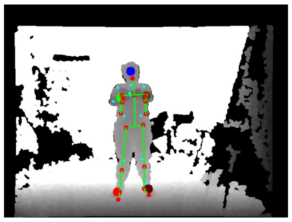

6.25.2: Frame58

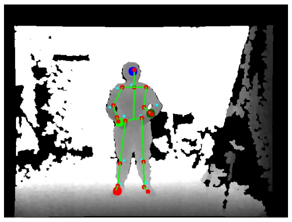

6.25.3: Frame82

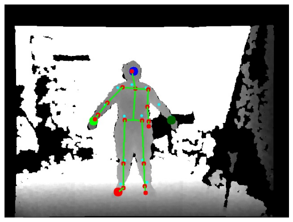

6.25.4: Frame94

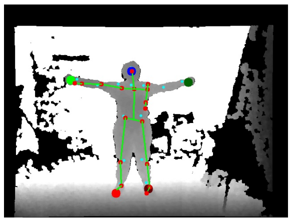

6.25.5: Frame110

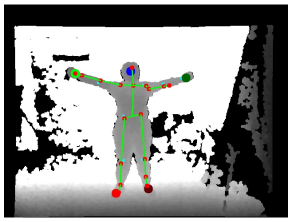

6.25.6: Frame130


6.25.7: Frame158


6.25.8: Frame182


6.25.9: Frame198


6.25.10: Frame230


6.25.11: Frame290

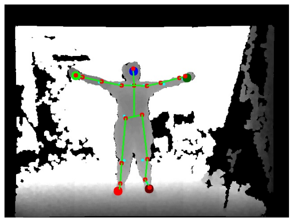

6.25.12: Frame302

Figure 6.25: Extracted Frames for M5 by S3

Clip 2: Motion M14 by Actor S4

In this clip, actor S4 performs the jumping motion, and raises the arms simultaneously. 12 frames are sampled from the tracking video as shown in Figure 6.28. JAE with standard deviation is plotted in Figure 6.26, and AE over frames is plotted in Figure 6.27.

In Figure 6.28, we can see that the tracker fails to track the arms from Frame 55. The torso also fails to follow the jumping motion from Frame 61. There are mainly two reasons for this failure. Firstly, the motion is too fast. There are only 10 frames from Frame 51 where the hands are at their lowest position, to Frame 61 where the hands are at their highest position. That is just one third of a second. The torso motion is also very fast. There are only 11 frames from Frame 57 where the torso is at its lowest position, to Frame 68 where the torso reaches its highest position. The dynamic model for making configuration predictions is far slower than this. Especially for the root translation, in order to stabilise the torso, its variance is set to be very small. Secondly, the depth map of the body parts deforms due to the fast motion and the relatively low frame rate. The shapes of the hands change, and become hard to detect (Figure 6.28.6). In Figure 6.28.7, the hands are out of the image and become impossible to be detected. As a result, the tracker fails. However, the system starts to recover once the endpoint body parts are correctly detected, and successfully recovers from Frame 91.

In Figure 6.26, the JAEs of both wrists are larger than 20cm, with a deviation of around 40cm. In Figure 6.27, there is a clear sharp peak from Frame 50 to Frame 90, when the jumping happens.

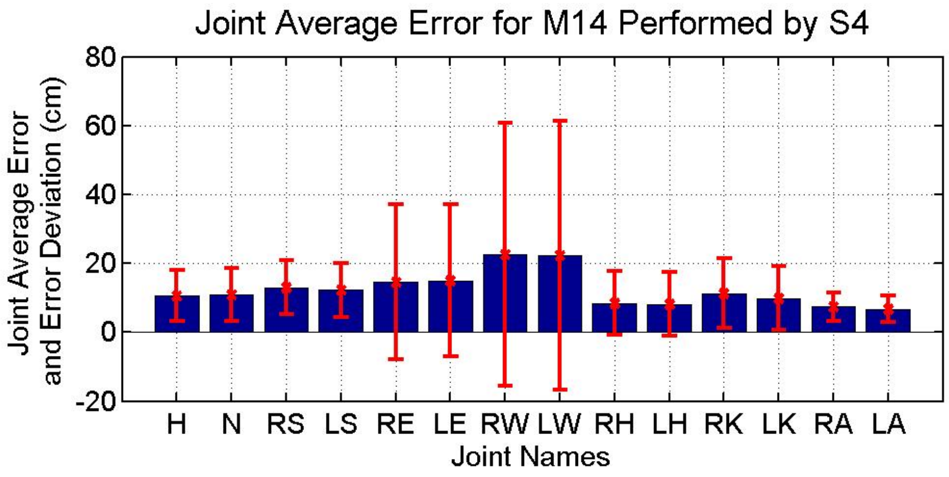


Figure 6.26: JAE with Deviation for M14 by S4

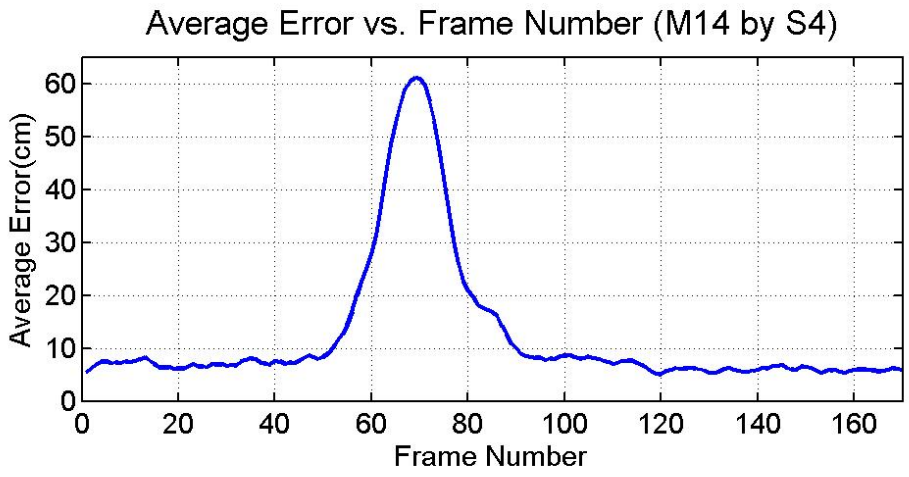
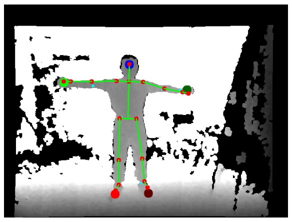
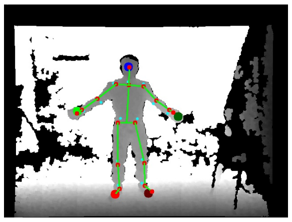
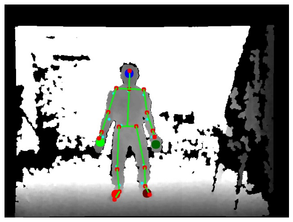
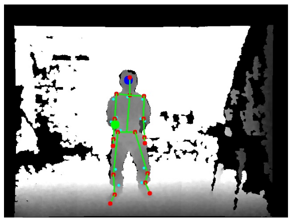


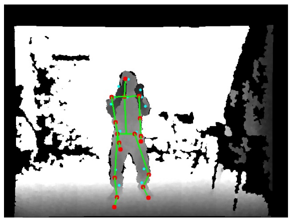
Figure 6.27: AE over Frames for M14 by S4

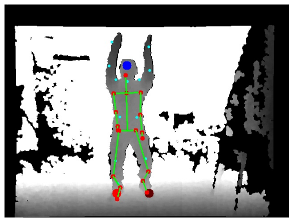

6.28.1: Frame1

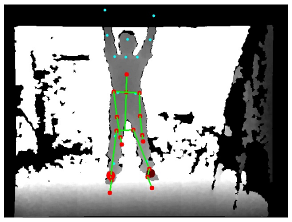

6.28.2: Frame45

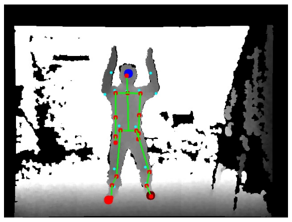

6.28.3: Frame51

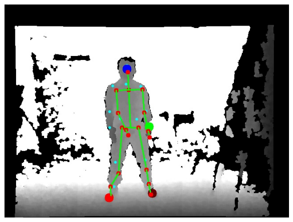

6.28.4: Frame55

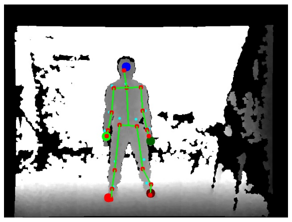

6.28.5: Frame57

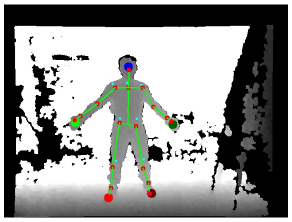

6.28.6: Frame61


6.28.7: Frame68


6.28.8: Frame78


6.28.9: Frame87


6.28.10: Frame91


6.28.11: Frame102

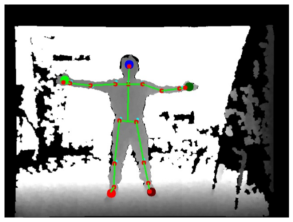

6.28.12: Frame126

Figure 6.28: Extracted Frames for M14 by S4

6.6 Summary

In this chapter, we described a novel full body pose tracking system which incorporates the body part detection algorithm into the particle filter framework. The LSC feature was generalized to not only recognize the endpoint body parts, but also to detect the limbs simultaneously. A new dataset was designed and collected for evaluating the algorithm, containing synchronized colour videos, depth videos, and ground truth captured from a marker based commercial MOCAP system. The proposed system runs at a frame rate of 4Hz on a laptop using one quarter of the CPU. The Overall Average Error for all the 110 videos was 10.7cm, with a standard deviation of 3.1cm. This system shows that efficient, accurate and robust human full body pose tracking can be achieved with a very simple, skeleton-only human model.

Chapter 7

Conclusions and Future Work

The design of an accurate, efficient and robust algorithm for full body pose tracking is a very challenging task. In this thesis, we developed a system for full body pose tracking and proposed an efficient body part recognition algorithm for providing strong bottom-up features. By integrating the body part detection algorithm into the tracking framework, it was shown that accurate, efficient and robust tracking can be achieved using a simple, skeleton-only model.

7.1 Conclusions

First, we presented a model-based full body human pose tracking system using a particle filter. A configurable human model was constructed, including a skeleton model and an outer shape model. A basic particle filter was implemented for upper body tracking, which fused time efficient cues from a monocular image sequence. It was demonstrated that as the number of particles increased, the error decreased monotonically. The system achieved real-time tracking for constrained motions on a standard PC. For this system, missing depth information was the main error source. From the experiments, it was shown that the two cues contributed equally to the tracking.

3D surfaces were then added to the human model, and a full body tracking system was developed for more general motions. Partitioned sampling was applied to deal with the high dimensionality problem, and the system was shown to be capable of running in near real-time in a single thread on a standard PC. Multiple visual cues were investigated, including a newly developed explicit depth cue. A comprehensive study on the relative importance of different cues was conducted, and the quantitative results were reported. The results demonstrated that the explicit depth cue was the most important cue, followed by the distance cue.

Second, we proposed a novel algorithm for detecting and identifying endpoint body parts from depth images. We proposed a novel Local Shape Context (LSC) Descriptor specifically for describing the shape features of body parts in depth images. This descriptor described the local shape of different body parts with respect to a given reference point on the human silhouette, and was shown to be effective at detecting and classifying endpoint body parts, while being computationally efficient. A new type of interest point was defined based on the LSC Descriptor. A hierarchical interest point selection algorithm was designed to further conserve computational resources. The detected endpoint body parts were then classified according to learned models of each class based on the LSC feature.

Finally, we described a novel full body tracking algorithm which incorporated the body part detection algorithm into the particle filter framework. The LSC feature was generalized to not only recognize the endpoint body parts, but also detect the limbs simultaneously. The use of this new feature enabled efficient and accurate tracking of full body motions by using a very simple, skeleton-only human model. A new dataset was collected for evaluating the algorithm, containing synchronized colour videos, depth videos, and ground truth captured from a marker based commercial MOCAP system. The Overall Average Error for all the 110 videos was 10.7cm, with a standard deviation of 3.1cm. The system showed good overall accuracy and robustness, but the accuracy could differ significantly for different motions or different subjects. The system failed when the subject was performing fast motions such as jumping, or motions with hard to detect body parts, such as crossing arms in front of the torso. However the system was also shown to be capable of recovering from the wrong poses when the body parts can be detected. The system ran in 4Hz on a laptop by using just one quarter of the CPU.

7.2 Future Work

As discussed in Section 6.1, only one statistical feature was used in the final integrated system, and the body parts were classified by simple thresholding with an empirically derived threshold. In the future, more statistical features can be extracted. A representative and comprehensive training dataset can be established and different classifiers can be applied for more general and accurate body part segmentation.

In the current system, the initial pose and the parameters of the human model are assumed to be known. However, these may not be known accurately in a real application scenario. Therefore, automatic initialization and bone length estimation should be considered in future work. Also, different particle filter implementations can be investigated to compare with the partitioned particle filter, such as the Annealed Particle Filter. Since the endpoint body parts are detected, inverse kinematics can be used and integrated into the importance sampling to improve the system's efficiency and robustness. Also, the algorithms can be optimized and implemented to run in parallel to further increase efficiency.

References

[1] J. Aggarwal and Q. Cai, "Human motion analysis: A review," *Computer Vision and Image Understanding (CVIU)*, vol. 73, no. 3, pp. 428–440, 1999.

[2] D. Gavrila, "The visual analysis of human movement: A survey," *Computer Vision and Image Understanding (CVIU)*, vol. 73, no. 1, pp. 82–98, 1999.

[3] T. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding (CVIU)*, vol. 81, no. 3, pp. 231–268, 2001.

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

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

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

[7] "Motion capture systems from vicon." `http://www.vicon.com/`.

[8] "Motion anyalysis corporation, the motion capture leader." `http://www.motionanalysis.com/`.

[9] "Optitrack - optical motion capture systems and tracking software." `http://www.naturalpoint.com/optitrack/`.

[10] "Phasespace motion capture." `http://www.phasespace.com/`.

[11] "Motion capture systems - research-grade for life science - ndi." http://www.ndigital.com/lifesciences/products-motioncapturesystems.php.

[12] "Simi reality motion system." http://www.simi.de/en/.

[13] "Organic motion: Marker-less motion capture." http://organicmotion.com/.

[14] G. Mori and J. Malik, "Estimating human body configurations using shape context matching," *7th European Conference on Computer Vision (ECCV), 2002*, pp. 150–180, 2002.

[15] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *9th IEEE International Conference on Computer Vision (ICCV), 2003*, pp. 750–757, IEEE, 2003.

[16] A. Agarwal and B. Triggs, "3d human pose from silhouettes by relevance vector regression," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2004*, vol. 2, pp. II–882, IEEE, 2004.

[17] R. Rosales and S. Sclaroff, "Learning body pose via specialized maps," in *Advances in Neural Information Processing Systems (NIPS) 15*, vol. 1, p. 2, 2002.

[18] A. Agarwal and B. Triggs, "Monocular human motion capture with a mixture of regressors," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)-Workshops, 2005*, pp. 72–72, IEEE, 2005.

[19] 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 *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2011*, pp. 1297–1304, IEEE, 2011.

[20] J. Deutscher and I. Reid, "Articulated body motion capture by stochastic search," *International Journal of Computer Vision (IJCV)*, vol. 61, no. 2, pp. 185–205, 2005.

[21] J. Gall, B. Rosenhahn, T. Brox, and H. Seidel, "Optimization and filtering for human motion capture," *International Journal of Computer Vision (IJCV)*, vol. 87, no. 1, pp. 75–92, 2010.

[22] T. Cham and J. Rehg, "A multiple hypothesis approach to figure tracking," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 1999*, vol. 2, IEEE, 1999.

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

[24] J. Bandouch, F. Engstler, and M. Beetz, "Accurate human motion capture using an ergonomics-based anthropometric human model," in *Proceedings of the 5th Conference on Articulated Motion and Deformable Objects (AMDO), 2008*, pp. 248–258, Springer, 2008.

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

[26] R. Kehl and L. Gool, "Markerless tracking of complex human motions from multiple views," *Computer Vision and Image Understanding (CVIU)*, vol. 104, no. 2, pp. 190–209, 2006.

[27] L. Ballan, A. Taneja, J. Gall, L. Van Gool, and M. Pollefeys, "Motion capture of hands in action using discriminative salient points," in *12th European Conference on Computer Vision (ECCV), 2012*, pp. 640–653, 2012.

[28] V. John, E. Trucco, and S. Ivekovic, "Markerless human articulated tracking using hierarchical particle swarm optimisation," *Image and Vision Computing*, vol. 28, no. 11, pp. 1530–1547, 2010.

[29] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Tracking the articulated motion of two strongly interacting hands," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2012*, pp. 1862–1869, IEEE, 2012.

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

[31] P. Davis and P. Rabinowitz, *Numerical integration.* Blaisdell Publishing Company London, 1967.

[32] H. Sidenbladh, M. Black, and D. Fleet, "Stochastic tracking of 3d human figures using 2d image motion," in *6th European Conference on Computer Vision (ECCV), 2000*, pp. 702–718, Springer, 2000.

[33] R. Urtasun, D. Fleet, A. Hertzmann, and P. Fua, "Priors for people tracking from small training sets," in *10th IEEE International Conference on Computer Vision (ICCV), 2005*, vol. 1, pp. 403–410, IEEE, 2005.

[34] J. MacCormick and M. Isard, "Partitioned sampling, articulated objects, and interface-quality hand tracking," *6th European Conference on Computer Vision (ECCV), 2000*, pp. 3–19, 2000.

[35] P. Azad, A. Ude, T. Asfour, and R. Dillmann, "Stereo-based markerless human motion capture for humanoid robot systems," in *IEEE International Conference on Robotics and Automation (ICRA), 2007*, pp. 3951–3956, IEEE, 2007.

[36] Y. Sun, M. Bray, A. Thayananthan, B. Yuan, and P. Torr, "Regression-based human motion capture from voxel data," in *Proceedings of the 17th British Machine Vision Conference (BMVC), 2006*, Citeseer, 2006.

[37] A. Agarwal and B. Triggs, "Recovering 3d human pose from monocular images," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 28, no. 1, pp. 44–58, 2006.

[38] M. Fontmarty, F. Lerasle, and P. Danes, "Data fusion within a modified annealed particle filter dedicated to human motion capture," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2007*, pp. 3391–3396, IEEE, 2007.

[39] Y. Chen, C. Huang, and L. Fu, "Upper body tracking for human-machine interaction with a moving camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009*, pp. 1917–1922, IEEE, 2009.

[40] D. Kulic, D. Lee, C. Ott, and Y. Nakamura, "Incremental learning of full body motion primitives for humanoid robots," in *8th IEEE-RAS International Conference on Humanoid Robots (Humanoid), 2008*, pp. 326–332, IEEE, 2008.

[41] M. Leventon and W. Freeman, "Bayesian estimation of 3-d human motion from an image sequence," tech. rep., MERL: A Mitsubishi Electric Research Laboratory, 1998.

[42] M. Sigalas, H. Baltzakis, and P. Trahanias, "Visual tracking of independently moving body and arms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009*, pp. 3005–3010, IEEE, 2009.

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

[44] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, no. 4, pp. 509–522, 2002.

[45] A. Bissacco, M. Yang, and S. Soatto, "Fast human pose estimation using appearance and motion via multi-dimensional boosting regression," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2007*, pp. 1–8, IEEE, 2007.

[46] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas, "Discriminative density propagation for 3d human motion estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005*, vol. 1, pp. 390–397, IEEE, 2005.

[47] A. Fathi and G. Mori, "Human pose estimation using motion exemplars," in *11th IEEE International Conference on Computer Vision (ICCV), 2007*, pp. 1–8, IEEE, 2007.

[48] H. Ning, W. Xu, Y. Gong, and T. Huang, "Discriminative learning of visual words for 3d human pose estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2008*, pp. 1–8, IEEE, 2008.

[49] R. Urtasun and T. Darrell, "Sparse probabilistic regression for activity-independent human pose inference," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2008*, pp. 1–8, IEEE, 2008.

[50] R. Poppe and M. Poel, "Comparison of silhouette shape descriptors for example-based human pose recovery," in *7th International Conference on Automatic Face and Gesture Recognition (FGR), 2006*, pp. 541–546, IEEE, 2006.

[51] J. Bandouch, F. Engstler, and M. Beetz, "Evaluation of hierarchical sampling strategies in 3d human pose estimation," in *Proceedings of the 19th British Machine Vision Conference (BMVC), 2008*, vol. 14, pp. 18–21, 2008.

[52] J. Deutscher, A. Davison, and I. Reid, "Automatic partitioning of high dimensional search spaces associated with articulated body motion capture," in *IEEE Computer Society Conference Conference on Computer Vision and Pattern Recognition (CVPR), 2001*, vol. 2, pp. II–669, IEEE, 2001.

[53] A. Balan, L. Sigal, and M. Black, "A quantitative evaluation of video-based 3d person tracking," in *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005*, pp. 349–356, IEEE, 2005.

[54] A. Balan and M. Black, "An adaptive appearance model approach for model-based articulated object tracking," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2006*, vol. 1, pp. 758–765, IEEE, 2006.

[55] L. Sigal, A. Balan, and M. Black, "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," *International Journal of Computer Vision (IJCV)*, vol. 87, no. 1, pp. 4–27, 2010.

[56] I. Mikić, M. Trivedi, E. Hunter, and P. Cosman, "Human body model acquisition and tracking using voxel data," *International Journal of Computer Vision (IJCV)*, vol. 53, no. 3, pp. 199–223, 2003.

[57] F. Caillette and T. Howard, "Real-time markerless human body tracking with multi-view 3-d voxel reconstruction," in *Proceedings of the 15th British Machine Vision Conference (BMVC), 2004*, vol. 2, pp. 597–606, 2004.

[58] S. Lin and I. Chang, "3d human motion tracking using progressive particle filter," *Advances in Visual Computing*, pp. 833–842, 2008.

[59] P. Azad, T. Asfour, and R. Dillmann, "Robust real-time stereo-based markerless human motion capture," in *8th IEEE-RAS International Conference on Humanoid Robots (Humanoid), 2008*, pp. 700–707, IEEE, 2008.

[60] M. Isard and A. Blake, "Icondensation: Unifying low-level and high-level tracking in a stochastic framework," in *5th European Conference on Computer Vision (ECCV), 1998*, pp. 893–908, Springer, 1998.

[61] D. Grest, J. Woetzel, and R. Koch, "Nonlinear body pose estimation from depth images," in *27th Annal Symposium of the German Association for Pattern Recognition (DAGM), 2005*, vol. 27, p. 285, Springer, 2005.

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

[63] M. Bray, E. Koller-Meier, and L. Van Gool, "Smart particle filtering for high-dimensional tracking," *Computer Vision and Image Understanding (CVIU)*, vol. 106, no. 1, pp. 116–129, 2007.

[64] L. Sigal, A. Balan, and M. Black, "Combined discriminative and generative articulated pose and non-rigid shape estimation," *Advances in Neural Information Processing Systems (NIPS)*, vol. 20, pp. 1337–1344, 2007.

[65] M. Siddiqui and G. Medioni, "Real time limb tracking with adaptive model selection," in *18th International Conference on Pattern Recognition (ICPR), 2006*, vol. 4, pp. 770–773, IEEE, 2006.

[66] I. Haritaoglu, D. Harwood, and L. Davis, "Ghost: A human body part labeling system using silhouettes," in *14th International Conference on Pattern Recognition (ICPR), 1998*, vol. 1, pp. 77–82, IEEE, 1998.

[67] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *10th IEEE International Conference on Computer Vision (ICCV), 2005*, vol. 1, pp. 90–97, IEEE, 2005.

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

[69] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard, "Tracking loose-limbed people," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2004*, vol. 1, pp. I–421, IEEE, 2004.

[70] L. Sigal, M. Isard, H. Haussecker, and M. Black, "Loose-limbed people: Estimating 3d human pose and motion using non-parametric belief propagation," *International Journal of Computer Vision (IJCV)*, vol. 98, no. 1, pp. 15–48, 2012.

[71] "Kinect for xbox 360 from microsoft corp." http://www.xbox.com/en-US/Kinect.

[72] Z. Bubnicki, *Modern control theory*. Springer, 2005.

[73] M. Arulampalam, S. Maskell, N. Gordon, T. Clapp, D. Sci, T. Organ, and S. Adelaide, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[74] Y. Ho and R. Lee, "A bayesian approach to problems in stochastic estimation and control," *IEEE Transactions on Automatic Control*, vol. 9, no. 4, pp. 333–339, 1964.

[75] G. Welch and G. Bishop, "An introduction to the kalman filter," 1995.

[76] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *The IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC), 2000*, pp. 153–158, IEEE, 2000.

[77] C. P. Robert, G. Casella, and C. P. Robert, *Monte Carlo statistical methods*, vol. 2. Springer New York, 1999.

[78] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *Radar and Signal Processing, IEE Proceedings F*, vol. 140, pp. 107–113, IET, 1993.

[79] J. MacCormick and A. Blake, "A probabilistic exclusion principle for tracking multiple objects," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 572–578, IEEE, 1999.

[80] K. Kanazawa, D. Koller, and S. Russell, "Stochastic simulation algorithms for dynamic probabilistic networks," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 346–351, Morgan Kaufmann Publishers Inc., 1995.

[81] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[82] N. Bergman, "Recursive bayesian estimation: navigation and tracking applications," *Department of Electrical Engineering, Linköping University, Linköping Studies in Science and Technology. Doctoral dissertation*, no. 579, 1999.

[83] J. Liu and R. Chen, "Sequential monte carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998.

[84] J. Hol, T. Schon, and F. Gustafsson, "On resampling algorithms for particle filters," in *IEEE Nonlinear Statistical Signal Processing Workshop, 2006*, pp. 79–82, IEEE, 2006.

[85] W. Gilks and C. Berzuini, "Following a moving targetmonte carlo inference for dynamic bayesian models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 1, pp. 127–146, 2002.

[86] C. Musso, N. Oudjane, and F. LeGland, "Improving regularised particle filters," 2001.

[87] J. MacCormick, *Probabilistic modelling and stochastic algorithms for visual localisation and tracking.* PhD thesis, University of Oxford, 1999.

[88] "CMU graphics lab motion capture database." http://mocap.cs.cmu.edu/.

[89] Z. Li and D. Kulic, "Particle filter based human motion tracking," in *11th International Conference on Control Automation Robotics & Vision (ICARCV), 2010*, pp. 555–560, IEEE, 2010.

[90] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library.* O'Reilly Media, Incorporated, 2008.

[91] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, no. 6, pp. 679–698, 1986.

[92] M. Heath, S. Sarkar, T. Sanocki, and K. Bowyer, "Comparison of edge detectors: a methodology and initial study," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 1996*, pp. 143–148, IEEE, 1996.

[93] Z. Li and D. Kulic, "A stereo camera based full body human motion capture system using a partitioned particle filter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010*, pp. 3428–3434, IEEE, 2010.

[94] R. Douc and O. Cappé, "Comparison of resampling schemes for particle filtering," in *4th International Symposium on Image and Signal Processing and Analysis (ISPA), 2005*, pp. 64–69, IEEE, 2005.

[95] M. Piccardi, "Background subtraction techniques: a review," in *IEEE International Conference on Systems, Man and Cybernetics, 2004*, vol. 4, pp. 3099–3104, IEEE, 2004.

[96] Z. Li and D. Kulic, "Local shape context based real-time endpoint body part detection and identification from depth images," in *Canadian Conference on Computer and Robot Vision (CRV), 2011*, pp. 219–226, IEEE, 2011.

[97] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Stanford time-of-flight motion capture dataset." http://ai.stanford.edu/~varung/cvpr10/.

[98] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *15th Berkeley symposium on mathematical statistics and probability*, vol. 1, p. 14, 1967.

[99] S. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[100] G. Carpaneto, S. Martello, and P. Toth, "Algorithms and codes for the assignment problem," *Annals of operations research*, vol. 13, no. 1, pp. 191–223, 1988.