4-2016

On the 3D point cloud for human-pose estimation

Kai-Chi Chan
Purdue University

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By CHAN, KAI CHI

Entitled
ON THE 3D POINT CLOUD FOR HUMAN-POSE ESTIMATION

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

C-S George Lee
Co-chair

Cheng-Kok Koh
Co-chair

Yung-Hsiang Lu

Zygmunt Pizlo

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): C-S George Lee and Cheng-Kok Koh

Approved by: Venkataramanan Balakrishnan 4/23/2016

Head of the Departmental Graduate Program Date

ON THE 3D POINT CLOUD FOR HUMAN-POSE ESTIMATION

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Kai-Chi Chan

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2016

Purdue University

West Lafayette, Indiana

This thesis is dedicated to my family.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my major advisors Professor Cheng-Kok Koh and Professor C. S. George Lee for the precious support of my Ph.D study, their valuable suggestions and guidance on my research direction, generous sharing of their professional and personal experiences, and encouragement in all the time of research and writing of this thesis.

Besides my advisors, I would like to thank Professor Yung-Hsiang Lu and Professor Zygmunt Pizlo for taking their valuable time to serve on my advisory committee, their insightful comments and encouragement. I would like to acknowledge the support from the Visual Perception Laboratory for conducting experiments. Especially, I would like to thank Professor Zygmunt Pizlo and Tae Kyu (Terry) Kwon for helping my human-motion-capturing experiments. Many thanks go to my colleagues Hyungju (Andy) Park, Yan Gu, and friends who have given me assistance and encouragements during the course of my study.

A special thanks to my parents, my brothers, my wife and my baby for supporting me spiritually throughout my studies. Without their precious support, it would not be possible to conduct this research.

Finally, I would like to acknowledge the financial support from the National Science Foundation. This research was supported in part by the National Science Foundation under Grants CNS-0958487, CNS-0960061 and IIS-0916807. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

LIST OF TABLES

ABSTRACT

Chan, Kai-Chi Ph.D., Purdue University, May 2016. On the 3D Point Cloud for Human-Pose Estimation. Major Professors: Cheng-Kok Koh and C. S. George Lee.

This thesis aims at investigating methodologies for estimating a human pose from a 3D point cloud that is captured by a static depth sensor. Human-pose estimation (HPE) is important for a range of applications, such as human-robot interaction, healthcare, surveillance, and so forth. Yet, HPE is challenging because of the uncertainty in sensor measurements and the complexity of human poses. In this research, we focus on addressing challenges related to two crucial components in the estimation process, namely, human-pose feature extraction and human-pose modeling.

In feature extraction, the main challenge involves reducing feature ambiguity. We propose a 3D-point-cloud feature called *viewpoint and shape feature histogram* (VISH) to reduce feature ambiguity by capturing geometric properties of the 3D point cloud of a human. The feature extraction consists of three steps: 3D-point-cloud pre-processing, hierarchical structuring, and feature extraction. In the pre-processing step, 3D points corresponding to a human are extracted and outliers from the environment are removed to retain the 3D points of interest. This step is important because it allows us to reduce the number of 3D points by keeping only those points that correspond to the human body for further processing. In the hierarchical structuring, the pre-processed 3D point cloud is partitioned and replicated into a tree structure as nodes. Viewpoint feature histogram (VFH) and shape features are extracted from each node in the tree to provide a descriptor to represent each node. As the features are obtained based on histograms, coarse-level details are highlighted in large regions and fine-level details are highlighted in small regions. Therefore, the features from the point cloud in the tree can capture coarse level to fine level information to reduce feature ambiguity.

In human-pose modeling, the main challenges involve reducing the dimensionality of human-pose space and designing appropriate factors that represent the underlying probability distributions for estimating human poses. To reduce the dimensionality, we propose a non-parametric action-mixture model (AMM). It represents high-dimensional human-pose space using low-dimensional manifolds in searching human poses. In each manifold, a probability distribution is estimated based on feature similarity. The distributions in the manifolds are then redistributed according to the stationary distribution of a Markov chain that models the frequency of human actions. After the redistribution, the manifolds are combined according to a probability distribution determined by action classification. Experiments were conducted using VISH features as input to the AMM. The results showed that the overall error and standard deviation of the AMM were reduced by about 7.9% and 7.1%, respectively, compared with a model without action classification.

To design appropriate factors, we consider the AMM as a Bayesian network and propose a mapping that converts the Bayesian network to a neural network called NN-AMM. The proposed mapping consists of two steps: structure identification and parameter learning. In structure identification, we have developed a bottom-up approach to build a neural network while preserving the Bayesian-network structure. In parameter learning, we have created a part-based approach to learn synaptic weights by decomposing a neural network into parts. Based on the concept of distributed representation, the NN-AMM is further modified into a scalable neural network called NND-AMM. A neural-network-based system is then built by using VISH features to represent 3D-point-cloud input and the NND-AMM to estimate 3D human poses. The results showed that the proposed mapping can be utilized to design AMM factors automatically. The NND-AMM can provide more accurate human-pose estimates with fewer hidden neurons than both the AMM and NN-AMM can. Both the NN-AMM and NND-AMM can adapt to different types of input, showing the advantage of using neural networks to design factors.

1. INTRODUCTION

1.1 Motivations and Objectives

A human pose is defined as a set of human-joint positions. Human-pose estimation (HPE) is the process of determining human-joint positions based on sensor measurements. HPE is important because many applications rely on human poses. Based on the nature of applications, they can be generally grouped into three categories: human-motion analysis, control, and surveillance. Figure 1.1 shows some applications in these three categories.

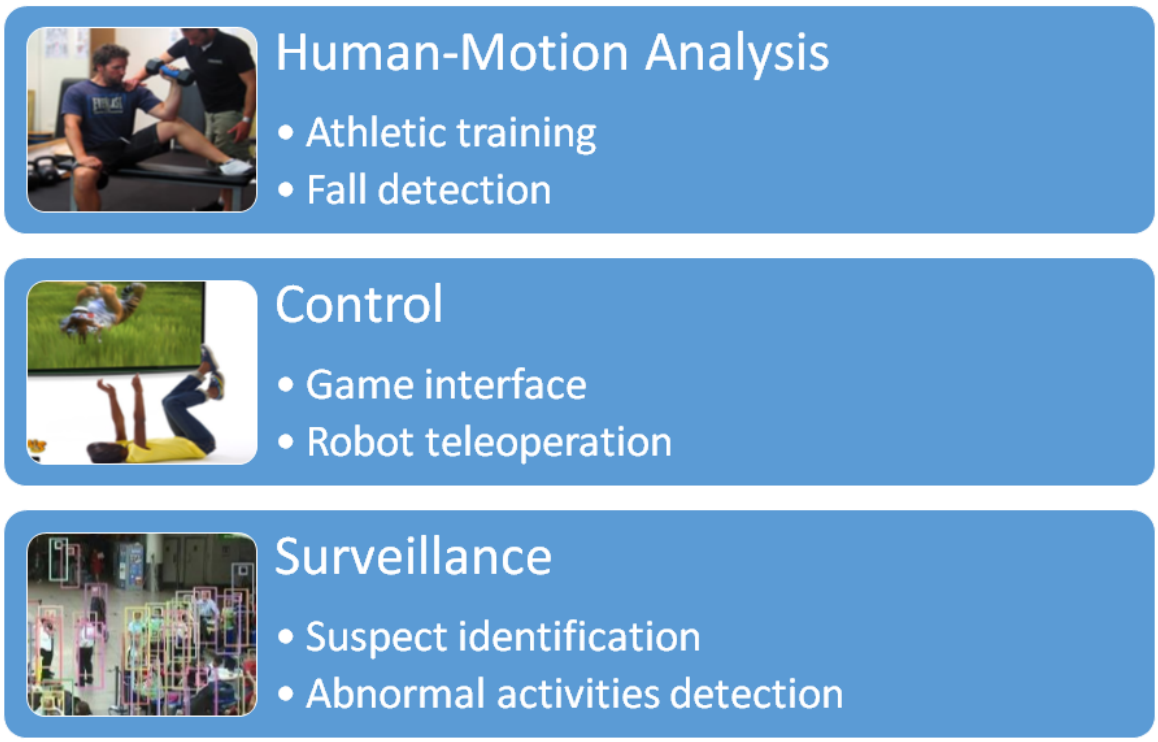


Fig. 1.1. Three main categories of applications related to human-pose estimation.

In human-motion analysis, human poses or human-body parts are tracked over a sequence of observations for different analysis purposes. For example, in athletic training,

tracking the movement of human poses can provide valuable information, such as body-part positions, in correcting body postures. Estimating human poses is useful in identifying a potential fall in healthcare facilities because falls have unique patterns and characteristics. Every year, more than 1.6 million U.S. adults are treated in the emergency room for fall-related injuries [1]. Those systems allow a rapid arrangement of assistance after a fall in order to reduce the adverse consequence of a fall.

Control applications utilize human-pose estimates to control software or machines. Compared with traditional control applications that use speech or external handheld devices such as joysticks, using human-pose estimates in control applications provides a more natural and intelligent communication interface between humans and machines. New software or machines that utilize the advance in the communication interface can also be created. For example, Microsoft Kinect is a control device that captures RGB-D data and estimates human poses for different control applications. It has been used extensively in the game industry. Many games that utilize body postures have been created. It is also used in robot teleoperation so that an operator can intuitively control a robot from a remote site through body postures. For control applications that use human-pose estimates, operators can give abstract instructions to machines and focus on high-level tasks rather than low-level implementation. For example, to control a robot to close a valve with both hands, an operator can move his/her hands accordingly and focus on turning the valve, instead of using an external device to control each hand or finger individually.

In surveillance, traditional surveillance systems have been implemented in many places, such as banks, airports, train stations, and shopping malls, for security and protection purposes. Those systems usually use visual cameras to record parts of places and create video clips for digital forensic investigation. However, there is usually a huge amount of video clips. It is tedious to manually identify useful information from them. Thus, there is a need of smart surveillance systems that analyze video clips in real time in order to identify useful information automatically and prevent abnormal events from happening. To identify or recognize individuals such as criminals in smart surveillance systems, techniques that are developed based on gait, which is a sequence of human poses, have been studied

extensively. In addition, abnormal activities such as theft can also be detected based on gait.

Traditionally, human poses are estimated using one or more charge-coupled device (CCD) cameras. Visual features, such as colors, edges, silhouettes, and textures, are used to represent human poses. Based on visual features, geometric features can be extracted by estimating depth information using stereo vision [2]. Although visual features are commonly used and are useful for reconstructing depth information, they are still ambiguous under different illumination conditions. For example, the color of a person in an image may change drastically between indoor and outdoor environments. Hence, the precision of estimating depth information is decreased when visual features are ambiguous. The decrease in the precision will then affect the consistence of geometric features. The availability of depth sensors allows us to obtain depth information directly and with less ambiguity. The sensors output 3D point clouds as observations and depth information will not be affected by the quality of visual features. Thus, it motivates us to use a depth sensor and derive a 3D-point-cloud feature for HPE.

Features are then used in a human-pose model to estimate a human pose. Modeling probability distributions of human poses is challenging because of the complexity and high dimensionality of human poses. Assumptions have been made to simplify the modeling process. One of them that is often used in the literature is that, in high dimensional space of human poses, there exist one or more low dimensional manifolds of possible human poses. It is common because some points in high dimensional space of human poses do not correspond to valid human poses. Examples of invalid human poses are those with joint angles exceeding their limits and improper length of limbs. We observe that in the literature, a human action is commonly defined as a sequence of human poses changing in a specific way. Different people perform the same action in a similar manner. Thus, we believe human-pose space is composed of action-specific manifolds. We want to exploit human actions to find low dimensional manifolds in high dimensional space of human poses. Usually, in finding low dimensional manifolds, high dimensional space of human poses is parameterized with low dimensional manifolds. For example, in principal com-

ponent analysis (PCA) [3], human-pose space is represented as a linear combination of principal components that are orthogonal to each other. To simplify the computation in modeling, we assume that the change of human actions follows a continuous-time Markov chain. Then, a human pose is estimated by finding the most probable human pose of a conditional probability distribution. Although parameterization can lead to a human-pose model that is easy to compute in training or in testing, the assumption on the relationship between human-pose space and low dimensional manifolds may not be valid. Thus, we extend the model that uses a continuous-time Markov chain to a model without making any assumption on the relationship. The extension is based on modeling human-pose space using artificial neural networks.

The advantages of using neural networks to model human-pose space are in three aspects: learning capability, distributed representation and adaptability. For the learning capability, the universal approximation theorem [4] states that a multilayer perceptron network, which is a neural network with a single hidden layer of hidden neurons, can approximate any continuous function. Thus, using neural networks to model human-pose space does not limit human-pose space to follow a particular type of conditional probability distributions of human poses. Instead, it allows a wide range of distributions to be considered in the modeling process. For the distributed representation, each concept, such as a human action or a human pose, in a neural network is represented by a group of neurons with a pattern of neural activities. Each neuron is also involved in representing a number of concepts. Representing a concept this way allows a more efficient coding than a local representation, which represents each concept by a neuron. For the adaptability, it is the ability of a neural network to handle data from different situations. For instance, a person may perform an action more frequent in one place than the other. It will affect the underlying probability distribution of human poses in modeling human-pose space. When neural networks are used in modeling human-pose space, the change in probability distributions can be adapted by varying synaptic weights in a neural network based on training data collected from a new situation. Thus, human-pose space can be adapted to various situations.

Using a neural network to model human-pose space involves two major steps: structure identification and parameter learning. To identify the structure of a neural network, we introduce a Bayesian network that represents our belief on the relationship among 3D-point-cloud features, human actions, and human poses. A Bayesian network is used because it is natural to represent causal, evidential and intercausal relationships that are indispensable to human understanding. It is a directed acyclic graph (DAG) in which nodes are random variables and absence of edges represents conditional independence assumptions on random variables. It defines a family of probability distributions in which conditional probability distributions are called factors. It can be shown that our human-pose model that uses a continuous-time Markov chain to model human actions (before the neural-network extension) is equivalent to a Bayesian network with predefined factors. Thus, in order to use a neural network to model human poses, we can use only the structure of a Bayesian network with factors being represented by some neural networks. Based on the relationship among random variables in a Bayesian network and their semantic meaning, we can systemically convert the structure of a Bayesian network to the structure of a neural network. In parameter learning, directly learning the parameters in a neural network may suffer from the vanishing-gradient problem [5] when the network has many layers. Thus, we first decompose a neural network into parts, which are neural networks. Then, we apply the backpropagation algorithm [6] to learn parameters in parts.

In this research, we investigate methodologies for estimating human poses of a human being captured by a depth sensor at a fixed position and orientation over time. We focus on reducing feature ambiguity and building a human-pose model that maps a feature to a human pose. Our research objectives are:

1. Investigating a 3D-point-cloud feature that is distinguishable among different human poses and captures the global and local properties of human poses,

2. Discovering low dimensional manifolds in high dimensional space of human poses, and

3. Designing factors in the modeling of human-pose space based on training data.

To achieve these objectives, we propose a 3D-point-cloud feature that captures both global and local properties of a human. The global properties are features extracted in large regions of a 3D point cloud of a human and hence represent coarse-level details. The local properties are feature extracted in small regions of the 3D point cloud and hence represent fine-level details. Using both properties, our proposed feature can capture coarse-level to fine-level information of the 3D point cloud. The dimensionality of human-pose space is then reduced by using low-dimensional manifolds to represent the human-pose space based on human actions. A human pose may appear in more than one action. The probability of a human pose appearing in each action is estimated by a similarity of features. A probability distribution in each manifold that represents an action is computed. The distributions in the manifolds are redistributed according to the stationary distribution of a continuous-time Markov chain that models the frequency of actions. Finally, we extend the human-pose model that uses human actions to reduce the dimensionality of human-pose space by realizing factors in the distribution of human poses using a neural network. Therefore, factors in the distribution can be designed automatically from training data.

A literature survey of existing work in the areas of feature extraction and human-pose modeling is given in the next section.

1.2 Literature Survey

In this section, we will review related works about HPE. Currently, most accurate HPE systems are developed using marker-based approaches. Those systems, such as Vicon and PhaseSpace, estimate human poses by first locating either active or passive markers that are attached on a human body. Human poses are then estimated by recovering positions of the attached markers through a triangulation algorithm. Since the early 70s, they have been used extensively in biomechanics, which is the study about internal and external forces acting on a human body. Specifically, the systems are often used as tools in two branches of biomechanics, namely kinematics and kinetics to study human movement and causes of human movement, respectively. As marker-based approaches are highly accurate, re-

searchers in biomechanics can focus on analyzing human motion without putting much effort on detecting humans and estimating human poses. In the 90s, the systems have been commonly used to generate computer animations for video games and films. Since then, the demand of HPE systems is increased. However, marker-based approaches cannot be applied on many applications, such as virtual-reality gaming and surveillance, because this type of approaches can only be used in severely restricted situations that a person is required to wear tight-fitting clothes and is surrounded by many calibrated sensors. Also, we can no longer assume attaching markers on a person. Markerless-based approaches for HPE become necessary in less constrained situations. Thus, they have drawn attention to many researchers from different research fields.

In the past, CCD cameras were commonly used in deriving most markerless-based approaches. Images that are captured by CCD cameras are 2D projections of 3D objects. Thus, depth information is lost. To generate depth information, at least two cameras are required. In 2010, Microsoft Kinect was released. It is different from CCD cameras because it can directly capture depth information by measuring a distorted infrared pattern emitted by the sensor. Since then, depth sensors become popular.

In this section, we will cover most related works that are derived based on CCD cameras. Recent works using depth sensors will also be covered. We will review related works according to two major components in HPE:

1. Features that represent the human body in observations, and

2. Human-pose models that map features to human poses.

Table 1.1 shows a summary of existing works that are included in this section. More comprehensive surveys can be found in [7] [8].

1.2.1 Feature

While features in general can represent any observation, in this thesis, we focus on HPE and thus a feature only refers to a representation of a human body in an observation.

Table 1.1 Summary of existing works for HPE.

Type	Method	Observation	Feature	Model
Generative	Hogg [9]	Visual Image	Color, Edge	WALKER
Generative	Wachter *et al.* [10]	Visual Image	Color, Edge	Invariant Extended Kalman Filter
Generative	Brand [11]	Visual Image	Silhouette	Hidden Markov Model
Generative	MacCormick and Isard [12]	Visual Image	Edge	Deformable Body Model
Generative	Deutscher *et al.* [13]	Visual Image	Edge, Silhouette	Annealed Particle Filtering
Generative	Deutscher *et al.* [14]	Visual Image	Edge ,Silhouette	Partitioned Annealed Particle Filter
Generative	Mitchelson and Hilton [15]	Visual Image	Color, Edge, Silhouette	Deformable Body Model
Generative	Plänkers and Fua [16]	Visual Image	Silhouette	Metaball Body Model
Generative	Lee and Cohen [17]	Visual Image	Color, Edge	Deformable Body Model
Generative	Anguelov *et al.* [18]	Depth Image	Depth	Markov Network
Generative	Kehl *et al.* [19] [20]	Visual Image	Color, Edge	Superellipsoid Body Model
Generative	Rodgers *et al.* [21]	Depth Image	Edge, Surface	Markov Network
Generative	Zhu *et al.* [22] [23]	Depth Image	Depth	Constraint Inverse Kinematics
Generative	Gall *et al.* [24]	Visual Image	Silhouette, SIFT	Deformable Body Model
Generative	Sun *et al.* [25]	Visual Image	Histogram of Oriented Gradients, EdgeField	Deformable Body Model
Generative	Siddiqui and Medioni [26]	Depth Image	Depth, Silhouette	Markov Chain
Generative	Lehment *et al.* [27]	Depth Image	Depth	Deformable Body Model
Generative	Charles and Everingham [28]	Depth Image	Silhouette, Depth	Pictorial Structure Model
Generative	Wang *et al.* [29]	Visual Image	Histogram of Oriented Gradients	Pictorial Structure Model
Generative	Yang and Ramanan [30]	Visual Image	Histogram of Oriented Gradients	Pictorial Structure Model
Generative	Liu *et al.* [31]	Visual Image	Color, Silhouette, SIFT	Markov Random Field
Generative	Pons-Moll *et al.* [32]	Visual Image, Inertial Sensor Data	Color, Silhouette, IMU Sensor Orientation	Deformable Body Model
Generative	Duan *et al.* [33]	Visual Image	Histogram of Oriented Gradients	Part-Based Model
Discriminative	Howe [34]	Visual Image	Silhouette	Markov Chain
Discriminative	Agarwal *et al.* [35]	Visual Image	Edge, Silhouette	Relevance Vector Machine
Discriminative	Sminchisescu *et al.* [36]	Visual Image	Silhouette	Mixture of Experts
Discriminative	Taycher *et al.* [37]	Visual Image	Edge	Conditional Random Field
Discriminative	Poppe [38]	Visual Image	Histogram of Oriented Gradients	Matching
Discriminative	Ramanan [39]	Visual Image	Color, Edge	Conditional Random Field
Discriminative	Fathi and Mori [40]	Visual Image	Motion	Matching
Discriminative	Okada and Soatto [41]	Visual Image	Histogram of Oriented Gradients	Support Vector Machine
Discriminative	Ferrari *et al.* [42]	Visual Image	Color, Edge	Conditional Random Field
Discriminative	Hofmann and Gavrila [43]	Visual Image	Color, Edge, Silhouette	Matching
Discriminative	Wang *et al.* [44]	Visual Image	Histogram of Oriented Gradients	Adaboost
Discriminative	Sedai *et al.* [45]	Visual Image	HoSC,HLAC	Relevance Vector Machine
Discriminative	Zhao *et al.* [46]	Visual Image	CP-SIFT	Regression
Discriminative	Taylor *et al.* [47]	Visual Image	Edge, Silhouette	Conditional Restricted Boltzmann Machine
Discriminative	Plagemann *et al.* [48]	Depth Image	AGEX	Boosting
Discriminative	Baak *et al.* [49]	Depth Image	Geodesic Extrema	Matching
Discriminative	Ye *et al.* [50]	Depth Image	Depth	Matching
Discriminative	Shotton *et al.* [51]	Depth Image	Depth	Randomized Decision Forest
Discriminative	Girshick *et al.* [52]	Depth Image	Depth	Regression
Discriminative	Tian *et al.* [53]	Visual Image	Silhouette	Regression
Discriminative	Straka *et al.* [54]	Visual Image	Silhouette	Matching
Discriminative	Stoll *et al.* [55]	Visual Image	Color	Sums of Spatial Gaussians
Discriminative	Sun *et al.* [56]	Depth Image	Depth	Regression
Discriminative	Taylor *et al.* [57]	Depth Image	Silhouette, Depth	Regression
Discriminative	Tian *et al.* [58]	Visual Image	Histogram of Oriented Gradients	Support Vector Machine
Discriminative	Chen and Yuille [59]	Visual Image	Edge	Convolutional Neural Network
Discriminative	Jain *et al.* [60]	Visual Image	Color	Convolutional Neural Network
Discriminative	Li *et al.* [61] [62]	Visual Image	Color	Convolutional Neural Network
Discriminative	Toshev and Szegedy [63]	Visual Image	Color	Convolutional Neural Network
Discriminative	Pfister *et al.* [64]	Visual Image	Color	Convolutional Neural Network
Discriminative	Ouyang *et al.* [65]	Visual Image	Color, Histogram of Oriented Gradients	Restricted Boltzmann Machine
Discriminative	Tompson *et al.* [66]	Visual Image	Color	Convolutional Neural Network
Hybrid	Sigal *et al.* [67] [68]	Visual Image	Color, Edge	Loose-Limbed Body Model
Hybrid	Gupta *et al.* [69]	Visual Image	Silhouette	OD-GPLVM
Hybrid	Ganapathi *et al.* [70]	Depth Image	AGEX	Dynamic Bayesian Network
Hybrid	Gall *et al.* [71]	Visual Image	Edge, Silhouette	Isomap
Hybrid	Gall *et al.* [72]	Visual Image	Color, Silhouette	Regression, Deformable Body Model
Hybrid	Yao *et al.* [73] [74]	Visual Image	Color, Edge, Silhouette, Optical Flow	Hough Forest
Hybrid	Gall *et al.* [75]	Depth Image	Depth	Clustering
Hybrid	Sedai *et al.* [76]	Visual Image	Histogram of Shape Context	Scaled Prismatic Model
Hybrid	Zuffi *et al.* [77]	Visual Image	Color, Histogram of Oriented Gradients	Deformable Structure

Features are important in HPE because of the variety of observations of humans such as human visual appearance. They are used as input to human-pose models. The choice of features often determines the choice of human-pose models. A typical approach for HPE begins with extracting appropriate features from an observation and then infers states,

such as human poses, of a model through some mappings defined by the model. While determining the most appropriate feature for HPE is useful, it is still contestable, especially in ranking features that are appropriate in all situations. It is also a common belief that combining multiple features can increase the accuracy of an estimation system. Thus, instead of ranking features, we will describe features that are commonly used. We will divide features into two types according to observations from which features are extracted. The first type is a visual feature. It is derived from 2D color/grayscale images captured by CCD cameras. The second type is a geometric feature. It is derived from 3D point clouds captured by depth sensors.

Visual Feature

One of the common visual features for HPE is histogram of oriented gradients (HOG) [78], which is derived based on the orientation histograms obtained from 2D color/grayscale images. Using HOG, Wang *et al.* [44] estimated human poses by combining the bottom-up classification for each joint position and the top-down classification for a skeleton model using the AdaBoost algorithm [79]. Wang *et al.* [29] adapted HOG to represent body parts and extended the concept of rigid body parts to hierarchical poselets that incorporated larger portions of body parts. Poppe [38] clustered HOG detected from input images using the k-nearest neighbor algorithm (k-NN) [3] to estimate 3D human poses.

Silhouettes can also be used to represent human poses by extracting edges around humans. Hofmann *et al.* [43] presented a multi-camera system to reconstruct silhouettes to represent a 3D human upper body. Brand [11] estimated human poses over time by matching body parts with the corresponding silhouettes. Based on silhouettes, Wachter *et al.* [10] derived right-elliptical cones to represent 3D human body parts.

Visual features can be combined for human-pose representation. Ramanan [39] used color and edges as cues in a conditional random field (CRF) [80]. Ferrari *et al.* [42] adapted the features in [39] and color histograms to estimate 2D upper-body human poses based on a tree-structured CRF. Pons-Moll *et al.* [32] estimated 3D human poses using silhouettes,

color and sensor data from ten inertial measurement units. Zhao *et al.* [46] proposed a corner-interest-point-based SIFT to estimate 3D human poses using Gaussian-process regression. Multi-view visual data was concatenated as one feature input to a regression model. Agarwal *et al.* [35] adopted edges and silhouettes as input to learn 3D human poses directly by relevance vector machine (RVM) [80]. Mitchelson *et al.* [15] used silhouettes, edges and color to estimate 3D human poses of multiple humans using multiple cameras. Charles and Everingham [28] proposed a shape model learning from silhouette and 3D human-pose data. The shape model was built based on the pictorial structure model (PSM) [81] and was used to model 3D human poses.

Using visual features, depth information can be estimated by stereo vision [2]. Kehl *et al*. [19] [20] proposed a multi-camera system to reconstruct the 3D surface of a human. Superellipsoid was then derived for HPE. Gall *et al*. [24] estimated human poses and the deformation of human surface jointly. A weighted least-squares method was used to fit human poses on the surface. Then, misaligned limbs were fitted using a particle-based global optimization method. Straka *et al*. [54] proposed a volumetric body-scans method. A skeletal graph was created based on voxel scooping. Then, geodesic distances between end nodes and head node were computed and matched with a template using the dynamic time warping algorithm (DTW) [82]. A human pose was estimated by matching positions of the nodes for head, limbs and inner joints from the template. Plänkers and Fua [16] proposed a surface model to represent an articulated human body. The surface was modeled using metaballs based on silhouettes. Then, the distance between the model and the observation was minimized using the Levenberg-Marquart algorithm [83].

Since visual features are derived from 2D images captured by CCD cameras, they lose depth information and are ambiguous under different illumination conditions. Thus, recent works start to use 3D-point-cloud features to reduce the ambiguities.

Geometric Feature

Using a depth sensor, Shotton *et al.* [51] used a single depth image and proposed a method to determine 3D positions of human joints. Depth image features, which were depth invariant and 3D translation invariant, were proposed. 3D positions of human joints were then estimated by classifying features on each pixel using randomized decision forests. Girshick *et al.* [52] adopted the depth image features [51] to generate candidates for each joint position using a regression forest. A tree structure of body parts was trained using a greedy-decision method and parameters were estimated using the expectation-maximization (EM) algorithm [84]. Rusu *et al.* [85] proposed the viewpoint feature histogram (VFH), which measured pan, tilt and yaw angles between 3D points. Plagemann *et al.* [48] proposed the accumulative geodesic extrema (AGEX) by extracting geodesic distances from pairs of points on a body part. Baak *et al.* [49] estimated 3D human poses by first matching 3D point clouds between consecutive depth images. Based on the difference, the 3D point cloud in the current observation was refined and matched with 3D point clouds in a human-pose database to estimate a human pose. Ye *et al.* [50] used a 3D surface mesh to represent a human, and to estimate human poses by human-pose detection.

Since the geometric features above are extracted from the whole human body, the body's properties — orientation and shape — in local regions are understated. It motivates us to investigate and derive a 3D-point-cloud feature called VISH that captures both the global and local properties of the 3D point cloud of a human body.

1.2.2 Human-Pose Model

Once features are extracted, the next step is to build a human-pose model that maps features to human poses. This section gives an overview of models that are commonly used for HPE. Models are summarized into three types: generative models, discriminative models, and hybrid models that combine the previous two models. Generative models describe joint probability distributions of human poses and features. Thus, generative models can generate all possible pairs of human poses and features according to the joint distributions.

Usually, a joint distribution is further decomposed into a prior distribution of human poses, and a distribution of features conditioned on a human pose. A significant work of using a generative model for estimating a human pose of a walking person was presented in the early 80s [9]. Walking patterns were generated in advance and an evaluation function of comparing each frame with a walking pattern was predefined. The hierarchical structure of the human body and constraints among body parts that were presented in the paper are still widely used in many related works. Generative models usually lead to a better generalization to unseen human poses compared with the discriminative models because the joint probability of features and human poses is estimated in the generative models.

Discriminative models, on the other hand, focus on the direct mapping between an input feature and a human pose without estimating the joint probability. They cannot be used to generate a feature corresponding to a human pose but they can compute the probability of a human pose being represented by a feature. The input feature is usually modeled as a constant instead of a random variable. Recently, neural networks have been shown to be useful in estimating human poses. Specifically, existing works utilize techniques in deep learning to represent observations or input features in multiple levels of abstraction, and then map the abstraction to human poses. We will include related works of neural networks for HPE in the following section as well. More comprehensive surveys about neural networks can be found in [86] [87].

Generative Model

Generative models have a long history in HPE. The high-level idea of these models is to formulate possible combinations of human poses and the corresponding features by a number of parameters. For example, parameters could be the position and orientation of a body part. An inference is then made by finding the most probable human pose that accounts for the given features of observations. The main challenge in these models is to search over a large amount of possible combinations. Searching human poses in such high dimensional space is computationally impractical. This motivates the development

of different dimension-reduction and search algorithms that utilizes properties of human poses such as the hierarchical structure of the human body.

Yang and Ramanan [30] modeled the hierarchical structure by a mixture of templates for each body part. The templates captured the contextual co-occurrence relationship between body parts. Each part was associated with a mixture component representing its type. HOG was used as an image feature and support vector machine (SVM) [88] was used to learn model parameters. Duan *et al.* [33] extended the mixture of templates [30] to a multi-scale model. The multi-scale model contained multiple layers with different numbers of body parts and different tree structures. In each layer, HOG was used to represent the observation of an image. The model between layers was represented by a tree structure. Dual decomposition was used for efficient inference. Anguelov *et al.* [18] proposed a method of rigid body-part detection to recover human poses. 3D meshes were registered to estimate the transformation between the 3D meshes and the reference mesh. Then, each point on the mesh was assigned a label representing a rigid part. Soft contiguity constraints were imposed to bias the contiguous assignment of rigid parts. Hard contiguity constraint were added to limit every rigid part to at most one connected component. EM algorithm was used to optimize the transformation and the labeling. Zhu *et al.* [22] [23] proposed a 2D HPE method based on anatomical landmarks. Landmarks were first detected and interpolated if occlusions were occurred to locate the head, neck and torso. The head, neck and trunk (HNT) templates were built and body parts were detected in the 3D point cloud from depth images. In the HNT templates, key points were further extracted to generate human-pose hypotheses based on the inverse kinematics. Then, arms were detected by blob detection. Constraints about joint limits and penetration were applied to reconstruct the poses. Human poses were estimated by fitting the detected body parts into observations. Su *et al.* [25] combined a top-down method to generate an initial human-pose estimate and a bottom-up method to iteratively refine the estimate. Gaussian process latent variable model (GPLVM) [80] was applied to match all training samples with an input to find an initial estimate. Rotation invariant EdgeField features were proposed to detect body parts

to limit searching seeds of candidates. The belief propagation (BP) inference [89] was then applied over the candidates using the maximum a posteriori (MAP) estimation.

Markov assumptions [89] are commonly applied to human-pose model to reduce the dimensionality of human-pose space. Liu *et al.* [31] proposed a maximum-a-posteriori Markov random field (MAP-MRF) method to segment two persons. Priors of the MAP-MRF were proposed based on shape and appearance features in the previous frame. Rodgers *et al.* [21] proposed a Markov network to estimate 3D human poses. Surfaces and edge discontinuities of body parts were used to model the probabilities of body parts in 3D human-pose space. Loopy belief propagation (LBP) [89] and iterative closest point (ICP) algorithm [90] were used to estimate and refine poses in the Markov network. Lee and Cohen [17] proposed a complementary jump proposal in the data-driven Markov chain Monte Carlo (MCMC) framework to estimate 3D human poses. Face, contour, color and edges were combined as a feature for the observation. The proposed model also included the dependency of image positions of body joints, such as head, and the observation. Lehment *et al.* [27] proposed an observation likelihood approximation. The likelihood was derived from the similarity between non-occluded 3D model points and observed 3D data points, and a penalized function for self collision.

Once a human pose model is defined, parameters of the model are estimated by optimizing the corresponding objective functions. Objective functions are often multimodal because human bodies of different human poses could be similar. Different optimization techniques are proposed to avoid local optimizers. Siddiqui and Medioni [26] proposed a MCMC method for HPE. The method combined a body-part detection and an observation-likelihood approach. The head of a person was first detected by the Canny edge detector in depth images. Other body parts were then detected heuristically using their 3D positions. The observation likelihood was measured based on foreground silhouettes, the Euclidean distance between the observed and estimated depth images, and the number of missing pixels in depth images. MacCormick and Isard [12] tracked the 2D position of a hand using partitioned sampling. A survival diagnostic and a survival rate were introduced to measure the reliability of particles in the particle filter. Partition sampling was able to use less parti-

cles to track a hand as an articulated object and use different numbers of particles to track different parts in the hand. Deutscher *et al.* [13] proposed the annealed particle filtering for HPE. Each limb was modeled using conic sections with elliptical cross-sections. A matching function that measured the likelihood of a possible human pose being the human pose in the scene was defined based on edges and foreground silhouette. Deutscher *et al.* [14] introduced a crossover operator in the annealed particle filter for estimating human motion. Search space was partitioned based on the variance of each particle.

Discriminative Model

Discriminative models attempt to map features directly to human poses. The mapping is usually derived in the form of a nearest-neighbor approximation, a regression or mixture distributions. These models are often fast and reliable when training data can represent situations well.

When deriving a mapping, a kinematic chain of human is usually incorporated to simplify human-pose models. Taycher *et al.* [37] modeled 3D human poses using CRF to represent a kinematic relationship. Stoll *et al.* [55] modeled both images and the kinematic relationship of a human using sums of spatial Gaussians. Sun *et al.* [56] proposed a conditional-regression-forest model and incorporated the kinematic relationship among 3D body-joint positions by introducing a latent variable. The latent variable also represented human attributes such as height, gender and torso orientation.

Human-pose space can be modeled by mixtures of simple models. Tian *et al.* [53] leveraged the learning problem of high-dimensional space of 3D human poses by introducing low-dimensional latent space for both input features and 3D human poses. The latent space was derived by the locality preserving projections (LPP) algorithm [91]. Then, Gaussian mixture model (GMM) [80] was trained to model the mapping of latent space between the input features and 3D human poses. As a result, the proposed model could deal with multimodalities in human-pose space. Sminchisescu *et al.* [36] proposed a discriminative Bayesian mixture-of-experts model to track 3D human poses based on silhouettes. Tian *et*

al. [58] proposed a discriminative model that is derived based on the generative model [30] by using latent nodes to represent the abstraction of body parts such as a left arm and a right leg. By adding the latent nodes, the relationship among nearby parts could be specified.

Regression techniques were proposed for fast computation. Sedai *et al.* [45] proposed a learning-based approach to combine shape and appearance features from single-view images to estimate 3D human poses. Taylor *et al.* [57] used a regression forest to estimate the correspondence between the features from depth and multi-view silhouette images, and points in a human model. Okada and Soatto [41] approximated the non-linear mapping between HOG and 3D human poses by multiple local linear regressors.

To reduce the jitter effect from the estimates, Howe [34] estimated human poses by minimizing three quantities, namely the Chamfer distance between a sequence of human poses from the video and the database, the temporal similarity between successive frames using Markov chain, and the smoothness among frames using quadratic splines. Fathi and Mori [40] proposed a motion exemplar approach to detect and track 2D human poses.

Neural networks with deep network architectures have been shown to achieve the state-of-the-art performance in HPE [47, 62–64]. Traditionally, neural networks have been used as function approximators that map observations such as images of humans or body parts to human poses. In the recent development of deep learning, a limited set of graphical models has been created. Graphical models in the set have been shown to be equivalent to some neural networks. Those neural networks are thus probabilistic in nature. They have been used extensively to extract features from data through modeling data distributions. Each graphical model in a limited set always has hidden random variables (neurons) that represent features at root nodes or intermediate nodes, and random variables that represent data at leaf nodes. Also, random variables are assumed to follow some specific distributions for efficient computation. With the convention that data neurons are arranged in the bottom layer, a feature (neuron) is more abstract when its corresponding layer is higher. Since neurons are stochastic, parameter-learning algorithms have been proposed to learn edge weights by considering probabilistic neural networks as graphical models. Probabilistic

neural networks can be categorized according to their edge types: undirected [92] [93], directed [94] [95], and both [96].

In the undirected case, Ackley *et al.* [92] proposed Boltzmann machines and a learning algorithm to learn their edge weights. Each neuron in a Boltzmann machine was a binary random variable representing a hypothesis about data, and each edge represented a pairwise constraint between two hypotheses. The proposed learning algorithm was sequential and was derived under the assumption that random variables were distributed according to sigmoid functions, and a weight matrix in a Boltzmann machine was symmetric. Apolloni and Falco [97] extended Boltzmann machines [92] by proposing a parallel learning algorithm that allowed a weight matrix to be asymmetric. Although the limitation of a weight matrix was eliminated, the transition of any two configurations of a Boltzmann machine was assumed to follow a specific Markov transition matrix. Another assumption that is commonly used to simplify the learning algorithm is to restrict connections between neurons. Smolensky [93] proposed restricted Boltzmann machines (RBMs) that only allowed connections between layers but not between neurons in a layer. By limiting the connectivity, a parameter-learning algorithm was derived based on MCMC and Gibbs sampling. More information about different learning algorithms for RBMs can be found in [98].

In the directed case, Neal [99] [94] presented sigmoid and noisy-OR belief networks, and derived learning algorithms for them. A sigmoid belief network consisted of binary random variables that were connected by directed edges. Conditional probabilities were computed by sigmoid functions. A noisy-OR belief network was composed of binary neurons that were OR gates with preceding neurons as input. Conditional probabilities were predefined. Under these formulations, the negative phase, which was required in training Boltzmann machines, was not needed in learning algorithms. Bengio and Bengio [100] extended sigmoid belief networks [99] [94] by adding a hidden layer to capture high-level features. Based on the extended sigmoid belief networks, Larochelle and Murray [95] imposed restrictions on weight changes to speed up the process of learning weights.

In the case when probabilistic neural networks contain both undirected and directed edges, Hinton *et al.* [96] proposed deep belief networks that combined both a RBM [93]

and sigmoid belief networks [100]. A RBM was used as a complementary prior to make a data posterior distribution factorize. Thus, an explaining away effect was eliminated and edge weights could be learned one layer at a time.

Typically, hidden layers of neural networks are used to extract features from observations through layer-wise training [86] [87]. As hidden layers are closer to the output layer of a neural network, features become more abstract. The output layer then uses extracted features to estimate human poses. For example, Ouyang *et al.* [65] proposed to use neural networks to extract appearance, body-type and spatial features. Once the features were computed, human poses were estimated by a neural network with one hidden layer and linear activation functions at the output. Taylor *et al.* [47] proposed the implicit mixture of conditional RBMs for HPE. With the proposed model, the history of human poses could be utilized for estimating a human pose.

Among different structures of neural networks, convolutional neural networks have been used extensively for HPE. Toshev and Szegedy [63] formulated the HPE problem as a regression problem that mapped an image of a human to a normalized human pose. A convolutional neural network was applied on an image to estimate human-joint positions. Then, a convolutional neural network was built for each human joint and was applied on an image region centered at a human-joint estimate to refine the estimate. Chen and Yuille [59] used convolutional neural networks to both detect body parts and learn the spatial relationship between body parts. Weights corresponding to the appearance of body parts and the spatial relationship were learned using the structured SVM. Jain *et al.* [60] presented a framework that used convolutional neural networks to extract low-level features and generate a kinematic model that represented the constraints among body parts. Li *et al.* [61] [62] proposed a two-step approach based on convolutional neural networks. First, convolutional neural networks were used to directly estimate positions of human joints in an image. Then, body-part detectors, which were developed based on convolutional neural networks, were applied on the image at the estimated position to classify body parts. Pfister *et al.* [64] suggested that, based on convolutional neural networks, upper-body joints could be estimated without performing foreground segmentation. Temporal information was utilized by using

multiple frames as input to convolutional neural networks. Tompson *et al.* [66] used convolutional neural networks to build body-part detectors and a spatial model to represent the relationship between body parts. The detectors and spatial model were combined together as a neural network and trained using the backpropagation algorithm [6].

Hybrid Model

Hybrid models combine generative and discriminative models so that the combined models contain both properties of model generalization and fast computation.

Gall *et al.* [72] proposed a multi-layer framework of global stochastic optimization, filtering and local optimization for 3D HPE. In the first layer, interacting simulated annealing (ISA) was used to initialize the 3D human pose from images. In the second layer, jitter was filtered from the initial estimate with a short delay. The refined estimate was then used as a shape prior for the level-set segmentation. Sedai *et al.* [76] proposed a supervised particle filter method to track 2D human poses. The mapping between human poses and the corresponding histogram-of-shape-context (HoSC) descriptors [45] was learned using a mixture of regressors. Each regressor was modeled by RVM. A human pose in each frame was tracked by a particle filter which contained the mapping and the likelihood distribution derived based on silhouette and edges. Gupta *et al.* [69] proposed an observation-driven Gaussian process latent variable model (OD-GPLVM) to include an embedding from the observation space to the latent space of 3D human poses. It provided faster inference and a more accurate estimate compared to GPLVM. In addition, the OD-GPLVM could learn and estimate human poses from the scene context.

Prior knowledge can be used to reduce the dimensionality of the human-pose space. Gall *et al.* [75] incorporated the prior knowledge of gender and height for detecting body parts and human poses. Zuffi *et al.* [77] utilized the body shape by extending PSM to deformable structures (DS) which captured the shape of each body part. PCA was used to learn a low dimensional linear model of the shape which was represented by a linear Gaussian model. Sigal *et al.* [67] [68] adopted the kinematic relationship among body parts by

proposing a loose-limbed model which was a probabilistic graphical model. Nodes in the model represented body parts and edges represented the kinematic, penetration and temporal constraints. The constraints were represented by a mixture of Gaussian kernels. The model was first initialized by body-part detectors. Human pose was estimated and tracked using the Particle Message Passing method. Ganapathi *et al.* [70] adopted a sequence of monocular depth images and implemented a generative model for a 48 degree-of-freedom human model, together with a discriminative model for body-part detection on a GPU. A kinematic chain was formulated as a directed acyclic graph and the state transition was modeled by a dynamic Bayesian network.

Recently, human action [73] has been used as prior knowledge. Gall *et al.* [71] proposed a model for estimating the prior probability of an action from action classification to separate 3D-human poses into action-specific manifolds. Gaussian processes were trained to map between 3D-human-pose space and low-dimensional space. The 3D-human pose in a previous frame was used as an initialization for finding the optimal 3D-human pose and action jointly using a particle-based annealing optimization scheme. Yao *et al.* [73] proposed the appearance-based and pose-based features to classify actions using the Hough forest [101]. The appearance-based features included color, dense optical flow and spatio-temporal gradients. The pose-based features included the joint-distance feature, the (normal) plane feature and the (normal) velocity feature. Yao *et al.* [74] further extended the model of estimating the prior probability of action in [71] by incorporating the action classification [73] into a single framework. The dimensionality of human-pose space was reduced by considering the most probable action determined by the action classification.

Since, in general, a human pose can belong to more than one action, for example, the human pose of hand waving can be described from the action of standing and the action of raising both arms, we extend this concept by allowing more than one action to describe a human pose.

1.3 Contributions of the Thesis

This thesis focuses on estimating human poses of a person, using a stationary depth sensor. We explore methodologies for solving the HPE problem in two aspects: feature extraction and human-pose modeling.

In the feature-extraction aspect, our research effort centers on reducing feature ambiguity. We propose a 3D-point-cloud feature that is derived from an observation taken by a depth sensor. It is different from traditional work in the way of collecting and using sensory data. In traditional work, CCD cameras are used as sensors to capture a 3D scene into 2D color/grayscale images as observations. Since the 3D-to-2D projection does not preserve distances and angles, the ambiguity of depth information exists in features and thus affects the accuracy of human-pose estimates. On the contrary, the proposed 3D-point-cloud feature uses a depth sensor that captures depth information (3D point clouds) directly.

In the human-pose-modeling aspect, our research effort centers on two major approaches. We first propose an approach for reducing the dimensionality of human-pose space by using multiple human actions. The dimensionality reduction is based on our observation that human poses from a human action are varied in a specific pattern. Based on this observation, human-pose space can be decomposed into low-dimensional manifolds that may be overlapped with each other. Thus, instead of modeling human-pose space directly, our proposed approach first models each manifold individually. Manifolds are then combined together to create human-pose space.

In our second approach, we propose a human-pose model that is constructed based on a feedforward neural network. The proposed neural-network-based human-pose model takes the proposed 3D-point-cloud feature as input and estimates a human pose. It is considered as an extension to the first approach because the proposed neural-network-based human-pose model utilizes the same idea in the first approach, except that in the first approach, factors are designed manually and predefined in advance, and in the second approach, factors are learned automatically from training data. Based on the automatic design of factors,

the second approach can be adapted to various situations by learning factors from training data that are obtained from different environments.

To investigate the feature extraction and the human-pose modeling in the HPE problem, our research efforts have been directed towards the development of a 3D-point-cloud HPE system framework, which is outlined in Figure 1.2. Within this 3D-point-cloud HPE estimation framework, we shall focus on the following three major tasks:

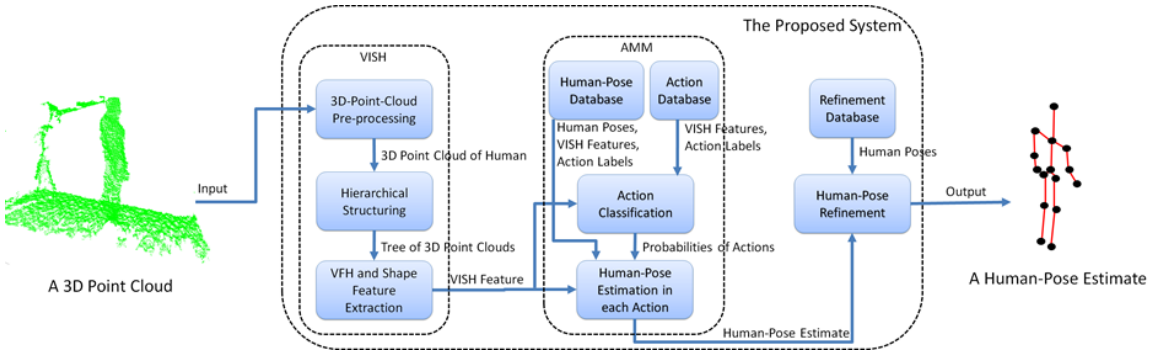


Fig. 1.2. Proposed 3D-point-cloud human-pose estimation system framework.

1. Derivation of the *viewpoint and shape feature histogram* (VISH) feature from a 3D point cloud. The proposed VISH feature captures spatial ordering of the global and local properties — orientation and shape — of 3D points from a human by extending the idea of histogram of oriented gradients (HOG). To handle the large number of 3D points in a 3D point cloud, the 3D points are summarized into a nonparametric distribution using viewpoint feature histogram (VFH) and shape features. The summarization is performed on overlapping 3D regions of the 3D point cloud so that the spatial ordering can be captured. The proposed VISH feature is then utilized in a later modeling process. Details about VISH feature can be found in Chapter 2.

2. Derivation of the non-parametric action-mixture model (AMM). As humans are highly articulated, modeling human-pose space directly is intractable. The proposed AMM lowers the dimensionality of human-pose space by incorporating the result from action classification to HPE. Using the prior knowledge of human actions, human poses

can be grouped according to their actions and modeled in low-dimensional manifolds. The classification result then determines the weighting coefficients in combing the low-dimensional manifolds. Details about AMM can be found in Chapter 3.

3. Extension of a 3D-point-cloud HPE system framework based on neural networks. We extend the 3D-point-cloud system framework by utilizing neural networks to estimate human poses. The extended system framework takes a 3D-point-cloud feature as input and estimates a human pose. Figure 1.3 shows the extended system framework. In the extension, instead of predefining AMM factors, we propose a mapping from AMM to a neural network so that its factors are designed automatically based on training data. We then use the concept of distributed representation to construct a scalable neural network for estimating human poses. Details about the extension can be found in Chapter 4.

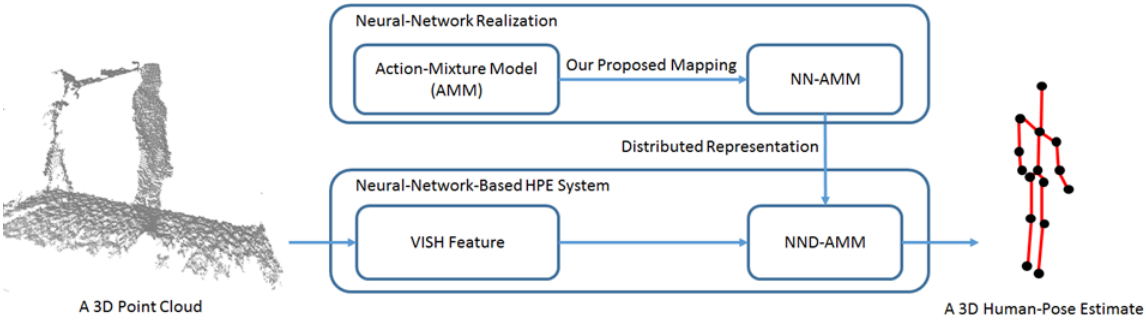


Fig. 1.3. An extension of the proposed HPE system based on neural-network realization.

The development of the 3D-point-cloud HPE system framework yields contributions in feature extraction and human-pose modeling. The contributions are:

1. Reduction of feature ambiguity by using the proposed 3D-point-cloud feature. By using a depth sensor to capture a 3D point cloud as an observation in HPE, we obtain the depth information of a person directly and thus reduce the ambiguity of depth information. Meanwhile, we introduce the hierarchical structuring of extracting features from a 3D point cloud to obtain coarse-level to fine-level information. Based

on the information, we can reduce the ambiguity of features extracted from observations.

2. Association of human actions into HPE to lower the dimensionality of human-pose space. In general, the search space grows exponentially with the dimensionality of human-pose space. Thus, searching human poses directly is intractable. We introduce the idea of using a low-dimensional manifold corresponding to an action in modeling human-pose space. We first propose a two-step approach (estimation and redistribution steps) in modeling the distribution of each manifold. Then, manifolds are combined according to a weighting function in which function values (weights) are computed by an action classifier.

3. Mapping the design problem of factors in AMM to the computational problem in neural networks. In general, factors in probabilistic models are manually designed and laborious. Yet, they may not represent underlying probability distributions. AMM suffers from the same issue. We realize the similarity between Bayesian networks and neural networks, and map the design problem in Bayesian networks to the computational problem in neural networks. This way, AMM can make use of the advantages of neural networks, namely learning capability, distributed representation, and adaptability, in the process of designing factors.

1.4 Organization of the Thesis

The organization of this thesis is described as follows. Chapter 2 describes the proposed 3D-point-cloud VISH feature that is extracted from a 3D point cloud captured by a stationary depth sensor. We shall first introduce the 3D-point-cloud pre-processing step that extracts 3D points corresponding to a human and removes 3D points from the environment. We then discuss the hierarchical-structuring step that partitions the pre-processed 3D point cloud and organizes the partitions in a tree structure. Features are extracted from each partition to form the proposed 3D-point-cloud feature. Finally, we shall compare our proposed

3D-point-cloud feature with some existing features, and describe computer-simulation results to illustrate the robustness of the proposed 3D-point-cloud feature.

Chapter 3 presents the proposed non-parametric action-mixture model (AMM) that utilizes the proposed 3D-point-cloud feature as input and estimates a human pose as output. We shall describe a mathematical formulation of incorporating multiple human actions in modeling human-pose space. An estimation step will be presented to individually model human-pose space for each human action. Since the same human pose may appear in multiple human actions, we shall consider this case by introducing a redistribution step. Based on the human-pose models, we shall merge them together by computing a weight for each human-pose model. A kinematic model will be presented to refine the human poses estimated from the AMM. Computer-simulation results will be described to illustrate the benefits of using multiple human actions in the modeling process.

Chapter 4 describes the extension of the proposed AMM based on neural networks. As we shall show the equivalence of a Bayesian network and a neural network under special situations, and test the adaptability of the extension, we shall begin with presenting two variants of the proposed 3D-point-cloud feature. Then, we shall describe the mathematical formulation of the proposed AMM as a Bayesian network, and present two important steps in mapping between a Bayesian network and a neural network: structure identification and parameter learning. After that, we shall discuss the realization of the proposed AMM by a neural network. Specifically, in addition to the two steps in the process of the realization, we shall describe a dictionary learning technique that yields a scalable realization. Finally, we shall present computer-simulation results to demonstrate the advantages of using neural networks.

In Chapter 5, we summarize the research results and explore future research direction.

1.5 Publications

Portions of this research have been submitted to and/or published in the following professional journal publications and/or professional conference proceedings:
Journal Publications:

1) Kai-Chi Chan, Cheng-Kok Koh and C.S. George Lee, ”A 3-D-Point-Cloud System for Human-Pose Estimation,” in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 11, pp. 1486–1497, Nov. 2014.

2) Kai-Chi Chan, Cheng-Kok Koh and C.S. George Lee, ”An Automatic Design of Factors in a Human-Pose-Estimation System using Neural Networks,” in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, to appear.

Conference Publications:

1) Kai-Chi Chan, Cheng-Kok Koh and C.S. George Lee, ”A 3D-Point-Cloud Feature for Human-Pose Estimation,” in *IEEE International Conference on Robotics and Automation*, pp. 1615-1620, May 2013.

2) Kai-Chi Chan, Cheng-Kok Koh and C.S. George Lee, ”Using Action Classification for Human-Pose Estimation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1176–1181, Nov. 2013.

3) Kai-Chi Chan, Cheng-Kok Koh and C.S. George Lee, ”Collaborative object tracking with motion similarity measure,” in *IEEE International Conference on Robotics and Biomimetics*, pp. 964–969, Dec. 2013.

4) Kai-Chi Chan, Cheng-Kok Koh and C.S. George Lee, ”Selecting Best Viewpoint for Human-Pose Estimation,” in *IEEE International Conference on Robotics and Automation*, pp. 4844-4849, May 2014.

5) Kai-Chi Chan, Cheng-Kok Koh and C.S. George Lee, ”Human-Pose Estimation with Neural-Network Realization,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4059–4064, Sep. 2015.

6) Kai-Chi Chan, Alper Ayvaci, and Bernd Heisele, ”Partially Occluded Object Detection by finding the Visible Features and Parts,” in *IEEE International Conference on Image Processing*, pp. 2130–2134, Sep. 2015. (Best Paper Award, 1st place)

2. 3D-POINT-CLOUD FEATURE

2.1 Introduction

Features play an important role in data processing because they can reduce undesired variations in observations, which are measured data, and facilitate the process of modeling. Features are used to represent observations. They can be binary, categorical or continuous. A robust feature should be informative and non-redundant. Sometimes, a feature also encapsulates semantic meaning. Deriving robust features has been an active research area mainly in computer vision and image processing. Although there are different kinds of features, each of them is essentially extracted by a process called feature extraction. Thus, deriving a feature can be defined as the task of deriving a feature-extraction process Λ, which is a function that maps an observation $o \in O$ to a feature vector $f \in F$. The set $O = \{o_1, \ldots, o_n\}$, where $n \in \mathbb{Z}^+$, represents a set of all observations (n observations) from which features are extracted. The set F represents a set of all possible features. Therefore, a feature-extraction process can be written as

$$\Lambda : O \mapsto F. \tag{2.1}$$

For example, when we use colors as features, we can define both observations and features as color images. The function Λ is an identity function. In extracting a common edge feature, namely a Sobel-edge feature, from a grayscale image, the feature-extraction process can be formulated in the form of Eq.(2.1) as follows. First, the set of all observations and the set of features should contain grayscale images, and possible edge magnitudes and orientations at each pixel, respectively. The function Λ should involve computations of

gradient magnitudes and orientations from two filtered images that are generated by convolving a grayscale image (an observation) with the following two filters

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

Historically, visual images (color/grayscale images) are common observations in markerless approaches for HPE. It is because many applications, such as surveillance systems and driver assistance systems, uses cameras as their input devices. At the same time, many techniques, such as human detection and human tracking, from research areas related to computer vision and image processing are traditionally built based on visual images. Thus, using visual images for HPE facilitates the integration of different techniques, and allows researchers to focus on HPE without putting much effort on other parts. It is therefore interesting to study the HPE problem using visual images as observations. One of the features that use visual images as observations is a HOG feature. It has shown to be useful in detecting humans. Being motivated by human detection, HOG features have been used to detect body parts, such as [65] [77], for HPE as well. Using the feature-extraction formulation in Eq.(2.1), we can describe HOG features by first defining the set of observations as a set of grayscale images and the set of features as a set of histograms of gradient directions for each small image region. The function Λ defines the computations of gradients, weighted votes of each region, and normalization of each region.

After the release of Microsoft Kinect, the development of depth sensors has been advanced quickly. As a result, researchers have begun to use depth images as observations. In a depth image, the intensity of each pixel represents the distance between a 3D point and a reference point such as the center of a depth sensor. 3D coordinates of a 3D point can be estimated based on the intensity of a pixel and intrinsic parameters of a depth sensor using a finite projective camera model [2]. Thus, instead of displaying an object in a 2D depth image, an object can be illustrated in 3D by plotting 3D points that are computed at depth pixels in 3D space. A collection of 3D points is called a 3D point cloud. Features that are

extracted from depth images are called 3D-point-cloud features. Depth images are popular because they can directly provide depth information that is lost in a visual image, which is a projection of 3D objects. 3D-point-cloud features can therefore capture geometric information. As depth information is important in HPE, we will focus on using depth images or 3D point clouds as observations and will investigate different types of 3D-point-cloud features in this chapter.

In the literature, different 3D-point-cloud features have been proposed based on different usages of depth information. A simple 3D-point-cloud feature can be a collection of 3D coordinates computed at each depth pixel that corresponds to a human. Elements (3D coordinates) in a simple 3D-point-cloud feature can be considered as equally important [50] or can have different weights [57] [56] that are learnt in a human-pose model. In addition, a 3D-point-cloud feature can be derived based on a comparison between each depth pixel with nearby pixels [51] [52]. The comparison allows a 3D-point-cloud feature to be depth-invariant; that is, it does not depend on the distance between a human and a depth sensor.

Among all the existing 3D-point-cloud features, two of them are easily accessible and therefore are used for evaluation in this chapter. They are all extracted from observations of 3D point clouds. The first one is called an *accumulative geodesic extrema* (AGEX) [48]. It is derived by exploiting a human-body property that geodesic distances on a human surface are largely invariant to human-body deformation. It is built by collecting a number of 3D points that are located at the longest distance from the centroid of a 3D point cloud of a human. Those points can be computed efficiently using the Dijkstra's algorithm with computational complexity $O(k \cdot (8n + n \log n))$, where k is the number of 3D points to be collected and n is the number of 3D points in a 3D point cloud of a human. As we can specify the number of 3D points to be collected, k can be considered as a constant. Thus, the computational complexity of extracting a AGEX feature becomes $O(n \log n)$. Although the feature-extraction process is efficient, the spatial ordering of collected 3D points is not preserved because 3D points may be collected in different orders from similar human poses.

Therefore, each 3D point is required to be classified to ensure a consistent ordering of 3D points.

The second 3D-point-cloud feature that is used in the experiment is a *viewpoint feature histogram* (VFH) [85]. It is mainly used in solving a more general problem of object recognition and pose (position and orientation) identification. A VFH feature is a 3D-point-cloud feature that collects a histogram of relative pan, tilt and yaw angles between a viewpoint direction and normals on a surface. The computational complexity is $O(n)$, where n is the number of 3D points in a 3D point cloud. Similar to visual features, histograms can summarize 3D points according to the shape of a human body, and can capture the curvature of a human-body surface. Although histograms can summarize characteristics in a 3D-point-cloud feature, details may be underestimated.

To highlight details, we hierarchically divide an observation, which is a 3D point cloud, of a human body into parts that are arranged in a tree structure. At the same time, we introduce the spatial ordering of parts to ensure that elements in a 3D-point-cloud feature are extracted from the same part of an observation of a human body. Previously, the concept of spatial ordering is used for 3D-point-cloud pre-processing [102] and 3D-surface reconstruction [103]. Based on this concept, we propose a 3D-point-cloud feature for HPE. We will show that the spatial ordering of parts is important in HPE and is useful to reduce the ambiguity of symmetric human poses.

To illustrate the performance of our proposed feature, we will use two common models, namely k-nearest neighbor algorithm (k-NN) [3] and support vector machine (SVM) [88], to map our proposed feature to human poses. When applying the two models, a set of training samples is first built. Each training sample contains a 3D-point-cloud feature and a ground-truth human pose that may be obtained from a marker-based approach or by manually marking on a depth image. To use k-NN, a 3D-point-cloud feature that is extracted from an observation is matched with all 3D-point-cloud features in a set of training samples. The closest k features in the set are retrieved and their corresponding human poses are averaged to form a human-pose estimate. To use SVM, a set of hyperplanes that separate each human pose with the largest margin is trained based on a set of training sample. A

human pose can then be estimated by finding a region in which a 3D-point-cloud feature lies.

In this chapter, we will describe our proposed 3D-point-cloud feature called *viewpoint and shape feature histogram* (VISH) [104]. It is inspired by the observation that a human-body shape is very distinctive and is evidential for different human poses. According to the formulation in Eq.(2.1), we can formulate our proposed VISH feature as follows. The set of observations is a set of 3D point clouds that contains a human and is captured by a static depth sensor. The set of features is a set of all possible VISH features. As we want to capture the shape of a human body, we build a VISH feature by aggregating an existing 3D-point-cloud feature, namely a VFH feature, and a shape feature that is computed based on depth values of 3D points (intensities of depth pixels). While extracting our proposed VISH feature, the function Λ is computed from both a 3D point cloud of a human and its decomposition. Thus, coarse-level information is extracted from a 3D point cloud of a human, and fine-level information is extracted from divisions of a 3D point cloud of a human. The novelty in our proposed VISH feature is the spatial ordering of VFH and shape features. The spatial ordering can capture both global and local properties, namely the orientation and shape, of 3D points from a human. These properties, when considered together, are important in resolving the ambiguity of symmetric human poses. The proposed VISH feature was tested on the Stanford TOF Motion Capture Dataset [70] using common human-pose models, namely k-NN and SVM. Two existing 3D-point-cloud features, namely VFH and AGEX features, were compared with our proposed VISH feature. Experimental results show that our proposed VISH feature can resolve the ambiguity of symmetric human poses. The proposed VISH feature can also reduce the overall error and standard deviation of human-pose estimates.

The structure of this chapter is as follows. Section 4.4 describes the proposed VISH feature, which represents the 3D point cloud of a human. Experimental results are discussed in Section 2.3. Conclusions are presented in Section 2.4.

2.2 Viewpoint and Shape Feature Histogram (VISH)

A VISH feature is defined as a representation of an observation of a 3D point cloud. In this section, we will describe the process of extracting a VISH feature from an observation of a 3D point cloud. As we will describe later, a VISH feature will be used to represent the distribution of 3D points of a human. There are three steps to extract a VISH feature from an observation of a 3D point cloud. The three steps are shown in Figure 2.1. They are 3D-point-cloud pre-processing, hierarchical structuring, and VFH and shape feature extraction. In the pre-processing step, 3D points corresponding to a human are extracted and 3D points corresponding to the background are removed to retain 3D points of interest. This step reduces the number of 3D points by keeping only relevant 3D points from a human for further processing. In the hierarchical structuring, The pre-processed 3D point cloud is represented as a node in a tree. The node is then partitioned and replicated into a tree structure. VFH and shape features are extracted from each node in a tree to provide a descriptor to represent each node. A VISH feature is formed by aggregating VFH and shape features from each node in a tree. We will explain each step in details in the following sections.

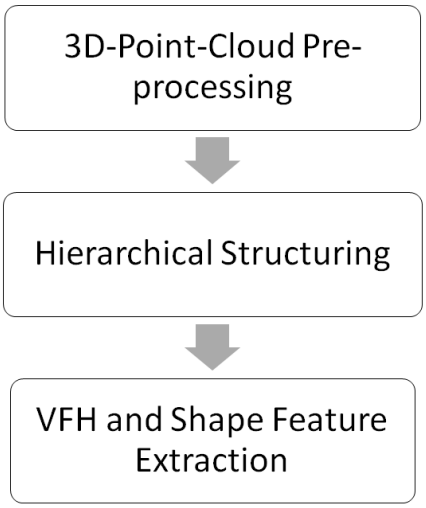


Fig. 2.1. Main components in the extraction of our proposed VISH feature.

2.2.1 3D-Point-Cloud Pre-processing

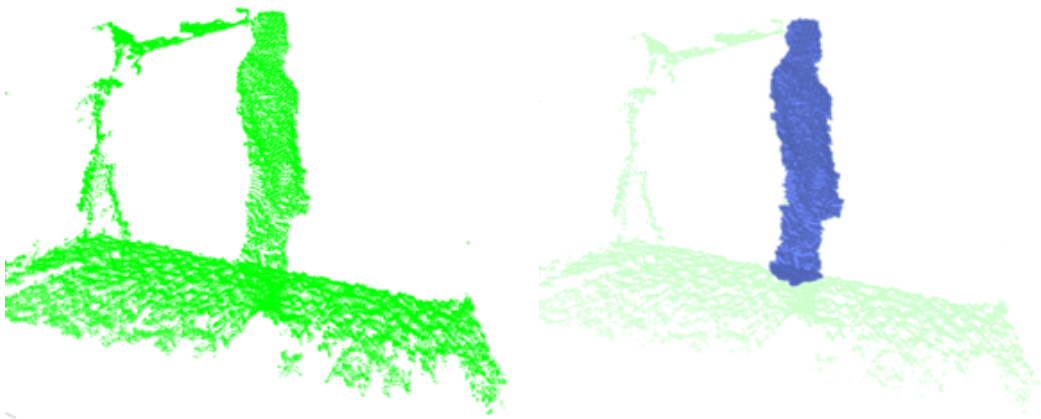
In this section, we simplify the process of human detection and segmentation in order to focus on HPE. We assume that a human, whose human pose is to be estimated, performs different actions in a predefined 3D region. Therefore, 3D points corresponding to a human can be extracted by filtering out 3D points outside a predefined 3D region. Let $\mathscr{P}$ be an observation of a 3D point cloud that is captured by a depth sensor. Let $\mathscr{R}$ be a set of all possible 3D coordinates (3D points) in a predefined 3D region. A set of 3D points corresponding to a human, $\mathscr{P}_A$, can be computed by intersecting a set $\mathscr{P}$ and a set $\mathscr{R}$. Mathematically, it is given by

$$\mathscr{P}_A = \mathscr{P} \bigcap \mathscr{R}, \tag{2.2}$$

where $\bigcap$ is the set intersection.

Figure 2.2 shows an example of using a predefined 3D region in the intersection method that extracts 3D points from a human. In Figure 2.2(a), it shows an observation of a 3D point cloud that is captured by a depth sensor. The observation consists of 3D points from the background and a human. For better illustration before and after the extraction process of 3D points from a human, 3D points captured by a depth sensor are depicted in green color in the figure. By retaining 3D points inside a predefined 3D region through the intersection method, 3D points from a human can be extracted. The extracted 3D points are depicted in blue color in Figure 2.2(b).

Because of the measurement noise from a depth sensor, there are outliers in a 3D point cloud $\mathscr{P}_A$. Smoothing or filtering techniques [105] are required to remove outliers by smoothing both spatial and temporal information of 3D points in a 3D point cloud $\mathscr{P}_A$. When using spatial information, we typically assume that the surface of a human body is smooth. Thus, any 3D point that is not on the surface will be considered as an outlier and will be removed. When using temporal information, we typically assume that a human is smoothly changing his/her postures. Thus, outliers can be detected by exploiting the consistency of 3D points over some consecutive 3D point clouds of a human. Since, in the experiment section, the two human-pose models, namely k-NN and SVM, for testing our



(a) (b)

Fig. 2.2. An extraction of 3D points from a person using a predefined 3D region. (a) An observation of a 3D point cloud. (b) The resulting point cloud as indicated by the blue color after filtering 3D points outside a predefined 3D region.

proposed VISH feature do not consider temporal information of a human, we apply only spatial smoothing to remove outliers in a 3D point cloud $\mathscr{P}_A$. We observe that outliers usually appear at random distances away from the surface of a human. Thus, we will remove outliers by comparing distances between neighboring 3D points in a 3D point cloud $\mathscr{P}_A$.

To detect outliers, we assume that the distance between two neighboring 3D points in a 3D point cloud $\mathscr{P}_A$ follows a continuous cumulative distribution function $F(\cdot)$. Any points that do not follow the distribution are considered as outliers. Under this assumption, outliers can be detected by the pseudo-residual method [106]. The formulation of the outlier detection is as follows.

Let D be a random variable representing the average distance between neighboring 3D points on the surface of a human. We use a random variable D as an input argument to a continuous cumulative distribution function $F(\cdot)$. The output of the function $F(D)$ is also a random variable. It can be shown that a random variable $F(D)$ follows a uniform distribution, denoted by U, from 0 to 1. Mathematically, the relationship between $F(D)$ and U is written as

$$F(D) \sim U(0,1). \tag{2.3}$$

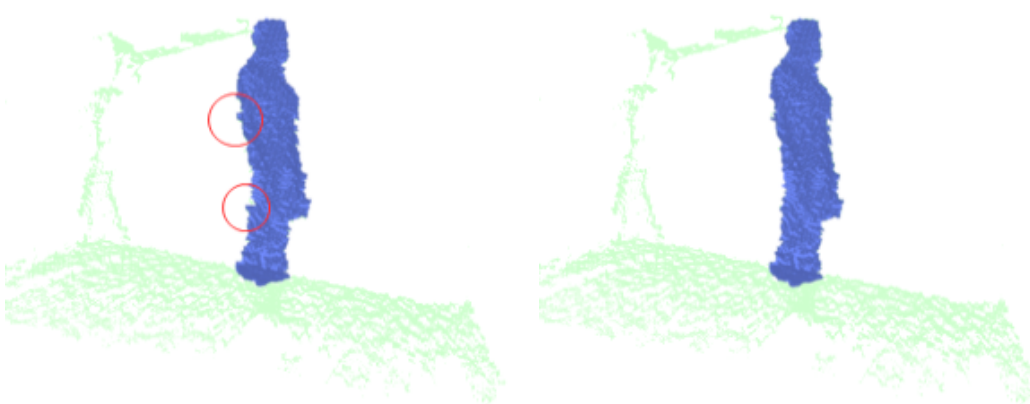
We can then define an auxiliary random variable, denoted as Z, for detecting outliers. It is defined as follows.

$$Z \triangleq \Phi^{-1}(F(D)), \tag{2.4}$$

where $\Phi^{-1}(\cdot)$ is the inverse of the cumulative distribution function of the standard normal distribution.

Thus, outliers in a 3D point cloud $\mathscr{P}_A$ can be removed by thresholding the deviation of the realization of an auxiliary random variable Z from the mean of the normal distribution. We will denote the resulting 3D point cloud after removing outliers by $\mathscr{P}_H$.

Figure 2.3 shows an example of the process of removing outliers. Before applying the pseudo-residual method, outliers are in a predefined 3D region because of the measurement noise from a depth sensor. In Figure 2.3(a), 3D points in a predefined region is highlighted in blue color. Outliers are indicated by red circles. After applying the proposed outlier-removal process, outliers are removed from the surface of a human in a predefined 3D region. The resulting 3D point cloud $\mathscr{P}_H$ after removing outliers is highlighted in blue color as shown in Figure 2.3(b).



(a) (b)

Fig. 2.3. An example of applying the pseudo-residual method to filter out outliers. (a) A 3D point cloud after removing 3D points outside a predefined 3D region. Outliers due to measurement noise are indicated by red circles. (b) The resulting 3D point cloud after filtering out outliers.

2.2.2 Hierarchical Structuring

In this section, we will create and utilize a spatial ordering in a 3D point cloud $\mathscr{P}_H$ that is generated by the previous step of 3D-point-cloud pre-processing. To create a spatial ordering, a tree is built to keep 3D regions as nodes in a consistent order. 3D regions are created based on a 3D point cloud $\mathscr{P}_H$. To build a tree, a 3D region that contains a 3D point cloud $\mathscr{P}_H$ of a human is first created. It is then divided into a number of 3D regions. All 3D regions are organized into a tree structure.

Specifically, each node in a tree represents a 3D region, which is defined as a set of all possible 3D points in a 3D region. An edge in a tree represents the process of duplicating 3D points from a node where an edge is connected to. When dividing 3D points in a node and building a tree, a 3D region is represented by a rectangular region (cuboid). While we use a rectangular region to represent a 3D region in this thesis, a 3D region can be generalized to and represented by other shapes such as spheres [107]. As we use the process above to build a tree, nodes at different levels of a tree capture a spatial ordering of sets of 3D points in 3D regions. At the zeroth level (root level) of a tree, the root node represents a 3D region containing a 3D point cloud $\mathscr{P}_H$. At other levels of a tree, nodes represent parts of the 3D region at a previous level. Mathematically, we formulate the whole process of building a tree structure as follows.

Let M^0 be a set of the smallest cuboid that contains 3D points in the root node at the zeroth level of a tree. In other words, M^0 contains the smallest cuboid that encapsulates all 3D points in a 3D point cloud $\mathscr{P}_H$. Let $S_n(\cdot)$ be a function that splits a cuboid into a set of exhaustive, continuous, mutually exclusive, and equal-sized cuboids, where $n \in \mathbb{Z}^+$ is the number of cuboids that are split. Let M^i be a set of cuboids that are returned by a function $S_{n_i}(\cdot)$ at the i-th level of a tree, where $n_i \in \mathbb{Z}^+$ is the number of cuboids that are returned by a function $S_{n_i}(\cdot)$ at the i-th level of a tree. We set n_0 to be 1 because, at the zeroth level, there is only one node (the root node) representing the smallest cuboid encapsulating a 3D point cloud $\mathscr{P}_H$. To indicate each element in a set M^i, we use $\mathscr{M}^i_j$ to denote the j-th element in

a set M^i. Based on the symbols defined above, we can derive a set M^i recursively by the following formula.

$$M^i = \bigcup_{j=1}^{|M^{i-1}|} S_{n_i}(\mathscr{M}_j^{i-1}), \quad \forall i = 1, 2, \ldots, h, \tag{2.5}$$

where $\bigcup$ is the set union, $|\cdot|$ is the cardinality of an input set, and h is the height of a tree with the convention that the height at the root level of a tree is 0.

In a tree, the j-th node at the i-th level is a cuboid and is denoted by $\mathscr{M}_j^i$, where $i = 0, \ldots, h$ and $j = 1, \ldots, |M^i|$. For each node $\mathscr{M}_j^i$, it contains a 3D region that is a set of all possible 3D points within that 3D region. 3D points in each node are then intersected with 3D points in a 3D point cloud $\mathscr{P}_H$ for feature extraction. The set intersection is equivalent to the extraction of 3D points from a 3D point cloud $\mathscr{P}_H$ in 3D regions specified by nodes in a tree.

Let $P_H(\cdot)$ be a function that performs the set intersection or the extraction process. It takes a cuboid (a node in a tree) as an input argument and outputs a set of 3D points of a 3D point cloud $\mathscr{P}_H$ in an input cuboid. To ensure a consistent ordering of sets of 3D points in 3D regions, we apply a function $P_H(\cdot)$ on each node of a tree in a breadth-first fashion. After traversing all nodes in a tree, we obtain a set of collections of 3D points that are originated from a 3D point cloud $\mathscr{P}_H$. We use W to denote the set. Mathematically, a set W is computed as follows.

$$W = \bigcup_{i=0}^{h} \bigcup_{j=1}^{|M^i|} \{P_H(\mathscr{M}_j^i)\}. \tag{2.6}$$

To illustrate the ideas that are introduced in this section, Figure 2.4 shows an example of the process of hierarchical structuring. The height of a tree is 1. The number of cuboids split, which is denoted as n_1, is 6. In the figure, the right side shows a 3D point cloud $\mathscr{P}_H$ that is generated by the process of 3D-point-cloud pre-processing. 3D points in the 3D point cloud $\mathscr{P}_H$ are highlighted in blue color. The smallest cuboid that contains the 3D point cloud $\mathscr{P}_H$ is depicted by thick red lines. A tree is built and is shown on the upper left region of the figure. The root node in the tree represents the smallest cuboid encapsulating

the 3D point cloud $\mathscr{P}_H$. M^0 is a set that contains the smallest cuboid. The cuboid that is represented by the root node is then divided into 6 sub-cuboids with equal volume by the function $S_6(\cdot)$. The six sub-cuboids are depicted by thin red lines on the right side of the figure. They are child nodes of the root node in the tree. Therefore, M^1 is a set that contains the six sub-cuboids. 3D points from the 3D point cloud $\mathscr{P}_H$ in each node of the tree are extracted and stored in W for feature extraction. In the example, W has seven elements. They are denoted as W_i, where $i = 1, \ldots, 7$. Each element in W is a set of 3D points from a part of the 3D point cloud $\mathscr{P}_H$. 3D points in each element are highlighted in blue color in the figure.

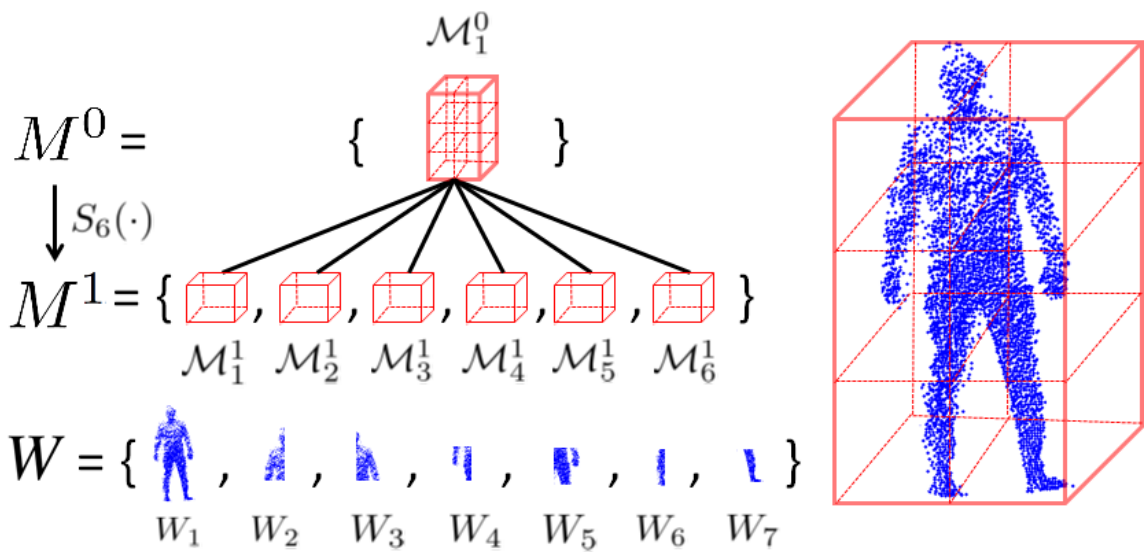


Fig. 2.4. An example of hierarchical structuring with the height of the tree is 1 and the number of cuboids split by $S_{n_1}(\cdot)$ is 6. The smallest cuboid, which contains the 3D point cloud $\mathscr{P}_H$ from the person, is shown on the right side. The cuboid is divided into six smaller sub-cuboids with equal volume by $S_{n_1}(\cdot)$. All the cuboids are arranged into a tree structure as shown on the left upper region. 3D points from the 3D point cloud $\mathscr{P}_H$ are extracted from each cuboid and grouped together in W for feature extraction.

2.2.3 Feature Extraction

In the previous step of hierarchical structuring, we create a set W, which contains collections of 3D points extracted from a 3D point cloud $\mathscr{P}_H$. As a VISH feature is formed

by aggregating two types of features in each element of a set W, in this section, we will describe the extraction process of two types of features.

The first type of features is a VFH feature. It is originally designed to estimate a six-degree-of-freedom pose (position and orientation) of rigid objects. It can describe the shape of rigid objects by capturing the curvature of their surfaces. As the shape of a human body is important in HPE, we will use it to describe the shape of the whole or part of a human body.

A VFH feature contains histograms of relative pan, tilt and yaw angles between the viewpoint direction of a depth sensor and normals on the surface of a human body. The process of extracting a VFH feature involves a computation of a surface normal at each 3D point. In each element of W, every 3D point should be assigned a direction that represents the direction of a surface normal. The direction of a 3D point is found by the following three steps. First, we compute the k nearest neighbors of a 3D point in the Euclidean space. Then, a 3D point together with the k nearest neighbors are assumed to be lying on a 2D plane. We find a 2D plane by minimizing the average perpendicular distance between a 2D plane, and a 3D point and the k nearest neighbors. After a 2D plane is found, a normal to a 2D plane is obtained. Finally, we set the direction of a normal of a 2D plane to point towards a depth sensor that is used to capture observations for HPE. The direction of a 3D point is set to be the direction of a normal. After defining a direction for each 3D point in an element of W, the relative pan, tilt and yaw angles between a 3D point and every 3D point of its k nearest neighbors can be computed.

Figure 2.5 shows an example of the computation of three angles. In the figure, the upper right region shows a 3D point, which is denoted by c, in red color and its nearest neighbors in orange color. After computing normals for all the 3D points (3D point c and its nearest neighbors), pairs are formed by selecting every 3D points in the nearest neighbors and a 3D point c. For each pair, a coordinate frame is assigned to the two 3D points in a pair. The lower left region of the figure shows a coordinate frame, which is indicated by red, green and blue arrows, is assigned to a pair of 3D points. Based on a coordinate frame, we can compute the three relative angles, which are defined as α, ϕ and θ in the figure.

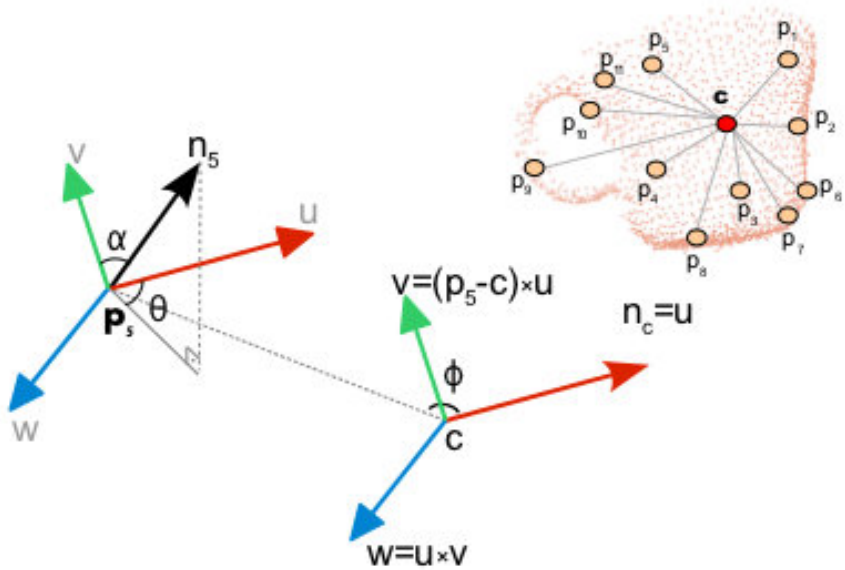


Fig. 2.5. Relative pan, tilt and yaw angles between two points in the extraction of a VFH feature [85].

The three angles of every pair of 3D points in an element of W are collected in a histogram with 308 bins, as suggested in [85], to form a VFH feature that represents that element. Figure 2.6 shows an example of a VFH feature. In the figure, the indices of bins are represented by the x-axis and the number of entries in bins are represented by the y-axis. A more detailed description of extracting a VFH feature can be found in [85].

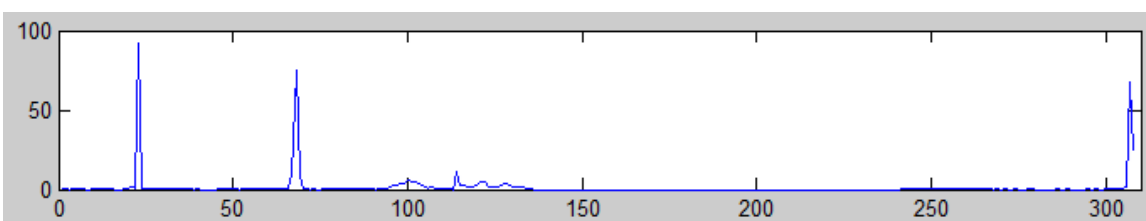


Fig. 2.6. An example of a VFH feature.

In addition to a VFH feature, we will extract a shape feature from each element of W. We will derive a shape feature based on pixel intensities of a depth image because pixel intensities have been shown to be important in HPE. As a VFH feature is a histogram-based feature, spatial information of 3D points in an element of W is lost. By using pixel

intensities in a shape feature, a shape feature can provide spatial information that is lost in a VFH feature.

Specifically, a shape feature is derived by measuring depth values of 3D points in an element of W around the 3D centroid of those 3D points. For notational brevity, our notations defined below will not distinguish elements of W. We will have notations that are the same for any one of the elements of W. The process of extracting a shape feature will be applied to each element of W. Let $\mathscr{X}$ be a set of 3D points in an element of W. Let $\mathbf{c}$ be a 3D centroid of $\mathscr{X}$. A 3D centroid can be computed as follows.

$$\mathbf{c} = \frac{1}{|\mathscr{X}|} \sum_{\mathbf{X} \in \mathscr{X}} \mathbf{X}, \tag{2.7}$$

where $|\mathscr{X}|$ is the number of 3D points in a set $\mathscr{X}$.

After a 3D centroid is computed, 3D points in a set $\mathscr{X}$ are projected on a 2D image, which is denoted by I. Note that an 2D image I is different from a depth image that is captured directly by a depth sensor. In an 2D image I, it only contains a projection of 3D points from a human. In a depth image captured by a depth sensor, it contains a projection of 3D points from both a human and the background. Thus, we need to compute a 2D image I.

To compute a 2D image I, we use a finite projective camera model [2] to represent a depth sensor. From a depth sensor, we measure its focal length, denoted as c_f, a skew parameter, denoted as s and a principal point, denoted as (p_x, p_y). All those data are stored in a camera calibration matrix. Let K be a camera calibration matrix. It can be written as follows.

$$K = \begin{pmatrix} c_f m_x & s & m_x p_x \\ 0 & c_f m_y & m_y p_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.8}$$

where m_x and m_y are the numbers of pixels per unit distance in image coordinates along the horizontal and vertical directions respectively.

The orientation and the center of the camera coordinate frame of a depth sensor can be represented by a 3×3 rotation matrix and a 3×1 position vector, respectively, in the world coordinate frame. Then, the mapping between a 3D point in a set $\mathscr{X}$ and an image point in an 2D image I can be represented as follows.

$$\mathbf{x} = KR \begin{pmatrix} 1 & 0 & 0 & -x_c \\ 0 & 1 & 0 & -y_c \\ 0 & 0 & 1 & -z_c \end{pmatrix} \mathbf{X}, \tag{2.9}$$

where $\mathbf{x}$ is the homogeneous coordinates of an image point in the camera coordinate frame of an 2D image I and $\mathbf{X}$ is the homogeneous coordinates of a 3D point in a set $\mathscr{X}$, K is a camera calibration matrix, R is a 3×3 rotation matrix, and (x_c, y_c, z_c) are the coordinates of the center of a depth sensor.

Based on Eq.(2.9), a 3D point in a set $\mathscr{X}$ can be mapped to a pixel in a 2D image I. In a 2D image I, some pixels may or may not be projections of 3D points in a set $\mathscr{X}$. If a pixel is not a projection of a 3D point, its intensity is zero. Some pixels may be a projection of one 3D point in a set $\mathscr{X}$. In this case, each pixel should have an intensity equal to the distance between a 3D point that is projected to that pixel and a depth sensor. Some pixels may be a projection of more than one 3D points in a set $\mathscr{X}$. In this case, each pixel should have an intensity equal to the shortest distance from any one of those 3D points to a depth sensor.

We define a position of a pixel in a 2D image I as follows. The position of the upper left pixel in a 2D image I is (0,0). The x coordinate of a pixel position increases towards the right side of an image. The y coordinate of a pixel position increases towards the bottom of an image. Let $I(u,v)$ be the intensity of a pixel in a 2D image I at a position (u,v). Let (u^*, v^*) be the position of a pixel in a 2D image I where a 3D centroid $\mathbf{c}$ is projected to. A window with a size $(2w+1) \times (2w+1)$ pixels is centered at the position (u^*, v^*) on an 2D image I to extract a shape feature, where w is a user-defined parameter to control the size of a window. Intensities of pixels within a window are grouped together to form a shape

feature. Concretely, let $\mathbf{f}_s(\cdot)$ be a mapping that takes a set of points $\mathscr{X}$ as input and output a shape feature. A shape feature $\mathbf{f}_s(\mathscr{X})$ can be expressed as

$$\mathbf{f}_s(\mathscr{X}) = (I(u^* - w, v^* - w), I(u^* - w, v^* - w + 1),$$

$$\ldots, I(u^* - w + 1, v^* - w), \ldots, I(u^* + w, v^* + w)). \tag{2.10}$$

Figure 2.7 shows an example of a shape feature when the parameter w is set to 8. Under this parameter setting, the size of a window and hence the size of a shape feature are $(2 \times 8 + 1)^2 = 289$ pixels. The actual shape feature is a vector with length equal to 289. In the figure, it is depicted as a 2D image for illustration purposes. The color at each pixel represents the distance between a 3D point that is projected to that pixel and a depth sensor that captures a human.

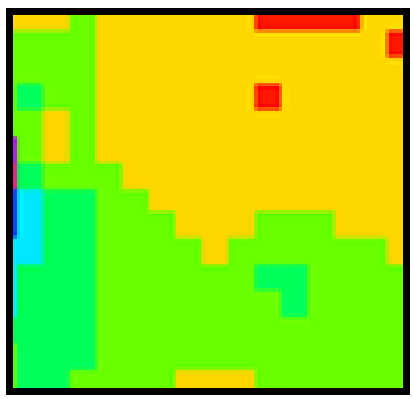


Fig. 2.7. An example of a shape feature.

To form a VISH feature, both VFH and shape features are extracted from every element of W, and are arranged in a consistent order in the form of a row vector. Let $\mathbf{f}(\cdot)$ be a mapping from a 3D point cloud $\mathscr{P}_H$ to a row vector that contains VFH and shape features. We use $\mathbf{F}$ to denote a VISH feature. A VISH feature $\mathbf{F}$ is considered as an aggregation of VFH and shape features that are extracted from all the elements of W. Based on the symbols above, it can be expressed as follows.

$$\mathbf{F} = (\mathbf{f}(W_1), \mathbf{f}(W_2), \ldots, \mathbf{f}(W_{|W|})), \tag{2.11}$$

where W_i is the i-th element in W, and $|W|$ is the size/cardinality of W.

Note that in a tree structure, 3D points in a child node are duplicated from its parent node. It provides an illusion that 3D points are redundant among different levels of a tree. However, 3D points at different levels provide different VFH and shape features. When VFH and shape features are used to summarize 3D points in child nodes at different levels. they describe different levels of details in the summarization. In addition, the ordering of elements in a set W provides useful information of spatial ordering of a set of 3D points from a human.

We analyze the time complexity of extracting a VISH feature as follows. The time complexity of extracting a VISH feature depends on the time complexity of extracting VFH and shape features. When deriving a VFH feature, the main computation is the computation of a histogram of relative pan, tilt and yaw angles. It involves the comparison of each 3D point with its k nearest neighbors. Thus, the time complexity of extracting a VFH feature is $O(kn)$, where k is the number of nearest neighbors and n is the number of 3D points in an element of W. As k is typically small compared to n, the time complexity can be rewritten as $O(n)$.

When deriving a shape feature, it involves two main computations that depend on the number of 3D points in an element of W. The first one is the computation of a 3D centroid from a set of 3D points in an element of W. It has a time complexity of $O(n)$, where n is the number of 3D points in an element of W. The second one is the computation of projecting 3D points from an element of W into a 2D image I. It has a time complexity of $O(n)$. Thus, the time complexity of extracting a shape feature from an element of W is $O(n)$.

When we analyze the time complexity of extracting a VISH feature, we need to consider every 3D point in nodes of a tree. In a tree, the root node contains 3D points of a 3D point cloud $\mathscr{P}_H$. As the function $S_n(\cdot)$ splits a cuboid into a set of exhaustive, continuous, mutually exclusive, and equal-sized cuboids, all nodes at the same level of a tree contain all 3D points of a 3D point cloud $\mathscr{P}_H$. Thus, each 3D point of a 3D point cloud $\mathscr{P}_H$ appears exactly $(h+1)$ times in a tree, where h is the height of a tree. The time complexity of extracting a VISH feature is therefore $O((h+1)n)$, where n is the number of 3D points in a

3D point cloud $\mathscr{P}_H$. Experimental results show that the height h of a tree is typically small compared to the number of 3D points n. Thus, the time complexity of extracting a VISH feature can be rewritten as $O(n)$. Compared with the two 3D-point-cloud features, namely VFH and AGEX features that will be analyzed in the experiment, extracting a VISH feature is asymptotically the same as extracting a VFH feature, and is asymptotically more efficient than extracting a AGEX feature, which has a time complexity of $O(n \log n)$ [48].

Our proposed VISH feature provides a representation of a 3D point cloud of a human in a predefined region. It captures global and local properties of a 3D point cloud in a representation. Through a human-pose model, a VISH feature can be used to estimate human poses. In the experiment, we will use two common models, namely k-NN and SVM, as our human-pose models to evaluate our proposed VISH feature and compare its performance with the other two 3D-point-cloud features. As we will show in the experiment, our proposed VISH feature can reduce the ambiguity of symmetric human poses (see Figures 2.9 and 2.10) and is more robust than the other two 3D-point-cloud features. However, the dimensionality of human-pose space in k-NN and SVM is high and hence the human-pose models k-NN and SVM do not perform well. Thus, we will propose a non-parametric action-mixture model (AMM) in the next chapter to further increase the accuracy of HPE by lowering the dimensionality of human-pose space.

2.3 Experimental Results

The proposed 3D-point-cloud feature was implemented using the point cloud library (PCL) [108] and tested on the Stanford TOF Motion Capture Dataset [70], which contains 28 video sequences. Each video sequence corresponds to one action. A human pose of a subject has 15 degrees-of-freedom (joints), namely head, neck, left/right shoulder, left/right elbow, left/right wrist, hip center, left/right hip, left/right knee and left/right ankle. Figure 2.8 shows a human pose in the dataset.

The number of frames of video sequences in the dataset is shown in Table 2.1. In the dataset, a subject performed different actions such as kicking and rotation, and was

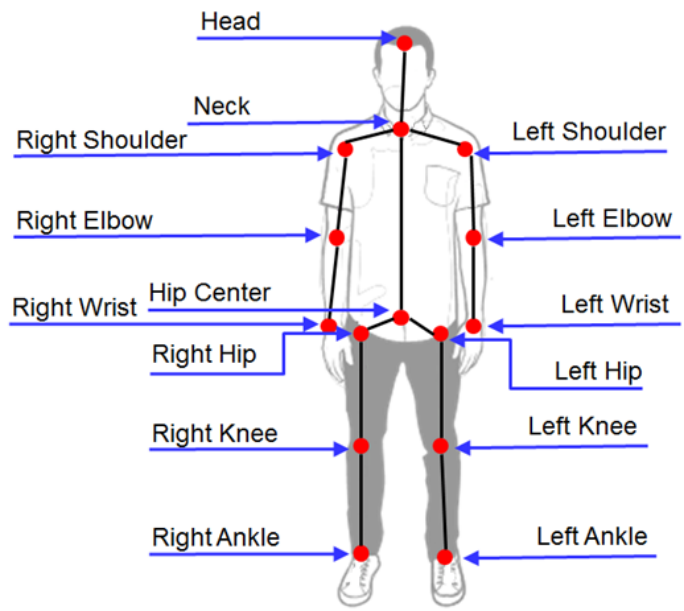


Fig. 2.8. A human pose of a subject in the dataset.

captured by a Swissranger SR4000 TOF sensor. Depth images were captured at 25 frames per second at a resolution of 176 $\times$ 144 pixels. The ground-truth 3D joint locations of the subject were recorded by a commercial motion-capturing system. When evaluating the performance of HPE, frames with missing ground-truth 3D joint locations were ignored. The error metric, ζ, for each video sequence was defined as

$$\zeta = \frac{1}{N_f}\sum_{s=1}^{N_f}\frac{1}{N_s}\sum_{i=1}^{N_s}\left\|\mathbf{j}_{s,i}-\tilde{\mathbf{j}}_{s,i}\right\|_2, \tag{2.12}$$

where N_f is the number of frames of a video sequence for testing, N_s is the number of 3D joint locations measured by the motion-capturing system in the s-th frame, $\mathbf{j}_{s,i}$ is the ground-truth 3D location of the i-th joint in the s-th frame,$\tilde{\mathbf{j}}_{s,i}$ is the estimated 3D location of the i-th joint in the s-th frame and $\|\cdot\|_2$ is the Euclidean norm.

Two existing 3D-point-cloud features, namely VFH and AGEX, were implemented for comparison. Two common models, namely k-NN and SVM, were used to estimate human poses based on the three 3D-point-cloud features. The 3D-point-cloud features were evaluated using 5-fold cross-validation.

Table 2.1 Total number of frames in each video sequence from the dataset.

Number of frames	Video index
100	0-5,8,9,14,16,18,
400	6,7,10-13,15,17,19,20-27

In the pre-processing step, the predefined 3D region was set to be the 3D space with the depth ranging from -3 meters to -2 meters. 50 closest 3D points were used to estimate the average distance of each 3D point. The distribution function $F(\cdot)$ of the average distance was set to be a normal distribution. 3D points were considered as outliers if their deviations from the mean of the normal distribution were larger than one standard deviation.

Two hierarchical levels were used in the tree structure. Six cuboids were split from the smallest cuboid containing the root node. To estimate the direction of a 3D point, 3D points within 0.01 meters from that 3D point were used. The size of the 2D region used for extracting a shape feature was set to be 17×17 pixels; that is, w was set to 8 in Eq.(2.10).

2.3.1 Comparison between VFH and VISH features

As our proposed VISH feature is derived from a VFH feature, we first compare the performance of VFH and VISH features qualitatively. For comparison purposes, k-NN was used as a human-pose model to map the two features to human poses based on the same observations.

Figures 2.9 and 2.10 show the estimation results generated by k-NN using VFH and VISH features, respectively. In both figures, the estimation results are shown in the form of 2D images of silhouettes. Those 2D images were created by first projecting 3D point clouds of a subject performing different actions onto 2D image planes. Then, silhouettes were detected in 2D images by thresholding the intensity of each pixel. The bottom left images in the two figures show an observation of a 3D point cloud from a subject. The subject was raising his/her left arm in an observation. Other images in both figures were matches that are returned by k-NN. They were ranked from left to right, and bottom to top. For example, the closest match was at the second column in the last row. The Eu-

clidean distance between the VFH/VISH feature corresponding to the closest match and the VFH/VISH feature corresponding to the image at the bottom left is the smallest. In Figure 2.9, some matches were raising the other arm. It showed that VFH features gave similar descriptions among symmetric human poses. Therefore, VFH features could not distinguish those symmetric human poses. In Figure 2.10, all matches were raising the left arm. Thus, the ambiguity of symmetric human poses was greatly reduced when an observation of the subject was represented by a VISH feature.

The main reason of the difference between the two estimation results was because VISH features could capture the spatial ordering of 3D point clouds in an observation and VFH features could not. When a VISH feature was used, both VFH and shape features in a VISH feature were extracted from 3D point clouds in a tree. Those 3D point clouds represented the whole body and body parts of a subject. However, when only a VFH feature was used to represent an observation, it was extracted from a 3D point cloud of the whole body of a subject directly. As a VFH feature was derived based on histograms of relative pan, tilt and yaw angles, extracting features from a 3D point cloud of the whole body directly could only capture global properties of a 3D point cloud. Any local properties (fine details) were suppressed. On the other hand, a VISH feature could capture both global and local properties by partitioning a 3D point cloud of the whole body into body parts in a consistent order.

2.3.2 Evaluation of our proposed VISH feature using k-NN

The goal of the experiment in this section was to evaluate the accuracy of HPE in a human-pose model when observations (3D point clouds) were represented by the three 3D-point-cloud features: VISH, VFH and AGEX features. We used k-NN as a human-pose model in this experiment. 80% of the dataset was used as training data and 20% of the dataset was used as testing data. Training samples in the training data were used to learn the mapping from a 3D-point-cloud feature to a human pose from the training data. When

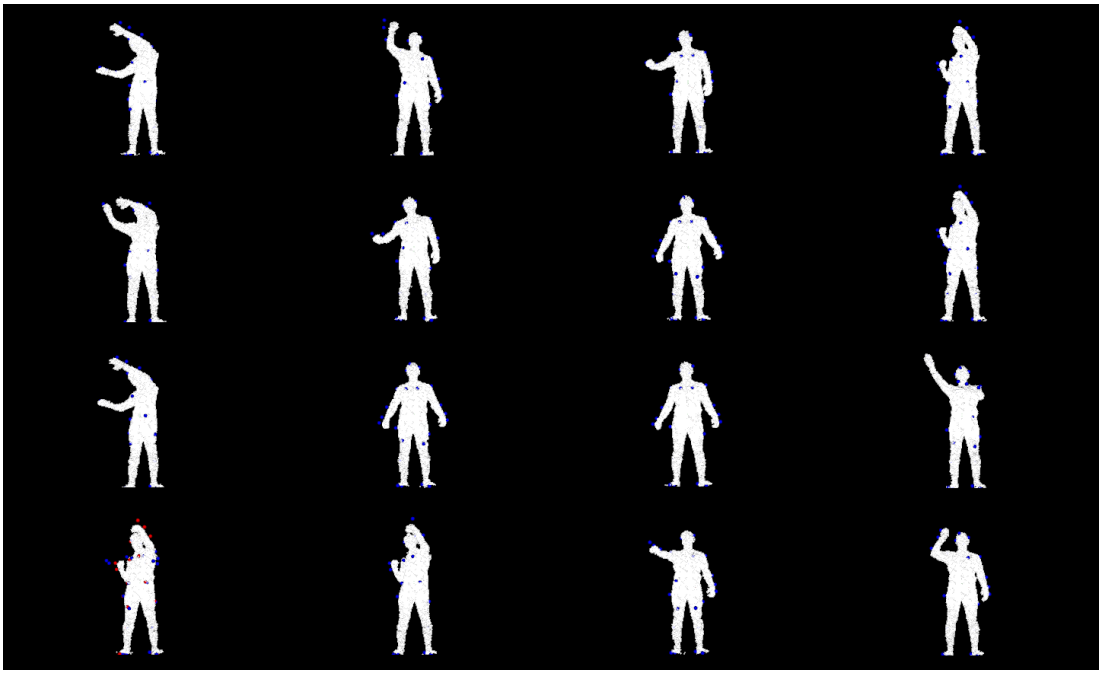


Fig. 2.9. Ambiguity of symmetric human poses represented by VFH features exists in some close matches using k-NN. The 3D point cloud at the bottom left is a query pose. Others are the 3D point clouds returned by k-NN. The returned 3D point clouds are ranked from left to right, and bottom to top.

observations were represented by the three 3D-point-cloud features, k-nearest human poses that were computed by k-NN were averaged to give the final estimates of human poses.

Table 2.2 shows the quantitative results when observations were separately represented by the three 3D-point-cloud features. The results were computed by k-NN when $k = 3$. A bar chart of errors of human-pose estimates incurred in the three 3D-point-cloud features is shown in Figure 2.11. The three bars (in blue, red, and green colors) in the figure corresponded to HPE errors when observations were represented by our proposed VISH features, VFH features, and AGEX features, respectively. When comparing the three bars, the blue bar was significantly lower than the other two bars. It showed that the error incurred in using our proposed VISH feature was the lowest compared with VFH and AGEX features. The results showed the importance of the property of spatial ordering in our proposed VISH feature. The error incurred in using a VFH feature was higher than our proposed VISH feature because a VFH feature suffered from the limitation of describing local properties of

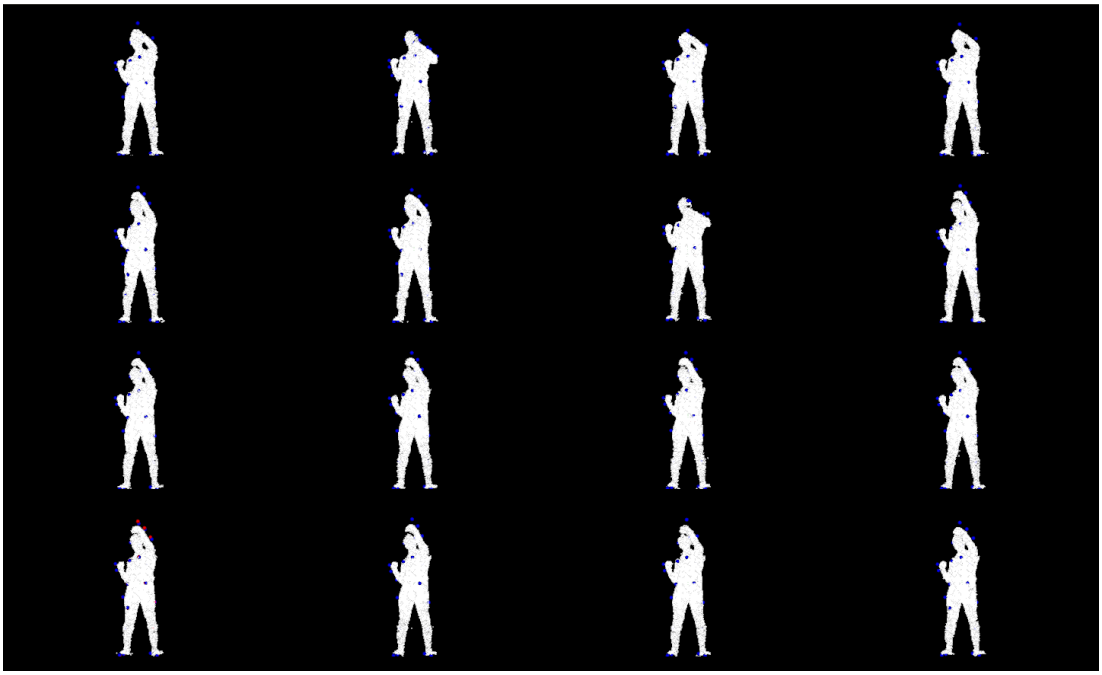


Fig. 2.10. Ambiguity of symmetric human poses is greatly reduced when observations are represented by VISH features.

an observation of a 3D point cloud. The error incurred in using an AGEX feature was the highest among the three 3D-point-cloud features. It was because when the limbs of a subject were moved, geodesic extrema of the limbs in AGEX features were switched or lost. The spatial ordering of geodesic extrema was not consistent. Thus, using AGEX features was not able to be used to look up human poses from the training data.

From Table 2.2, the overall error and standard deviation of the HPE error incurred in using our proposed VISH feature were 0.017m and 0.012m, respectively. Compared with VFH and AGEX features, their overall errors were about 2 times and 4.5 times as much as the overall error incurred in using our proposed VISH feature, respectively. The standard deviations of using VFH and AGEX features were about 2 times and 2.7 times as much as the standard deviation incurred in using our proposed VISH feature, respectively. It showed quantitatively that our proposed VISH feature was more robust in representing observations of 3D point clouds of a subject. It was more discriminative than the other two 3D-point-cloud features.

Table 2.2 A quantitative result of feature evaluation using k-NN when k=3. Numbers on the left and inside the parentheses are errors and standard deviations (in meters), respectively.

Test case	VISH	VFH	AGEX
0	0.015 (0.008)	0.030 (0.021)	0.042 (0.023)
1	0.016 (0.012)	0.039 (0.027)	0.073 (0.037)
2	0.013 (0.008)	0.028 (0.019)	0.053 (0.030)
3	0.021 (0.012)	0.026 (0.013)	0.084 (0.051)
4	0.015 (0.008)	0.032 (0.014)	0.047 (0.024)
5	0.014 (0.009)	0.039 (0.023)	0.051 (0.029)
6	0.012 (0.008)	0.038 (0.021)	0.055 (0.029)
7	0.016 (0.012)	0.033 (0.022)	0.054 (0.027)
8	0.012 (0.010)	0.038 (0.025)	0.055 (0.026)
9	0.015 (0.008)	0.040 (0.028)	0.065 (0.036)
10	0.019 (0.010)	0.033 (0.023)	0.078 (0.059)
11	0.013 (0.007)	0.024 (0.016)	0.056 (0.030)
12	0.012 (0.008)	0.025 (0.019)	0.077 (0.049)
13	0.012 (0.010)	0.026 (0.026)	0.068 (0.038)
14	0.020 (0.016)	0.025 (0.019)	0.085 (0.057)
15	0.014 (0.008)	0.034 (0.020)	0.061 (0.031)
16	0.017 (0.012)	0.065 (0.042)	0.071 (0.034)
17	0.023 (0.015)	0.058 (0.034)	0.086 (0.043)
18	0.015 (0.011)	0.034 (0.027)	0.079 (0.030)
19	0.016 (0.011)	0.042 (0.030)	0.071 (0.048)
20	0.020 (0.013)	0.040 (0.025)	0.082 (0.058)
21	0.022 (0.014)	0.043 (0.023)	0.085 (0.045)
22	0.014 (0.009)	0.042 (0.022)	0.073 (0.046)
23	0.015 (0.011)	0.048 (0.043)	0.089 (0.069)
24	0.021 (0.022)	0.049 (0.059)	0.141 (0.121)
25	0.020 (0.020)	0.032 (0.025)	0.096 (0.070)
26	0.018 (0.012)	0.030 (0.030)	0.097 (0.058)
27	0.023 (0.019)	0.060 (0.057)	0.177 (0.090)
Overall	**0.017 (0.012)**	**0.038 (0.027)**	**0.077 (0.046)**

Figure 2.12 shows the overall error of human-pose estimates incurred in using the three 3D-point-cloud features when the value of k in k-NN was changed to 1, 2, 4, 8, and 16. When the value of k was changed, the overall error incurred in using our proposed VISH feature was the lowest among the three 3D-point-cloud features. The overall error incurred in using a AGEX feature was the highest. The trend of the change of the overall error in-

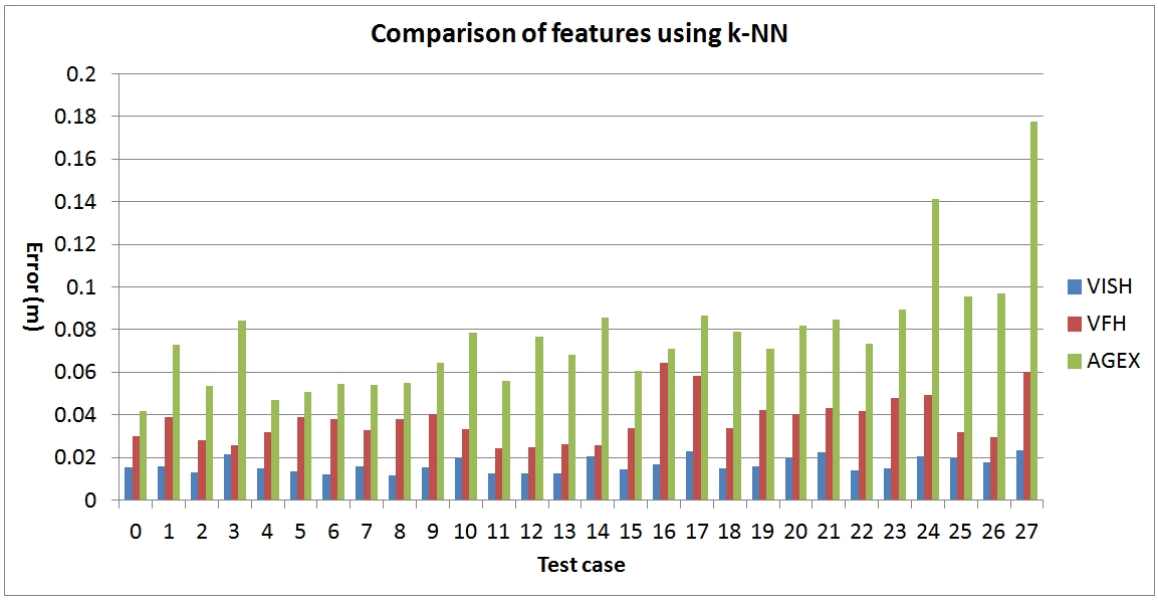


Fig. 2.11. Evaluation of VISH, VFH and AGEX features using k-NN (when k=3).

curred in using the three 3D-point-cloud features was the same. When the value of k started to increase from one, the overall error incurred in using the three 3D-point-cloud features was decreased because the measurement noise from a depth sensor in the 3D point cloud $\mathscr{P}_H$ was averaged out. As the value of k increased, the overall error was first decreased, but started to increase because details of a 3D point cloud $\mathscr{P}_H$ were also averaged out when k was too big. Overall, our proposed VISH feature was robust across a wide range of k in k-NN.

2.3.3 Evaluation of our proposed VISH feature using SVM

In this experiment, our goal was to evaluate the accuracy of human poses that were estimated by another common model, namely SVM, when observations were represented by the three 3D-point-cloud features: VISH, VFH, and AGEX features. 80% of the dataset was used as training data and 20% of the dataset was used as testing data. As SVM was a classification method, we created class labels for each training sample in the training data. Each class label was represented by a human-pose prototype that was the same for all

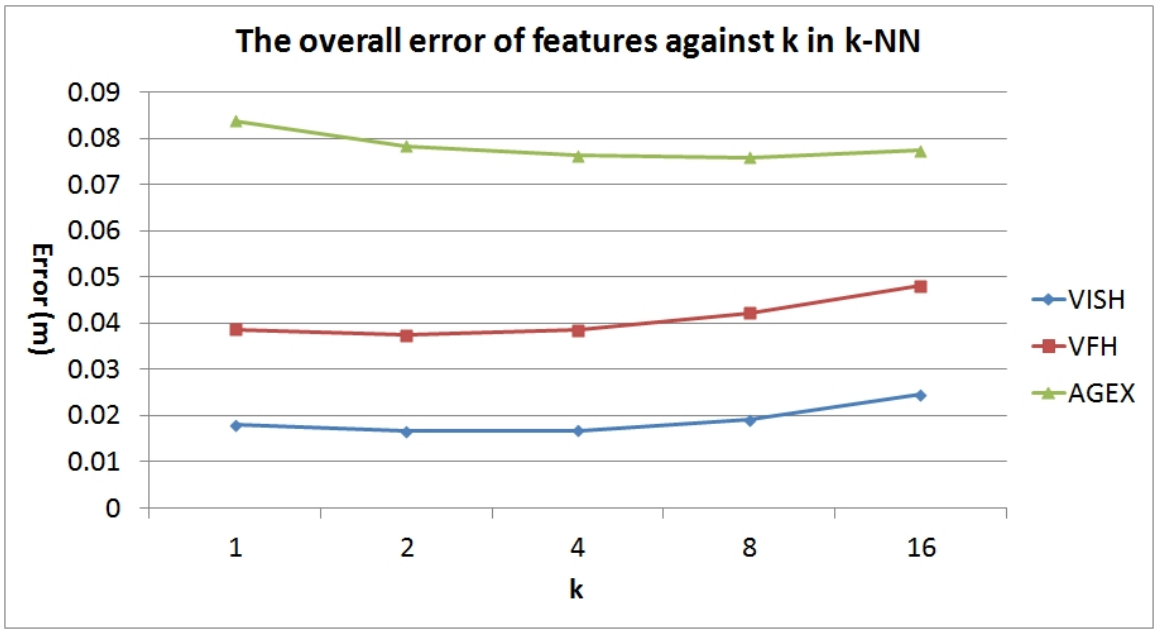


Fig. 2.12. Comparison of VISH, VFH and AGEX features using k-NN under different values of k.

members (training samples) belonging to that class. We used the k-means algorithm [3] to find 1500 class labels (human-pose prototypes) in the training data.

Table 2.3 shows the quantitative results when observations were separately represented by the three 3D-point-cloud features. A bar chart of errors of human-pose estimates incurred in the three 3D-point-cloud features is shown in Figure 2.13. The three bars (in blue, red, and green colors) in the figure corresponded to HPE errors when observations were represented by our proposed VISH features, VFH features, and AGEX features, respectively. By comparing the three bars, the error incurred in using our proposed VISH feature was the lowest among the three 3D-point-cloud features. It was consistent with the previous results when k-NN was used in the evaluation of 3D-point-cloud features.

From Table 2.3, the overall errors incurred in using VFH and AGEX features were about 1.5 times and 2.3 times as much as the overall error incurred in using our proposed VISH feature, respectively. The standard deviations incurred in using VFH and AGEX features were about the same as the standard deviation incurred in using our proposed VISH feature. From the results, it assured that our proposed VISH feature performed better than the other

two 3D-point-cloud features. Note that the overall error and standard deviation of human poses estimated by SVM were higher than the error of human poses estimated by k-NN. The main reason was that in the process of creating class labels in SVM, a quantization error was incurred in assigning a human-pose prototype to each training sample.

Table 2.3 A quantitative result of feature evaluation using SVM. Numbers on the left and in the parentheses are errors and standard deviations (in meters), respectively.

Test case	VISH	VFH	AGEX
0	0.036 (0.043)	0.038 (0.051)	0.069 (0.053)
1	0.072 (0.076)	0.126 (0.091)	0.122 (0.083)
2	0.031 (0.047)	0.033 (0.055)	0.074 (0.044)
3	0.071 (0.072)	0.077 (0.080)	0.101 (0.053)
4	0.021 (0.035)	0.064 (0.045)	0.103 (0.047)
5	0.039 (0.044)	0.065 (0.061)	0.075 (0.069)
6	0.040 (0.060)	0.065 (0.070)	0.101 (0.067)
7	0.034 (0.063)	0.047 (0.064)	0.133 (0.065)
8	0.055 (0.059)	0.104 (0.085)	0.164 (0.056)
9	0.019 (0.034)	0.040 (0.062)	0.123 (0.051)
10	0.059 (0.075)	0.075 (0.078)	0.138 (0.066)
11	0.015 (0.035)	0.022 (0.049)	0.164 (0.051)
12	0.030 (0.047)	0.056 (0.069)	0.146 (0.067)
13	0.030 (0.050)	0.030 (0.052)	0.099 (0.040)
14	0.048 (0.055)	0.054 (0.070)	0.103 (0.050)
15	0.027 (0.043)	0.056 (0.063)	0.144 (0.069)
16	0.023 (0.043)	0.082 (0.064)	0.084 (0.061)
17	0.079 (0.085)	0.128 (0.098)	0.202 (0.085)
18	0.056 (0.068)	0.071 (0.065)	0.077 (0.055)
19	0.051 (0.076)	0.085 (0.088)	0.151 (0.093)
20	0.071 (0.077)	0.114 (0.089)	0.140 (0.074)
21	0.101 (0.097)	0.130 (0.094)	0.173 (0.076)
22	0.064 (0.094)	0.136 (0.109)	0.194 (0.067)
23	0.071 (0.094)	0.133 (0.117)	0.143 (0.082)
24	0.138 (0.158)	0.173 (0.185)	0.204 (0.138)
25	0.098 (0.107)	0.106 (0.102)	0.149 (0.094)
26	0.072 (0.104)	0.078 (0.106)	0.158 (0.105)
27	0.135 (0.144)	0.139 (0.163)	0.201 (0.132)
Overall	**0.057 (0.071)**	**0.083 (0.083)**	**0.133 (0.071)**

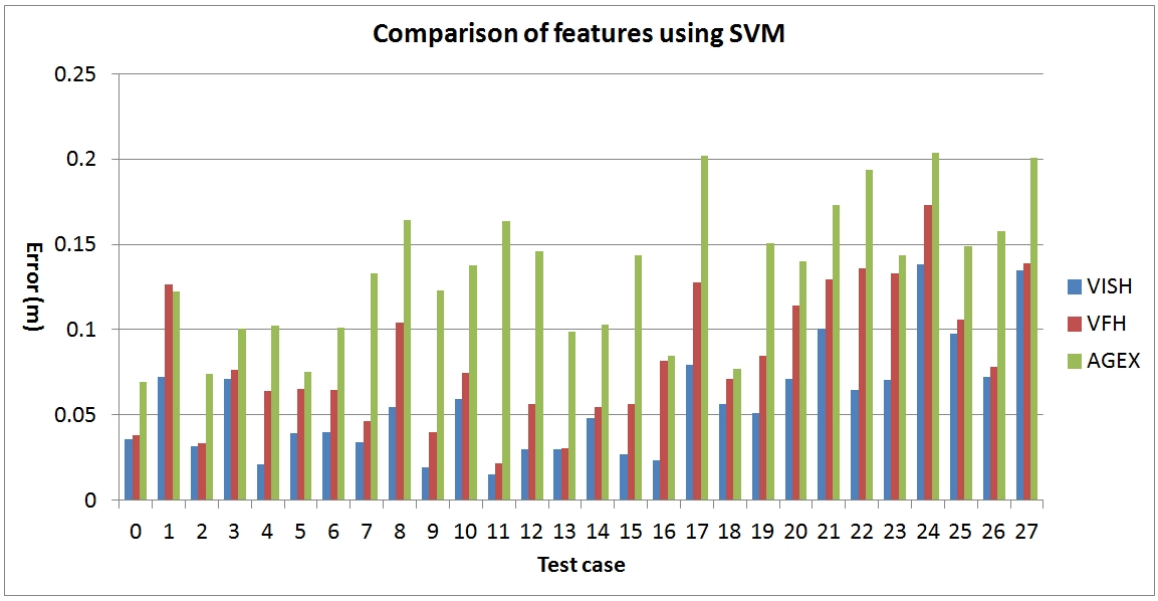


Fig. 2.13. Evaluation of VISH, VFH and AGEX features using SVM.

2.4 Conclusions

In this chapter, we have described the importance of features in data processing. Robust features should be informative and non-redundant in order to facilitate the process of modeling. Variations of observations from the same human poses can be reduced when observations are represented by features. We have formulated the process of feature extraction as a function mapping from an observation to a feature. As depth sensors become popular, 3D point clouds become common observations recently. Therefore, we mainly focus on 3D-point-cloud observations and features. Using 3D-point-cloud features, geometric properties of humans can be utilized for HPE.

As we are inspired by the property of spatial ordering in both visual and 3D-point-cloud observations, we propose a 3D-point-cloud feature called *viewpoint and shape feature histogram* (VISH). It is a 3D adaptation of a HOG feature and captures the shape of a human body. The extraction process of our proposed VISH feature is composed of 3D-point-cloud pre-processing, hierarchical structuring, and VFH and shape feature extraction. In the pre-processing step, methods of region-based thresholding and pseudo-residual are used to

extract 3D points from a person. Those 3D points are then organized into a tree structure. VFH and shape features are extracted separately from each node in a tree. A VISH feature is formed by combining VFH and shape features from all nodes. Therefore, it preserves the spatial ordering of a 3D point cloud of a person. The spatial ordering can capture global and local properties of 3D points in a 3D point cloud. These properties can greatly remove the ambiguity of symmetric human poses.

Two existing 3D-point-cloud features, namely VFH and AGEX, were implemented and compared with our proposed VISH feature. Two common models, namely k-NN and SVM, were used to map a 3D-point-cloud feature to a human pose. Experiment results showed that our proposed VISH feature incurred less errors than the other two features using k-NN and SVM. The results suggested that our proposed VISH feature could describe a 3D point cloud more accurately for 3D HPE. The time complexity of extracting our proposed VISH feature is asymptotically the same as that of extracting a VFH feature, which is $O(n)$, where n is the number of 3D points in a 3D point cloud of a human. The extraction of a VISH feature is more efficient than the extraction of a AGEX feature, which is of time complexity $O(n \log n)$.

In the next chapter, we will propose and describe a human-pose model that maps our proposed 3D-point-cloud feature to a human pose. Our proposed model is different from k-NN and SVM that we will utilize human actions to decompose human-pose space into low-dimensional manifolds.

3. HUMAN-POSE MODELING BASED ON HUMAN ACTIONS

3.1 Introduction

Human-pose models estimate human poses based on features of human observations. They are important in the process of HPE because they can explore and utilize feature properties to achieve an accurate estimation of human poses. For example, a logistic-regression model [80] can selectively adopt some dimensions of features where those dimensions have effects on the accuracy of estimation. The model achieves that by putting more weights on those dimensions. Meanwhile, human-pose models can include assumptions on human poses to simplify the estimation process and achieve a fast estimation. Sometimes, assumptions come from the equipment that captures human observations. For instance, most human-pose models in motion-capturing systems are assumed that calibrated and synchronized sensors, such as cameras and markers, are attached on a human body. Thus, human-joint positions can be estimated directly from marker positions that are captured at the same time instance. In addition, it is common that a sensor is assumed to capture human observations at a high frame rate. Based on the assumption, the change of human poses in human observations is continuous across time [7]. Then, we can represent the temporal coherency of human poses by a model, such as the Gaussian dynamical model in [109]. Human poses can be estimated by using human-pose estimates at the previous frame as initial estimates. Moreover, assumptions are usually made on the structure of a model in order to simplify the inference process in HPE. One common structure in the literature is the conditional random field [110]. With its tractable property, many human-pose models, such as the deformable structure [77] and the mixture-of-parts model [30], have been built.

In general, the main component in a human-pose model is prior knowledge about human poses. We can utilize prior knowledge in deriving a human-pose model, and exploit our belief in the process of estimation. As a result, human-pose estimates are biased to

realistic human poses and the computation of estimation is tractable. Prior knowledge is especially critical in the estimation process when features are not reliable in situations such as occlusions or changes of environment condition. In those situations, features might be missing or ambiguous because human observations are occluded or are different from the features in a set of data for training a human-pose model. Prior knowledge could provide more reliable information about human poses and could reduce or eliminate the negative impact on estimating human poses. A common kind of prior knowledge is a kinematic structure such as the length of a person's limb or the connectivity of human joints. It reduces the variance of human-joint positions estimated by a human-pose model. Examples of models using kinematic structures are the loose-limbed body model [68] and the pictorial structure [81].

In addition to the improvement of estimation when features are unreliable, prior knowledge can be used to reduce the dimensionality of human-pose space, which contains all possible human poses for HPE. As our body is highly articulated and deformable, the degree-of-freedom (number of human joints) of a human pose is high. In our work, the degree-of-freedom of a human pose is 15. Human joints in a human pose are head, neck, left/right shoulder, left/right elbow, left/right wrist, hip center, left/right hip, left/right knee and left/right ankle. Directly modeling the human-pose space associated with a human-pose model is complex and often intractable. However, it is a common belief that human poses lie on low-dimensional manifolds in human-pose space because many points in human-pose space do not correspond to human poses. That belief can be considered as a kind of prior knowledge and can be exploited in the process of estimation in order to eliminate unrealistic human poses and hence reduce the dimensionality of human-pose space. For instance, a human-pose model can be formulated as a conditional model [56] that uses person information such as the height of a person and the length of a person's limb. Then, a low-dimensional manifold that contains human poses of a specific human physique could be defined and utilized in the estimation process.

Despite the variety of human-pose models in the literature, deriving them can be unified as follows. We define the derivation of a human-pose model as the task of deriving a

mapping Γ, which maps a feature vector $f \in F$ to a human pose $p \in P$. The set F is a set of all possible features. The set P is a set of all possible human poses. Therefore, a human-pose model can be written as

$$\Gamma : F \mapsto P. \tag{3.1}$$

Each human-pose model (mapping Γ) is associated with a cost function (or a probability distribution). A cost function depends on a feature f and a human pose p. It can encode the probability of producing human poses in P as the output of a mapping Γ. It can also be multi-modal spatially or temporally because different human poses may have similar features. Prior knowledge can be embedded in a cost function by defining a cost function that increases the probability of generating plausible human poses (human poses of particular types) as the output of a mapping Γ. When a feature f is extracted from an observation, a human pose p is estimated by optimizing a cost function with the extracted feature f being considered as constant. As our body is highly flexible, most cost functions in existing human-pose models are highly non-linear and non-convex. In that case, a global optimum is not guaranteed. Instead, a local optimum is found by some standard optimization techniques. For example, a cost function may be linearized and a local optimum is found by following the gradient of a cost function with respect to parameters of human poses.

Different types of human-pose models, such as generative and discriminative models, can be described using the formulation above. Generative models generate a number of possible human poses, and evaluate the similarity between a feature from a possible human pose and a feature extracted from an observation of a person. In general, a mapping Γ in a generative model involves computations in three steps, namely human-pose representation, feature creation, and feature matching. In the first step, a human pose is represented using a number of parameters such as the position and orientation of body parts. Using different values of parameters in a human-pose representation, possible human poses can be created. Based on the generated human poses, features are generated in the second step. Usually,

assumptions on observations are made in this step because observations in real life are hard to generate. The difficulty comes from unexpected events. For example, the clothing of a person is unpredictable and the background is cluttered. Finally, the generated features are compared with features extracted from observations of a person. The similarity between features is encoded in a form of a cost function. The output of a mapping Γ is a possible human pose that creates features closest to features of observations.

On the other hand, a mapping Γ in a discriminative model directly estimates a human pose from a given feature without the step of feature creation. A cost function in a discriminative model is a probability distribution of human poses given a feature. The output of a mapping Γ is the most probable human pose according to a probability distribution (a cost function). As discriminative models do not require the creation of features, their computation is usually faster than the computation in generative models.

Since the high degree-of-freedom of human poses induces complex human-pose models, we use knowledge about human actions as prior knowledge in order to discover low-dimensional manifolds in human-pose space. Figure 3.1 shows the idea. In the figure, each data point represents a human pose. Human poses coming from an action should share some common properties that are described by a low-dimensional manifold. As the same human pose may appear in different actions, there should be some data points that lie on two or more low-dimensional manifolds.

Some previous works [71] [73] have been using human action as prior knowledge as well. They showed that human action was useful in HPE. Gall *et al.* [71] proposed a model for estimating human poses by computing the probability of actions based on action classification. Using the probability, a human pose was assigned in an action-specific manifold. Yao *et al.* [73] proposed the appearance-based and pose-based features to classify actions using the Hough-forest algorithm [101]. Yao *et al.* [74] further extended the model in [73] into a single framework. The dimensionality of human-pose space was reduced by considering the most probable action determined by action classification. In general, a human pose can appear in more than one action. For example, the human pose of hand waving can appear in the actions of standing and raising both arms. Therefore, we extend this concept

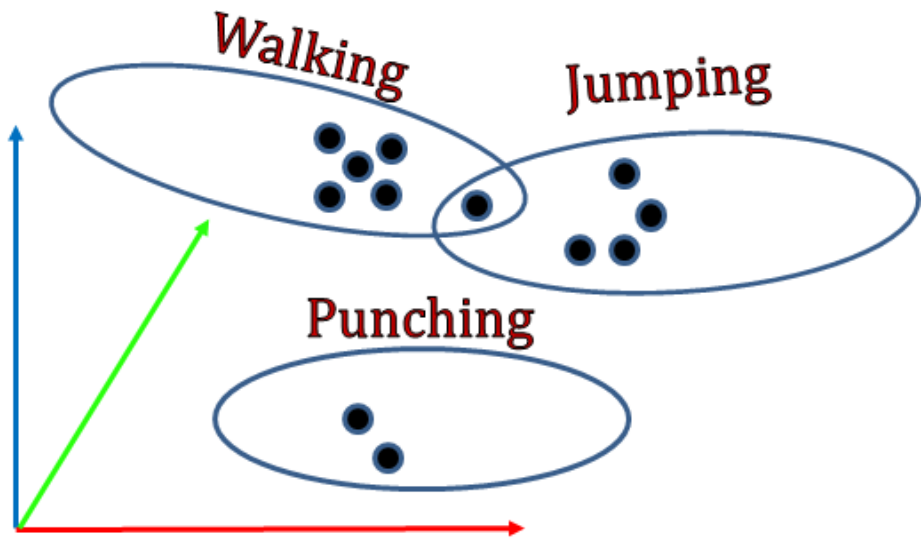


Fig. 3.1. The concept of using human actions to discover low-dimensional manifolds in human-pose space.

in a discriminative model by considering that each human pose is originated from multiple actions. As we will show in the experiment section, using multiple actions can lead to a more accurate human-pose estimate than using a single action.

In this chapter, we will propose a discriminative model that utilizes human actions in HPE. The discriminative model is called the action-mixture model (AMM). It uses a 3D-point-cloud feature called a *viewpoint-and-shape-feature-histogram (VISH)* feature, which is described in Chapter 2, as input to capture both global and local properties of a 3D point cloud of a human. It then uses the result from action classification to represent human-pose space using low-dimensional manifolds in estimating human poses. To avoid accumulating errors from previous frames, our proposed model uses temporal information only for training but not for testing. Human-pose space is represented by low-dimensional manifolds, each of which corresponds to one action. The proposed AMM is different from the previous works [73] [71] in that a human pose may appear in more than one action. One limitation of the proposed AMM is that human-pose space in the proposed AMM is discrete. Thus, there are quantization errors in human poses estimated by AMM. To reduce the quantization error, a 3D-point-cloud system is proposed. The proposed 3D-point-cloud

system is shown in Figure 3.2. It uses a kinematic model is to refine human poses estimated by the proposed AMM. In the kinematic model, the angle of each body part is parameterized by a quaternion [111] to explicitly represent the spatial relationship between body parts. The proposed 3D-point-cloud system was tested on the Stanford TOF Motion Capture Dataset [70]. Computer simulations showed that the proposed 3D-point-cloud system reduced the overall error and standard deviation of human-pose estimates compared with some existing approaches.

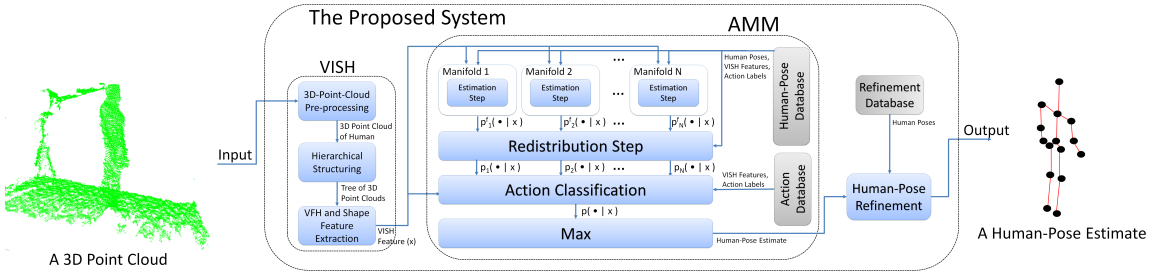


Fig. 3.2. The proposed 3D-point-cloud system for HPE.

The structure of this chapter is as follows. Section 3.2 describes the proposed AMM, which estimates human poses from a VISH feature. Section 3.3 describes a kinematic model that reduce the quantization error in human-pose estimates. Experimental results are discussed in Section 3.4. Conclusions are presented in Section 3.5.

3.2 Action-Mixture Model (AMM)

The *Action-Mixture Model* (AMM) is a human-pose model that takes a VISH feature of a 3D point cloud of a human as input and estimates the corresponding human pose. We assume that each human pose comes from an action. We also assume that a human-pose estimate is one of the human poses in a human-pose database. Main components in AMM are as shown in Figure 3.3. In the figure, both the action database and human-pose database are prepared in advance. In the experiment, they were built using a benchmark dataset. Each data in the action database contains a ground-truth human pose (a set of 15 human-joint positions) and an action label indicating the ground-truth action that the human pose

belongs to. Each data in the human-pose database contains a ground-truth human pose, a VISH feature, and an action label. More details about the experimental setup can be found in Section 3.4. For a given VISH-feature input from a person, the probability of each human pose in the human-pose database being the human pose of the person is estimated. We formulate the probability as a function of actions. Thus, the probability is computed for each action. It is computed by arranging human poses in the human-pose database according to actions that they belongs to. Meanwhile, a VISH-feature input is classified into one or more actions using the action database. The result of action classification is the probability of a human pose belonging to an action. Based on the probability, weighting coefficients for each action are determined for combining the estimation results from each action. They appear in a form of an unnormalized probability mass function (pmf). Finally, the probability of human poses in the human-pose database being the human pose from a given VISH-feature is computed by combining the probability for each action according to weight coefficients. The most probable human pose in the human-pose database becomes the output of the proposed AMM.

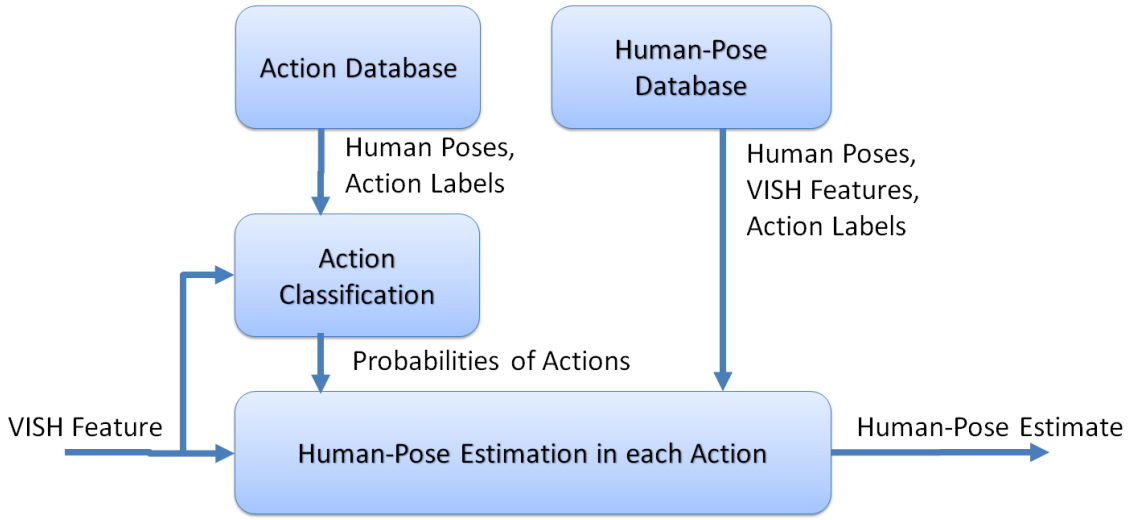


Fig. 3.3. Main components in the action-mixture model (AMM).

Based on the concept described above, AMM is formulated as follows. We represent AMM as a Bayesian network. A Bayesian network is a directed acyclic graph (DAG) in

which nodes are random variables and absence of edges represents conditional independence assumptions on random variables. Three random variables are defined in AMM. The first random variable, denoted by $\mathbf{X}$, is an observed variable. It represents a VISH feature. The second random variable, denoted by A, is a hidden variable. It represents an action. The third random variable, denoted by Y, is a target variable. It represents a human pose. We define the joint distribution of action and pose variables, conditioned on a VISH variable as a product of two terms, namely the probability of an action variable, conditioned on a VISH variable, and the base distribution that defines the probability of a human-pose variable, conditioned on action and VISH variable. Mathematically, the joint distribution is given by

$$p(a, y \mid \mathbf{x}) = p(a \mid \mathbf{x}) p(y \mid a, \mathbf{x}), \tag{3.2}$$

where $\mathbf{x}, a, y$ are observed values of VISH, action and pose variables, respectively. Figure 3.4 shows the DAG and the factor graph of AMM.

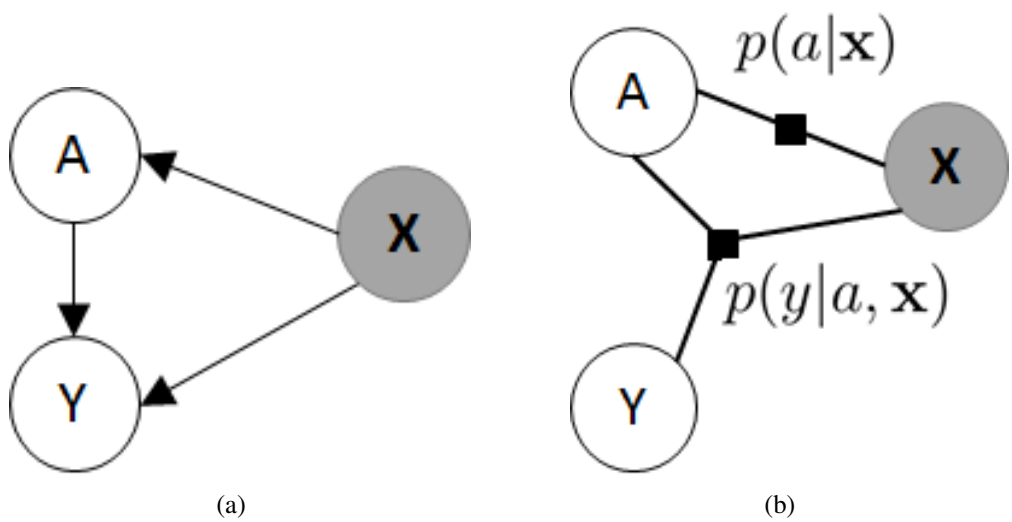


Fig. 3.4. (a) DAG and (b) factor graph of AMM. The observed variable $\mathbf{X}$ is shaded.

Inference on a human-pose variable is done by first marginalizing the joint distribution in Eq. (3.2) over an action variable. The conditional probability distribution of a human pose that is denoted as y is given by

$$p(y \mid \mathbf{x}) = \sum_{a=1}^{N} p(a, y \mid \mathbf{x}), \tag{3.3}$$

where N is the number of actions. Then, a human pose corresponding to a VISH feature $\mathbf{x}$ is estimated by the most probable human pose according to the human-pose distribution computed by Eq. (3.3); that is,

$$y^* = \arg \max_{y \in \{y_1, \ldots, y_M\}} p(y \mid \mathbf{x}), \tag{3.4}$$

where y^* is the most probable human pose among all human poses in the human-pose database, y_i is the i-th possible human pose that is called the i-th key pose in the human-pose database, and M is the number of key poses.

For the sake of computing the joint distribution in Eq. (3.2), the probability of an action variable conditioned on a VISH variable is rewritten as an unnormalized pmf divided by a normalizing constant. Mathematically, the joint distribution in Eq. (3.2) is rewritten as follows.

$$p(y \mid \mathbf{x}) = \frac{1}{Z} \sum_{a=1}^{N} g_a(\mathbf{x}) p(y \mid a, \mathbf{x}), \tag{3.5}$$

where $\{g_a(\mathbf{x})\}_{a=1}^{N}$ is the unnormalized pmf for action classification and Z is a normalizing constant.

As the estimation process is to find the most probable human pose among all key poses in the human-pose database, the normalizing constant Z can be eliminated and thus is not calculated. In other words, AMM incorporates the result from action classification to HPE through the unnormalized pmf $\{g_a(\mathbf{x})\}_{a=1}^{N}$. Thus, the computation of the joint distribution in Eq. (3.5) involves computations of two terms, namely the unnormalized pmf of action classification and the base distribution of HPE in each action. In modeling the unnormalized pmf of action classification, the bootstrap aggregating algorithm (bagging) [112], which improves the accuracy of classification, is used. In modeling the base distribution,

human poses are modeled in each action. As the distribution of human poses is unknown, it is derived based on the instance-based learning algorithm [80].

3.2.1 Action Classification

A VISH-feature input is classified by bagging classification trees. Bootstrap samples are generated from the action database to train tree classifiers for predicting the action associated with a VISH-feature input. The tree classifiers are then aggregated by voting. Let $\phi(x,\mathscr{L})$ be the tree classifier that is trained using the bootstrap sample $\mathscr{L}$. The weight of the HPE in each action is then modeled as the relative frequency of the action predicted by the tree classifier over all bootstrap samples; that is,

$$g_a(x) = E_L[\delta(\phi(x,\mathscr{L}) - a)], \tag{3.6}$$

where $E_L[\cdot]$ is the expectation operator over all bootstrap samples and $\delta(\cdot)$ is defined as

$$\delta(e) = \begin{cases} 1 & \text{if } e = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

For each bootstrap sample $\mathscr{L}$, the tree classifier $\phi(x,\mathscr{L})$ is trained by the classification and regression trees algorithm (CART). All nodes except leaves in the tree classifier are discrete splitting functions that split the space of a single dimension, denoted as d, of the input feature into 2 subspaces. The leaves in the tree classifier are the indices of action classes. The splitting function depends on the twoing criterion which is given by

$$\begin{aligned} twoing(d,\mathscr{S}_{d_\leq},\mathscr{S}_{d_>},\mathscr{S}) = & \frac{|\mathscr{S}_{d_\leq}||\mathscr{S}_{d_>}|}{4|\mathscr{S}|^2} \times \\ & \left(\sum_{i=1}^{N}\left|\frac{|\mathscr{S}_{d_\leq}\cap\mathscr{S}^i|}{|\mathscr{S}_{d_\leq}|} - \frac{|\mathscr{S}_{d_>}\cap\mathscr{S}^i|}{|\mathscr{S}_{d_>}|}\right|\right)^2, \end{aligned} \tag{3.8}$$

where $\mathscr{S}_{d_\leq}$ and $\mathscr{S}_{d_>}$ are two mutually exclusive and exhaustive sets over the training set $\mathscr{S}$ filtered by parent nodes, $\mathscr{S}^i$ is the set of human poses from the i-th action in the set $\mathscr{S}$

and $|\cdot|$ is the cardinality of the input set. Note that the training set $\mathscr{S}$ at the root node is the bootstrap sample $\mathscr{L}$.

The optimal dimension d and the subspaces $\mathscr{S}_{d_\leq}, \mathscr{S}_{d_>}$ are determined by minimizing the twoing criterion. New nodes are added repeatedly until all the poses in the training set $\mathscr{S}$ belong to the same action class or the maximum tree depth is reached. Then, the tree classifier is pruned to avoid overfitting by minimizing the cost-complexity pruning measure, denoted as α, which is given by

$$\alpha = \frac{\varepsilon(T', \mathscr{S}) - \varepsilon(T, \mathscr{S})}{|leaves(T)| - |leaves(T')|}, \tag{3.9}$$

where T is a tree classifier, T' is the tree classifier after replacing a node by the index of the action class that the majority of human poses in the training set $\mathscr{S}$ belongs to, $\varepsilon(T, \mathscr{S})$ is the classification error from the tree classifier T over the training set $\mathscr{S}$ and $leaves(T)$ is the number of leaves in the tree classifier T.

The tree classifier is pruned and stored repeatedly until the number of iterations reaches a predefined number. Then, among all the tree classifiers, the tree classifier which minimizes the classification error over a validation set of human poses is selected as the final tree classifier $\phi(x, \mathscr{L})$.

3.2.2 Base Distribution for HPE

The base distribution $p(y \mid a, \mathbf{x})$ in Eq. (3.5) is modeled using the instance-based learning algorithm [80]. Instances are pairs of ground-truth human poses and the corresponding VISH features. They are collected in advance to form a database called human-pose database, denoted as $\mathscr{D}$. The human-pose database defines the range of the base distribution. In other words, it contains all possible human-pose estimates. The base distribution is quantized by human poses from the human-pose database, denoted as y'. In the human-pose database, the human poses from the i-th action are grouped to form a subset of the human-pose database, denoted as $\mathscr{D}_i$.

The base distribution is estimated in two steps: estimation step and redistribution step. The estimation step calculates the Euclidean distance between the VISH-feature input and the VISH features from the human-pose database, and produces a probability distribution of human poses in each action as output. The redistribution step weights the probability distributions among all actions from the estimation step according to the likelihood of the VISH-feature input occurred in different actions. The outputs from the two steps are then combined to give the estimate of the base distribution. Note that the redistribution step is different from the action classification discussed in Sectoin 3.2.1 that the redistribution step utilizes the temporal information of actions in the human-pose database.

Estimation Step

The base distribution in Eq. (3.5) is computed through the computations of two parts. The first-part computation is performed in the estimation step. Recall that the base distribution is the probability distribution of human poses in the human-pose database being a human-pose estimate. It is computed for each action. We expect that similar human poses yield similar VISH features. Thus, we define the first part of the base distribution based on the inverse of the Euclidean distance between a VISH-feature input and VISH features of human poses in the human-pose database. Mathematically, the first part is represented by an unnormalized probability distribution. We define the unnormalized probability of a human pose in the i-th action as

$$\widetilde{p}_i^f(x' \mid x) = \begin{cases} \frac{1}{\|x-x'\|_2+z} & \text{when } x' \in A_i, \\ \frac{1}{N} \cdot \frac{1}{\|x-x'\|_2+z} & \text{when } x' \notin A_i, \quad \forall i = 1,2,\ldots,N, \end{cases} \tag{3.10}$$

where x' is the VISH feature of a human pose in the human-pose database, i is the index of an action, A_i is the set of VISH features that the corresponding human poses are in $\mathscr{D}_i$, z is a small constant to avoid division by zero, N is the number of actions, and $\|\cdot\|_2$ is the Euclidean norm.

Based on the Eq. (3.10), the unnormalized probability of a human pose is larger if a VISH-feature input is closer to a VISH feature of a human pose in the human-pose database in the Euclidean space. We include the factor $\frac{1}{N}$ in Eq. (3.10) to penalize the case when an action associated with a VISH feature from the human-pose database is different from the action of the unnormalized probability.

Based on the unnormalized probability, we can compute the probability of a human pose by normalization. Mathematically, the probability of a human pose in the i-th action is given by

$$p_i^f(x' \mid x) = \frac{\widetilde{p}_i^f(x' \mid x)}{\sum_{x' \in A} \widetilde{p}_i^f(x' \mid x)}, \tag{3.11}$$

where A is the union of the feature sets $A_1, A_2, \ldots, A_N$.

To illustrate the idea in the estimation step, we show an example of computing the probability of a human pose estimated in the estimation step. In the example, a person, whose human poses is estimated, performs three actions. There are ten human poses, which are represented by circles in the figures, in the human-pose database. Probability distributions for the three actions are computed according to Eq. (3.11). The probability distributions of the three actions are shown in Figures 3.5, 3.6, and 3.7. In the figures, the size of circles represents the probability distributions. After the computation of the probability distributions, the size of circles is changed according to the probability distributions. Typically, the size of circles in the action that corresponds to the distribution is larger than the size of circles in other actions. It is because there is a factor of $1/N(N=3)$ being multiplied to the unnormalized probability of other actions in Eq. (3.10). The unnormalized probability of other actions is not zero because we believe that human poses from different actions may be similar.

Redistribution Step

In the second part, we compute another quantity that contributes to the base distribution in Eq. (3.5). We expect that the quantity should reflect the probability of observing human poses from different actions. Specifically, the probability of a human pose in the human-

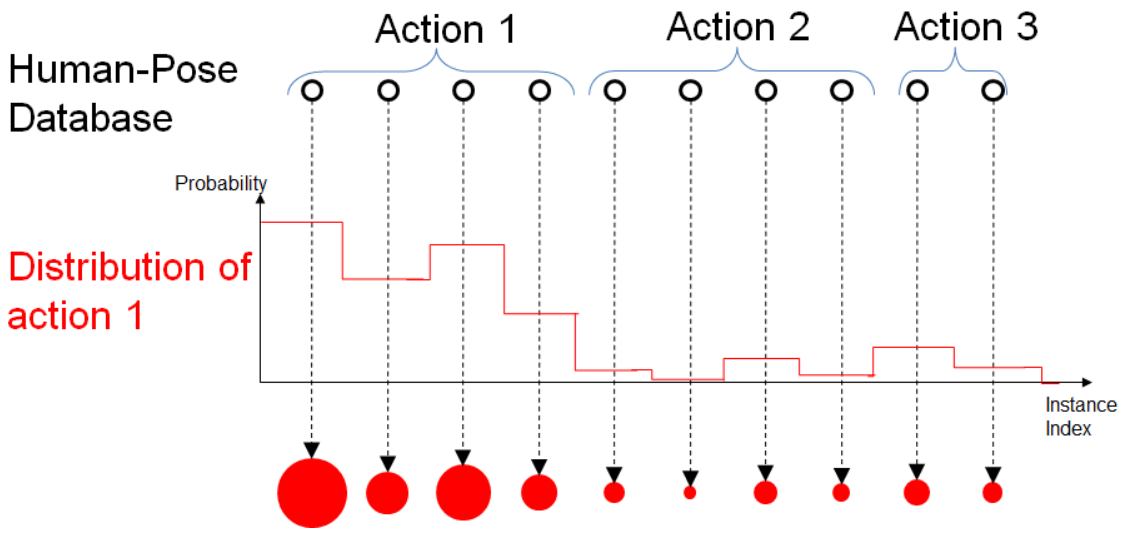


Fig. 3.5. An example of the computation in the estimation step for action 1. Each circle represents a human pose in the human-pose database. The size of a circle represents the probability calculated in the estimation step. Refer to the text for a detailed description.

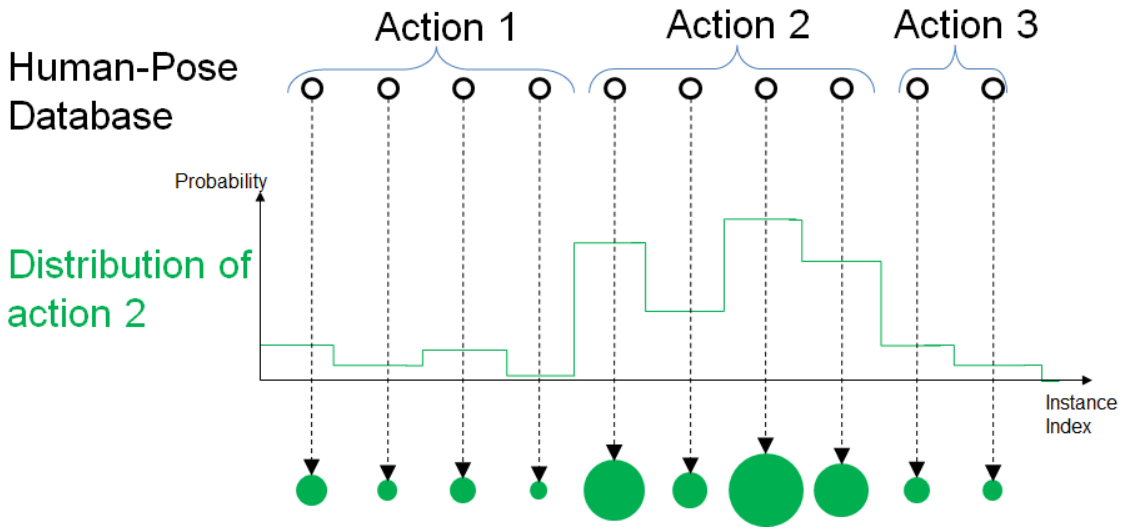


Fig. 3.6. An example of the computation in the estimation step for action 2. Each circle represents a human pose in the human-pose database. The size of a circle represents the probability calculated in the estimation step. Refer to the text for a detailed description.

pose database should be increased after knowing that a human is performing the action that contains the human pose. Based on the expectation, we define the quantity as the sum of the probability distributions estimated from the estimation step weighted according to the

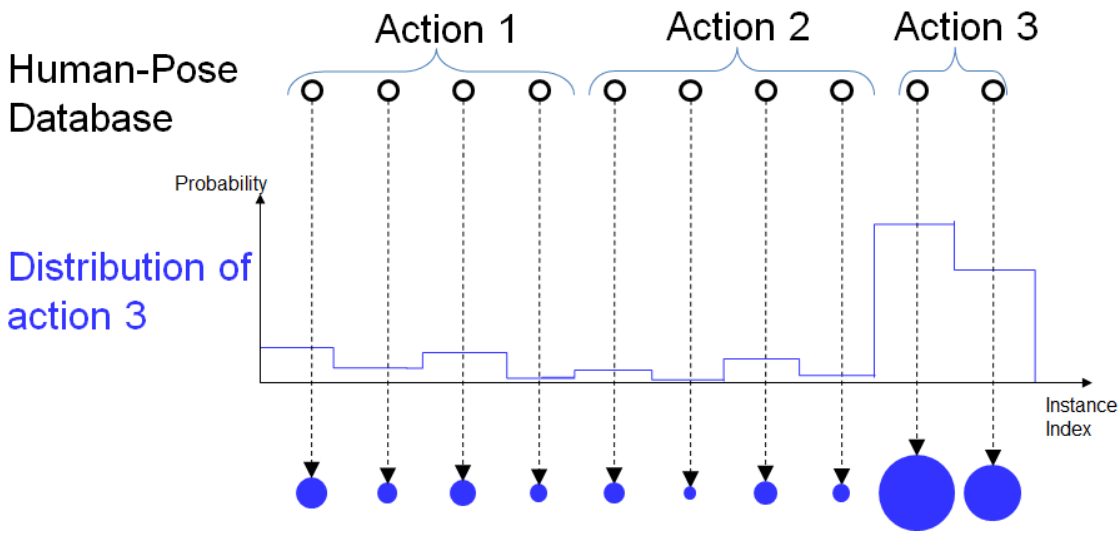


Fig. 3.7. An example of the computation in the estimation step for action 3. Each circle represents a human pose in the human-pose database. The size of a circle represents the probability calculated in the estimation step. Refer to the text for a detailed description.

likelihood of a VISH-feature input occurred in different actions. To model the occurrence of an action, we assume that, given the present action in a sequence of actions, the rest of the past actions is irrelevant for predicting the future actions. With the assumption, the weight is formulated as the stationary probability in a continuous-time Markov chain, which is trained using VISH features in the human-pose databases.

Let $X(t)$ be the continuous-time Markov chain with the state space $I = \{1, 2, ..., N\}$ for $t \geq 0$. The state space contains the indices of N actions. We assume that the Markov chain is temporally homogeneous. Based on the assumptions above, for any $s > 0$, the probability of the Markov chain being at state j at time $t + s$ given the current and the past states could be simplified as follows.

$$\begin{aligned} p(X(t+s) = j | X(s) = i, X(s_n) = i_n, ..., X(s_0) = i_0) \\ = p(X(t) = j | X(0) = i), \end{aligned} \tag{3.12}$$

where $s_0, s_1, ..., s_n, s$ are the $n+2$ samples of time such that $s > s_n > \cdots > s_0 \geq 0$ and $i_0, i_1, ..., i_n, i$ are the corresponding states.

The (i, j) entry of the transition probability matrix, denoted as Q, of the Markov chain $X(t)$ is defined as

$$Q(i,j) = \begin{cases} \lim\limits_{h \to 0} \frac{p(X(h)=j|X(0)=i)}{h} & \text{when } i \neq j, \\ \lim\limits_{h \to 0} \frac{p(X(h)=i|X(0)=i)-1}{h} & \text{when } i = j. \end{cases} \tag{3.13}$$

For $i \neq j$, the transition probability measures the jump rate of the Markov chain from state i to j. For $i = j$, the transition probability is the negation of the rate at which the Markov chain leaves state i. The jump rate is estimated by modeling the transition of actions in a Poisson process [113] using the temporal information in the human-pose database.

Let λ_{ij} be the arrival rate of a state from i to j. The arrival rate between two actions is calculated by the normalized dynamic time warping algorithm (DTW) [82] that measures the similarity between two actions; that is,

$$\lambda_{ij} = \begin{cases} \frac{z_1}{DTW(i,j)+r} & \text{when } DTW(i,j) < \tau, \\ 0 & \text{when } DTW(i,j) \geq \tau, \end{cases} \tag{3.14}$$

where z_1 is a constant, τ is a predefined threshold, $DTW(i, j)$ is the distance between the i-th and j-th actions calculated by the normalized DTW and r is a uniform random variable between 0 and 1.

The normalized DTW is used because actions may be different in time or speed. Given two actions, the normalized DTW calculates their matching cost under the optimal alignment by warping the two actions. The matching cost of a pair of frames in two actions is defined as the Euclidean distance between the VISH features extracted from the 3D point clouds in the two frames.

As one kind of action can be changed to another action at any time, we assume the arrival of a state is equally likely at all time. Thus, if one unit of time is divided into m

intervals, the probability of the arrival of state j from state i in each interval is $\frac{\lambda_{ij}}{m}$. The probability of the first arrival after time t can be approximated by

$$(1-\frac{\lambda_{ij}}{m})^{tm} \xrightarrow[m\to\infty]{} e^{-\lambda_{ij}t}. \tag{3.15}$$

Therefore, the inter-arrival time is exponentially distributed with rate λ_{ij}. Hence, the (i,j) entry of the transition probability matrix Q is given by

$$Q(i,j) = \begin{cases} \lambda_{ij} & \text{when } i \neq j, \\ -\sum_{k=1,k\neq i}^{N} \lambda_{ik} & \text{when } i = j. \end{cases} \tag{3.16}$$

The state space I is partitioned into a minimum number, denoted as M, of mutually exclusive and exhaustive sets such that the continuous-time Markov chain with any of the M partitioned sets is irreducible. Let $X_k(t)$ be the continuous-time Markov chain with the k-th partitioned set I_k, where $k = 1,2,\ldots,M$. The transition probability matrix, denoted as Q_k of the Markov chain $X_k(t)$ can be formed from the transition probability matrix Q by deleting its rows and columns of the corresponding actions that are not in the state space of the Markov chain $X_k(t)$. If the Markov chain $X_k(t)$ is positive recurrent, then the stationary distribution of the Markov chain $X_k(t)$, denoted as π_k, can be found by solving $\pi_k Q_k = 0$; otherwise, the stationary distribution is set to be uniform to indicate equal importance of each action.

The probability distributions estimated from the estimation step are then weighted according to the stationary distribution as follows,

$$p_i^a(x' \mid x) = \pi_k(i) \sum_{j\in I_k} p_j^f(x' \mid x), \quad i \in I_k, \tag{3.17}$$

where $\pi_k(i)$ is the stationary distribution of state i in the Markov chain $X_k(t)$.

To illustrate the idea, we show an example of the computation in the redistribution step. The example utilizes the probability distributions that are computed in an example of the estimation step as shown in Figures 3.5, 3.6, and 3.7. It is shown in Figure 3.8. In the

figure, each circle represents a human pose in the human-pose database. The size of a circle represents the probability that is calculated in the estimation step. In this example, there are three actions and the number of human poses in the human-pose database is 10. We assume that the minimum number M that partitions the state space I is 3. In other words, the k-th partitioned set I_k of the state space I is k, where $k = 1, 2, 3$. The probability distribution in the left side of the figure is the stationary distribution of a continuous-time Markov chain that measures the similarity of actions. Based on the probability distributions that are computed in the estimation step, the quantity that is computed in the redistribution step is calculated by multiplying the probability of each action specified by the stationary distribution.

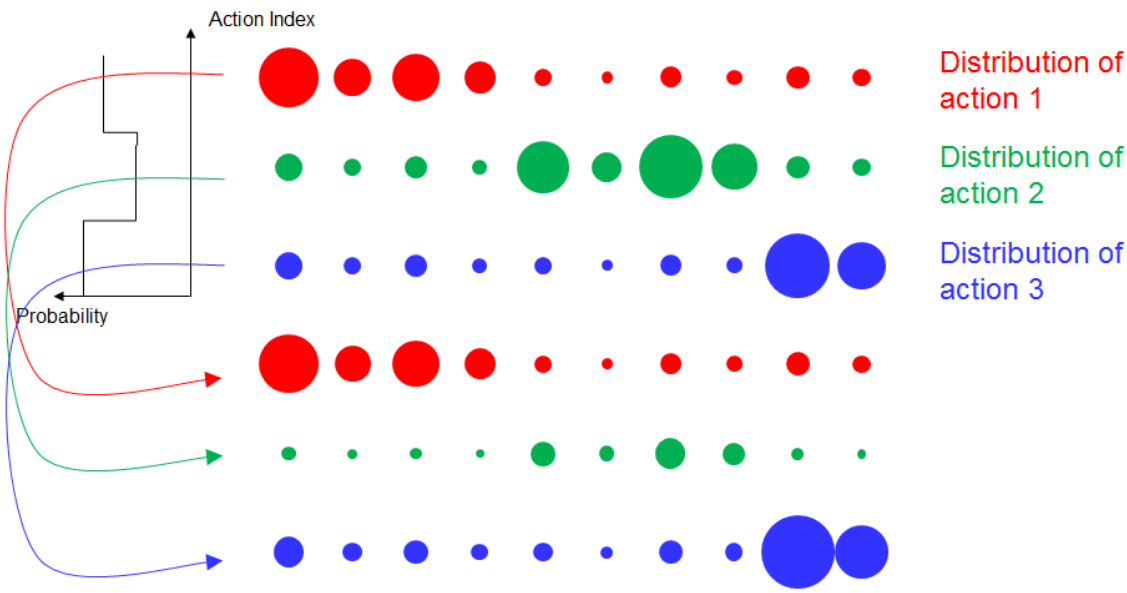


Fig. 3.8. An example of the computation in the redistribution step. Each circle represents a human pose in the human-pose database. The size of a circle represents the probability calculated in the estimation step. The distribution in the left is the stationary distribution of a continuous-time Markov chain. Refer to text for more details.

After computing the probability distribution of human poses in each action in the estimation step, and the probability distribution of human poses that are redistributed according to the likelihood of a VISH-feature input occurred in different actions in the redistribution

step, the two probability distributions are first combined as an unnormalized base distribution. The unnormalized base distribution is then normalized to form the base distribution in AMM. We denote the unnormalized base distribution as $\tilde{p}_i(y'|x)$. The unnormalized base distribution is computed by linearly combining the probability distributions computed from the two steps. Mathematically, it is given by

$$\tilde{p}_i(y' \mid x) = u^f p_i^f(x' \mid x) + u^a p_i^a(x' \mid x), \tag{3.18}$$

where x' is a VISH feature associated with a human pose y', u^f and u^a are user-defined constants. If u^f is larger (smaller) than u^a, the probability distribution estimated from the estimation step will have more (less) influence on the unnormalized base distribution.

The base distribution $p(y' \mid a = i, x)$ is then derived by normalizing the unnormalized base distribution. It is given by

$$p(y' \mid a = i, x) = \frac{\tilde{p}_i(y' \mid x)}{\sum_{y' \in D_i} \tilde{p}_i(y' \mid x)}. \tag{3.19}$$

After computing the base distribution in AMM for each action, base distributions from all actions are aggregated according to the result of action classification. Figure 3.9 shows an example of the aggregation of base distributions to form a probability distribution of each human pose in the human-pose database being a human pose estimated by AMM. In the figure, circles represent human poses in the human-pose database. Their sizes represent the probability of human poses according to the base distribution computed through the estimation and redistribution steps. The color of each circle represents the action corresponding to the base distribution. A, B, and C are probabilities of a VISH-feature input belonging to actions 1, 2, and 3, respectively. They are also the weights in combining the base distributions. They are computed by performing bagging, which is a method of the action classification, on a VISH-feature input. To combine the base distributions in the figure, base distributions are multiplied by A, B, and C and are added together. The sum is the probability distribution of human poses in AMM. The human pose corresponding to the

largest circle is the human-pose estimate. As we will show in the experiment, the proposed AMM can increase the accuracy of human-pose estimates by utilizing actions in HPE.

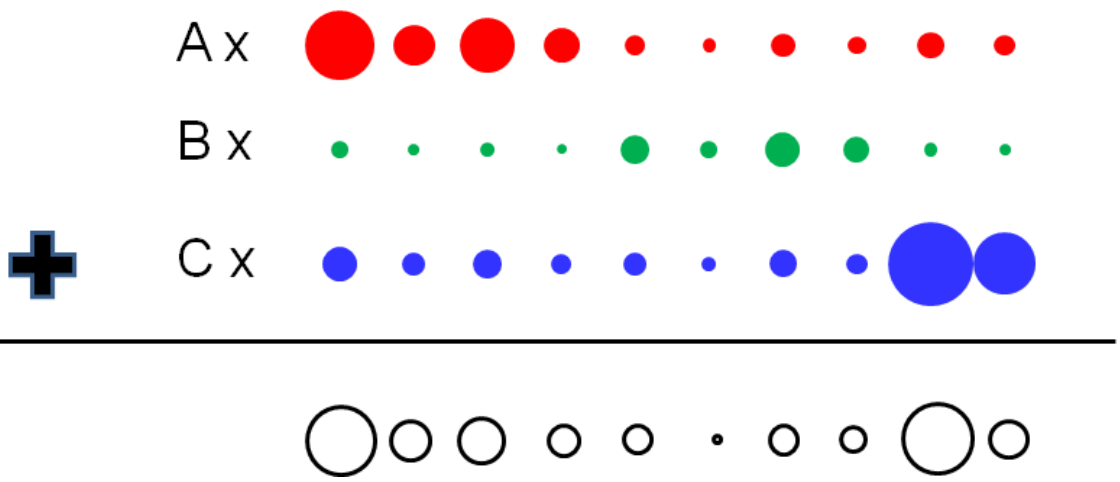


Fig. 3.9. An example of an aggregation of base distributions in AMM. Circles represent human poses in the human-pose database. Their sizes represent the probability of the corresponding human poses being a human-pose estimate. There are three actions in this example. A, B, C are coefficients in the aggregation. They are computed by estimating the probability of a human pose coming from the three actions through a method of action classification called bagging.

Although AMM can increase the accuracy of human-pose estimates, human-pose estimates are limited to one of the human poses in the human-pose database. In other words, human poses are estimated in discrete space. Thus, quantization error is induced in human-pose estimates. In the next section, we will describe a kinematic model that reduces the quantization error.

3.3 Kinematic Model

A human pose that is estimated by our proposed AMM is one of the human poses in the human-pose database. As human poses in the human-pose database cannot fully cover all possible human poses in real-world situations, a human-pose estimate may contain a quantization error. To reduce the quantization error, the spatial relationship between body

parts of the human poses estimated by AMM is modeled by a parametric distribution to simulate the variation.

We assume that there is an underlying probability distribution governing the position and orientation of body parts. We define the distribution as a DAG $G = (V, E)$ as shown in Figure 3.10, where V corresponds to a set of vertices and E corresponds to a set of edges. In the figure, the vertex 1 defines the probability distribution of a human pose in the human-pose database being a human-pose estimate. Since the vertex 1 defines the probability distribution, it is represented by a softmax random variable [80] of a human pose estimated by AMM. Let $V^- = V \setminus \{1\}$. Each vertex $s \in V^-$ corresponds to a body part and there is a random variable, denoted as O_s, representing the orientation of the body part with respect to its parent in the kinematic chain that is represented by Figure 3.10. The parent of the torso is set to be null and the orientation of the torso is measured with respect to the normal of the floor plane. The length of each body part is assumed to be fixed and the orientation is represented by a quaternion. As a result, O_s lies on a 4D manifold.

As vertices in the DAG in Figure 3.10 are random variables, a configuration of a human (a set of orientations of body parts) is stochastic in nature. We denote a stochastic configuration of a human as C. A stochastic configuration can be written as

$$C = \{O_2, O_3, \ldots, O_{15}\}. \tag{3.20}$$

Let $c, o_2, ..., o_{15}$ be the realization of the random variables $C, O_2, \ldots, O_{15}$, respectively. Given the graph G, the probability of a configuration c can be written as

$$p(c) = p(c \mid v_1) = p(\{o_2, o_3, \ldots, o_{15}\} \mid v_1) = p(o_2, o_3, \ldots, o_{15}), \tag{3.21}$$

where the parentheses and the conditioning event v_1 are removed for notational simplicity.

Modeling the probability distribution $p(C)$ is generally intractable because of the high dimensionality of human-pose space. However, we can exploit the dependencies and independencies in the graph G to simplify the computation of the probability distribution $p(C)$

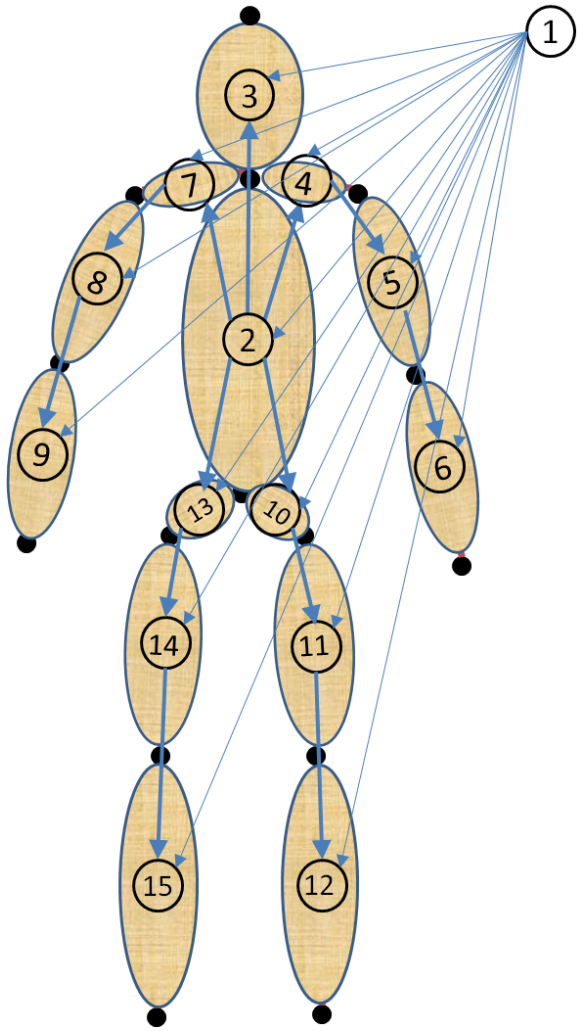


Fig. 3.10. The kinematic relationship between body parts of a human. The vertices are: 1. human pose estimated by the proposed AMM, 2. torso, 3. head, 4. left shoulder, 5. left upper arm, 6. left lower arm, 7. right shoulder, 8. right upper arm, 9. right lower arm, 10. left hip, 11. left upper leg, 12. left lower leg, 13. right hip, 14. right upper leg, 15. right lower leg. Arrows represent dependencies between vertices.

based on the kinematic chain in Figure 3.10. Mathematically, the probability distribution $p(C)$ can be rewritten as

$$p(C) = \prod_{s \in V^-} p(O_s \mid pa(O_s)), \tag{3.22}$$

where $pa(\cdot) : V^- \mapsto V^-$ is a mapping from a vertex to its parent in the kinematic chain and the parent of the torso is $\emptyset$ by definition.

Using the kinematic relationship between body parts, the probability distribution in Eq. (3.22) is tractable. For each body part, $p(O_s \mid pa(O_s))$ is assumed to be a Gaussian distribution with a mean vector μ_s and a positive-definite variance matrix Σ_s; that is,

$$p(O_s \mid pa(O_s)) = \mathcal{N}(\mu_s, \Sigma_s), \tag{3.23}$$

where $\mathcal{N}$ is a Gaussian distribution.

To find the parameters μ_s and Σ_s in the kinematic model, a refinement database, denoted as $\mathcal{T}$, is created. The parameters can then be found by maximizing the log-likelihood,

$$\sum_{n=1}^{|\mathcal{T}|} \log p(c^n) = \sum_{n=1}^{|\mathcal{T}|} \sum_{s \in V^-} \log p(O_s^n \mid pa(O_s^n)), \tag{3.24}$$

where c^n is the n-th configuration in the refinement database $\mathcal{T}$ and O_s^n is the orientation of a body part at vertex s in the n-th configuration.

When refining a human pose using the kinematic chain, body parts are divided into observable and unobservable groups. The orientation of a body part in the observable group can be determined by finding the orientation of the line joining the joint positions at the two ends of the body part. The joint positions are obtained from the human pose estimated by the AMM. Inference is made based on the orientation of the body parts in the observable group to estimate the orientation of the body parts in the unobservable group. Mathematically, the set of vertices in the graph G is divided into evidential (observable) and non-evidential (unobservable) sets; that is,

$$V^- = V_e^- \cup V_n^- \quad \text{and} \quad V_e^- \cap V_n^- = \emptyset, \tag{3.25}$$

where V_e^- is the set of evidential vertices and V_n^- is the set of non-evidential vertices.

Given the orientation of the body parts in V_e^-, the orientation in the non-evidential vertices can be estimated by the most likely configuration, denoted as c^*, of the probability distribution of configuration; that is,

$$c^* = \underset{c \in C,\ s \in V_e^-}{\arg\max}\ p(c \mid o_s). \tag{3.26}$$

Based on the factorization structure of the distribution in the kinematic model, Eq. (3.26) can be calculated efficiently using belief propagation [89]. The orientation of body parts that maximizes Eq. (3.22) are then converted to a human pose and averaged with the human pose estimated from the proposed AMM to output the refined human-pose estimate.

3.4 Experimental Results

The proposed AMM and 3D-point-cloud system were implemented using the point cloud library (PCL) [108]. They were tested on the Stanford TOF Motion Capture Dataset [70], which contains 28 video sequences. Each video sequence corresponds to one action. The skeleton of the subject has 15 degrees-of-freedom (human joints), namely head, neck, left/right shoulder, left/right elbow, left/right wrist, hip center, left/right hip, left/right knee and left/right ankle. The number of frames of the video sequences is shown in Table 4.1. In the dataset, a subject performed different actions such as kicking and rotation, and was captured by the Swissranger SR4000 TOF sensor. The range images were captured at 25 frames per second at a resolution of 176×144 pixels. The ground-truth 3D joint locations of the subject were recorded by a commercial motion-capturing system.

Table 3.1 Total number of frames in each video sequence from the Stanford TOF Motion Capture Dataset.

Number of frames	Video index
100	0-5,8,9,14,16,18,
400	6,7,10-13,15,17,19,20-27

When evaluating the performance of HPE, frames with missing ground-truth 3D joint locations were ignored. The error metric, ζ, for each video sequence was defined as

$$\zeta = \frac{1}{N_f}\sum_{s=1}^{N_f}\frac{1}{N_s}\sum_{i=1}^{N_s}\left\|\mathbf{j}_{s,i}-\tilde{\mathbf{j}}_{s,i}\right\|_2, \tag{3.27}$$

where N_f is the number of frames of the video sequence for testing, N_s is the number of 3D joint locations measured by the motion-capturing system in the s-th frame, $\mathbf{j}_{s,i}$ is the ground-truth 3D location of the i-th joint in the s-th frame,$\tilde{\mathbf{j}}_{s,i}$ is the estimated 3D location of the i-th joint in the s-th frame and $\|\cdot\|_2$ is the Euclidean norm.

The dataset was divided into 20% for building the human-pose database, 30% for building the refinement database, 40% for building the action database and 10% for testing. To reduce the bias in dividing the dataset, the dataset was randomly divided 10 times with different random seeds in each trial. When the dataset was randomly divided, the proposed system was evaluated using 5-fold cross-validation. In the estimation step, the constant z in Eq. (3.10) was set to 0.1. In the redistribution step, the constant z_1 in Eq. (3.14) was set to 1. The threshold τ was calculated by subtracting the standard deviation of the values given by DTW from the mean of the values. When estimating the unnormalized base distribution, the constants u^f and u^a in Eq. (3.18) were set to 1. The non-evidential set in the kinematic model contained the vertices of lower arms and legs.

Three tests were performed to evaluate the proposed AMM, together with the 3D-point-cloud system that utilized both the proposed AMM and the kinematic model that was described in Section 3.3. In the first test, the proposed AMM was compared with a human-pose model that didn't utilize human actions. The comparison would verify the importance of using human actions in HPE. The performance of the proposed 3D-point-cloud system was then compared with the proposed AMM and the human-pose model that didn't utilize human actions. The comparison could show the importance of the kinematic model. In the second test, the number of actions being used in the proposed 3D-point-cloud system was considered as a parameter. The performance of the proposed 3D-point-cloud system was tested under different number of actions. This test could show the importance of using

multiple actions, instead of a single action, in the proposed 3D-point-cloud system. In the third test, the proposed 3D-point-cloud system was compared with some existing works that utilizes the same dataset, namely the Stanford TOF Motion Capture Dataset [70]. The test would show that the proposed 3D-point-cloud system achieved the state-of-the-art performance.

3.4.1 Evaluation of AMM

To evaluate the performance of using actions and the kinematic model in HPE, three approaches were implemented to estimate human poses. In the first approach (VISH), human poses were estimated by the nearest neighbors using VISH features. In the second approach (VISH+AMM), human poses were estimated by the proposed AMM using VISH features. In the third approach, human poses were estimated by the proposed 3D-point-cloud system.

Table 3.2 shows the errors and standard deviations of HPE incurred by the three approaches. When comparing VISH+AMM with VISH, the overall error and standard deviation in VISH+AMM were reduced compared with that in VISH. The reduction of the overall error and standard deviation were about 7.9% and 7.1% respectively. It showed that the result of action classification in the proposed AMM was useful in reducing errors of human-pose estimates. Using the kinematic model, the proposed 3D-point-cloud system further reduced the overall error and standard deviation of human-pose estimated by the proposed AMM. When comparing the proposed 3D-point-cloud system with VISH+AMM, the overall error and standard deviation in the proposed 3D-point-cloud system were reduced by 8.2% and 9.8%, respectively. When comparing the proposed 3D-point-cloud system with VISH, the overall error and standard deviation of the proposed 3D-point-cloud system were reduced by 15.5% and 16.2%, respectively. Thus, the experiment showed that the kinematic model could reduce the quantization error of human poses estimated by the proposed AMM.

Table 3.2 The errors (in meters) of HPE. Numbers on the left and in the parentheses are the errors and standard deviations of HPE, respectively.

Trial	VISH	VISH+AMM	Proposed System
1	0.0304 (0.0387)	0.0275 (0.0334)	0.0251 (0.0295)
2	0.0289 (0.0308)	0.0265 (0.0306)	0.0244 (0.0280)
3	0.0287 (0.0365)	0.0263 (0.0276)	0.0241 (0.0247)
4	0.0304 (0.0391)	0.0269 (0.0342)	0.0247 (0.0311)
5	0.0276 (0.0307)	0.0255 (0.0289)	0.0236 (0.0264)
6	0.0294 (0.0354)	0.0280 (0.0349)	0.0255 (0.0309)
7	0.0290 (0.0333)	0.0273 (0.0325)	0.0248 (0.0284)
8	0.0287 (0.0352)	0.0264 (0.0330)	0.0245 (0.0306)
9	0.0293 (0.0368)	0.0271 (0.0372)	0.0248 (0.0335)
10	0.0285 (0.0343)	0.0268 (0.0341)	0.0246 (0.0306)
Overall	**0.0291 (0.0351)**	**0.0268 (0.0326)**	**0.0246 (0.0294)**

3.4.2 Evaluation of using Multiple Actions in the Proposed 3D-Point-Cloud System

To test the efficiency of using multiple actions in the proposed 3D-point-cloud system, the proposed 3D-point-cloud system was modified such that the number of actions in the base distribution of the proposed AMM was considered as a parameter for HPE. We denoted the number of actions by N_a. The number of actions N_a was varied from 1 (one action) to 28, which was the total number of actions in the dataset. As some actions may occur more likely than the other, using less likely actions may not reduce the error of human-pose estimates. Thus, the 28 actions were first sorted in a list in descending order according to the pmf that was computed based on bagging in action classification. Then, N_a actions were selected from the first N_a actions in the sorted list. As a result, more likely actions would be selected.

Figure 3.11 shows the change of the overall error and standard deviation of human pose estimated by the proposed 3D-point-cloud system. In the figure, when the number of actions N_a increased initially, the overall error and standard deviation incurred by the proposed 3D-point-cloud system were decreased. The change showed that multiple actions should be considered in the proposed 3D-point-cloud system to yield a better representation of human poses and hence increase the accuracy/precision of human-pose estimates.

However, as the number of actions N_a further increased, the overall error and standard deviation stopped to decrease. It was because the newly included actions in the proposed 3D-point-cloud system were less likely to be occurred. Thus, the weights that were used to combine the base distributions in AMM were small, and those action did not reduce the errors.

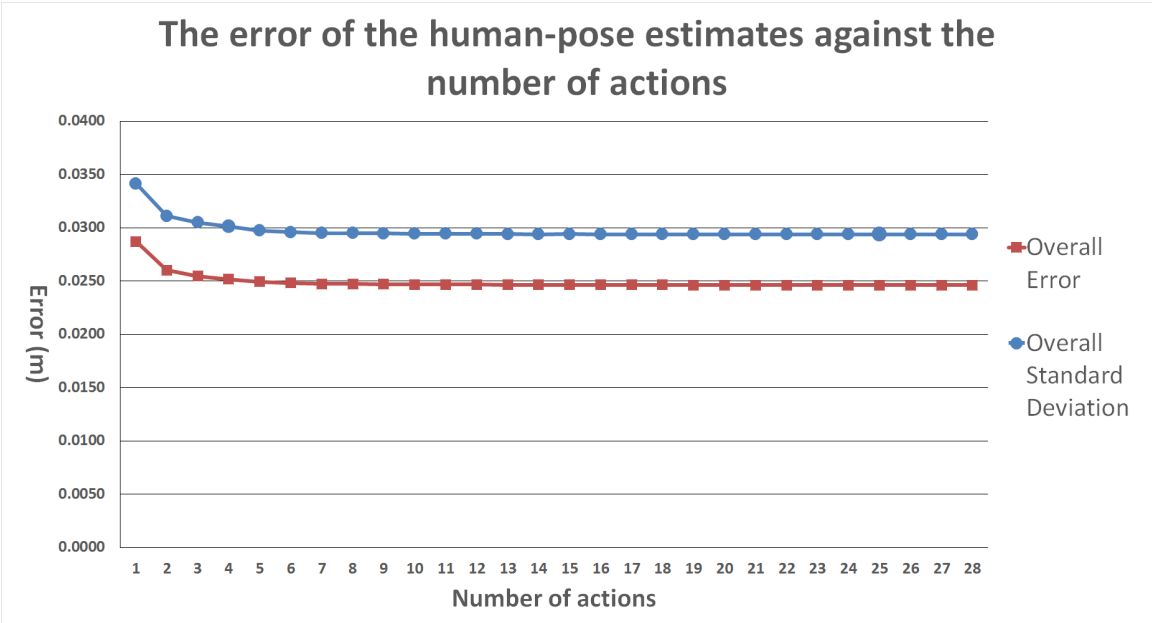


Fig. 3.11. The changes of the overall error and standard deviation of HPE using different numbers of actions.

3.4.3 Comparison between the Proposed 3D-Point-Cloud System and Existing Works

The overall error and standard deviation of human poses estimated by the proposed 3D-point-cloud system were compared with the errors reported in some existing works that utilized the Stanford TOF Motion Capture Dataset [70]. Table 3.3 shows the overall errors and standard deviations incurred in those approaches. Among all the approaches, the proposed 3D-point-cloud system incurred the lowest overall error and standard deviation. The result showed that the result from action classification and the kinematic model could reduce the errors in HPE.

Note that the overall standard deviation incurred the proposed 3D-point-cloud system was larger than the overall error because the human poses in the human-pose database for estimating the base distributions for different actions could not fully describe the human poses in the test dataset. Thus, for some human poses estimated by the proposed 3D-point-cloud system, the errors were larger than the errors of other human-pose estimates.

Table 3.3 The overall errors and standard deviations of HPE using the Stanford TOF Motion Capture Dataset [70].

	Overall Error (m)	Overall Standard Deviation (m)
HC and EP Method [70]	0.1	N.A.
Data-driven Hybrid Method [49]	0.0618	0.0424
Exemplar Method [50]	0.038	N.A.
Proposed 3D-Point-Cloud System	0.0246	0.0294

3.5 Conclusions

In this chapter, we have defined humna-pose models as a mapping from a feature of an observation of a human to a human pose. We have also described the main component of using human-pose models. The main component is to utilize prior knowledge in order to increase the accuracy of human-pose estimates. This is particularly important when observations are not reliable. Then, we have exploited human actions as a form of prior knowledge in HPE. Specifically, we have used human actions to discover low-dimensional manifolds in human-pose space. The main idea of using human actions has been formulated in our proposed AMM. A low-dimensional manifold for each action was described by a base distribution in the proposed AMM. A base distribution for an action is estimated through the two steps, namely the estimation and redistribution steps. The estimation step calculates the Euclidean distance between a VISH-feature input and VISH features extracted from observations of human poses in the human-pose database. It produces a probability distribution of using a human pose in the human-pose database as a human-pose estimate as an output for each action. The redistribution step weights the probability distributions among all actions from the estimation step according to the likelihood of observing an VISH-feature

input being occurred in different actions. The likelihood is estimated by bagging. The outputs from the two steps are then combined to form a probability distribution that measures how likely a human pose being a human-pose estimate. The most probable human pose becomes the human pose estimated by the proposed AMM.

As human poses estimated by the proposed AMM are in discrete space, a kinematic model is used to model the spatial relationship of body parts in continuous space to reduce the quantization error in the proposed AMM. However, modeling the spatial relationship between every pair of body parts is intractable. Thus, we utilized the kinematic chain that only encodes the spatial relationship of adjacent body parts. By combining the proposed AMM and the kinematic model, a 3D-point-cloud system is formed.

Experimental results showed that using human actions in HPE could increase the accuracy and precision of human-pose estimates. By using multiple actions in the proposed 3D-point-cloud system, the error of human-pose estimates was decreased more than the error of human poses estimated by using a single action in the proposed 3D-point-cloud system. The number of actions being considered in base distributions, which was defined in the proposed AMM, could be determined adaptively by the pmf that is computed based on action classification. The overall error and standard deviation of the proposed 3D-point-cloud system were reduced by 15.5% and 16.2% respectively compared with a human-pose model without using actions. The proposed 3D-point-cloud system achieved the state-of-the-art performance.

In the next chapter, we will extend the proposed AMM by automatically learning the prior knowledge from training data that trains a human-pose model. Specifically, we would introduce a mechanism that designs the weights and base distributions in the proposed AMM from training data. The main advantage of using automatic design is to eliminate the laborious step of manual definitions in the proposed AMM. In addition, both the weights and base distributions could be adapted automatically from training data. We will compare the performance of human-pose model with and without the automatic design to verify the advantage of using automatic design.

4. NEURAL-NETWORK-BASED MODELING

4.1 Introduction

Since there are uncertainties from different sources such as noise and perceptual limitations, probabilistic models are used to accommodate uncertainties. In particular, Bayesian networks [114] [89] are commonly used because they naturally represent causal, evidential and intercausal relationships, which are indispensable to human understanding [115]. As we discuss in Chapter 3, we have built an action-mixture model (AMM) [116] [117] to map a VISH feature (in Chapter 2) to a human pose. AMM can be considered as a Bayesian network that models the level of uncertainty of human actions and a relationship between human actions and HPE. Using human actions, AMM achieves the state-of-the-art performance in HPE. By building a structure of a Bayesian network, humans can express their beliefs about those relationships. Other examples of Bayesian networks include a kinematic model [70] and a motion-exemplar tracking model [40] for HPE (HPE).

When building Bayesian networks, designing appropriate factors, which are conditional probability distributions, is important [110]. This is because factors can change not only probability distributions that model the level of uncertainty, but also relationships encoded by the structures of Bayesian networks. For example, a Gaussian-mixture model (GMM) [118] and an independent component analysis (ICA) [119] share the same structure, but relationships among random variables are different, because factors in a GMM are multivariate Gaussian functions, while factors in an ICA are univariate Gaussian and Laplace functions. Typically, factors are designed by humans. The designing process is task-oriented and laborious. For instance, factors in AMM were designed according to our proposed similarity measures (more details in Section 3.2). Factors may be designed to simplify learning and inference computations. For example, in the motion-exemplar tracking model [40], factors were set manually to be Gaussian functions to approximate

dependencies between human joints so that the process of estimating human-joint positions was simplified. Although, in the literature [7] [8], it is common to manually design factors, they could be changed when the underlying probability distributions change. For example, a human-pose distribution is changed when an environment changes from a library to a gymnasium because running postures are observed more often at a gymnasium. This motivates us to investigate a universal method for designing and modeling factors automatically so that they can be adapted to time-varying probability distributions in different situations. In particular, we are interested in extending our previous work AMM on HPE, so that factors can be designed or learned automatically. We realize that artificial neural networks (NNs) can adapt to time-varying probability distributions based on training data. Thus, we utilize the learning capability of NNs to learn, design and realize factors of AMM to achieve better HPE.

Neural networks are inspired by the human brain. They consist of a number of processing elements called neurons that are interconnected. Depending on neuron connections and activation functions, various types of neural networks such as perceptron networks and multilayer perceptron networks are created. Traditionally, hidden neurons have been used as computational units for mapping between input and output. Recently, in the research field of deep learning [86] [87], they are also interpreted as input features [120] [121] and filters [60] [122]. In addition, hidden neurons in some neural networks [94] [96] [98] are considered as hidden random variables that change the nature of neural networks from deterministic to probabilistic.

Neural networks with deep network architectures have been shown to achieve the state-of-the-art performance in HPE [47, 62–64]. They can be generally categorized into two groups according to their usages. In the first group, NNs are used as function approximators that map observations such as images of humans or body parts to human poses. For example, Toshev and Szegedy [63] formulated the HPE problem as a regression problem that mapped an image of a human to a normalized human pose. A convolutional neural network (CNN) was applied on an image to estimate human-joint positions. Then, a CNN was built for each human joint and was applied on an image region centered at a human-

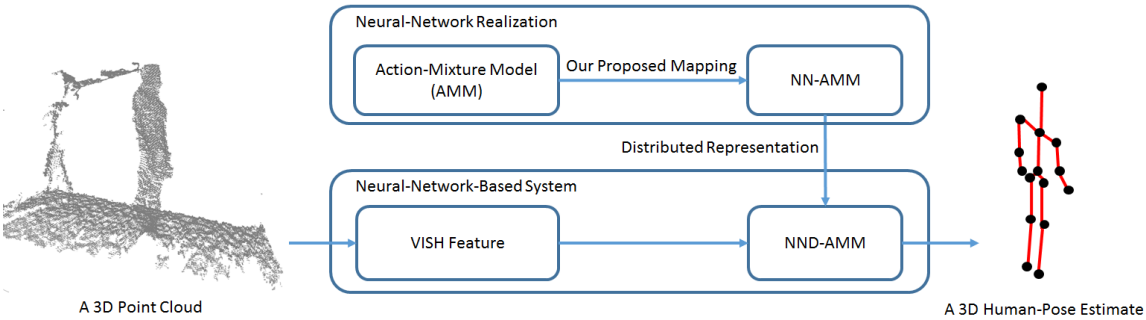


Fig. 4.1. Neural-network-based HPE system. The proposed system takes a 3D point cloud as input and represents it with a VISH feature. Then, the VISH feature is passed to NND-AMM, which is a neural network built by applying our proposed mapping on AMM and the concept of distributed representation. NND-AMM finally estimates a 3D human pose as output.

joint estimate to refine the estimate. In the second group, parts of NNs are used to extract features for HPE. Typically, hidden layers of NNs are used to extract features from observations through layer-wise training [86] [87]. As hidden layers are closer to the output layer of a NN, features become more abstract. The output layer then uses extracted features to estimate human poses. For example, Ouyang *et al.* [65] proposed several NNs to extract image features. Once image features were extracted, human poses were estimated by a NN with hidden neurons used as computational units. Although these approaches are promising, they cannot provide insights into their network architectures. Thus, it is hard to change the architectures to adapt to a 3D-point-cloud input, which is an input to AMM, and model factors of AMM. We will utilize the structure and semantic meaning of random variables in AMM, and transfer them to a NN. In particular, we will use a feedforward NN because of the static nature of Bayesian networks. Based on AMM, the architecture of a feedforward NN could be designed and interpreted systematically.

In this chapter, we first propose a mapping that converts a Bayesian network into a feedforward NN. AMM is extended by considering it as a Bayesian network and converting it into a feedforward NN, denoted by NN-AMM, through the proposed mapping. The advantages of using the proposed mapping compared with our previous work are that designing factors is automatic based on training data for training a feedforward NN, and factors can

be adapted to different situations. Semantic meaning in AMM can also be transferred to the feedforward NN so that neurons or layers of neurons have semantics. Using the proposed mapping, a NN-based HPE system is built. The proposed system takes a 3D point cloud as input and computes a 3D human-pose estimate (see Figure 4.1). In the proposed system, a viewpoint-and-shape-feature-histogram (VISH) feature [104] (in Chapter 2) is extracted from a 3D point cloud. Based on the concept of distributed representation, NN-AMM is modified to form a scalable feedforward NN, denoted by NND-AMM. Two variants of VISH features are generated to evaluate the adaptability of the three models, namely AMM, NN-AMM, and NND-AMM. The first variant is produced by occluding a part of a 3D-point-cloud input. The second variant is produced by reconstructing VISH features from the first variant using a linear model with additive Gaussian noise. The linear model is used because we will show that, as a special case of Bayesian networks, it is equivalent to a feedforward NN under certain types of inference. Experiments were conducted to compare the performance of the three models based on their accuracy of human-pose estimates and their adaptability to the two variants of VISH features.

The structure of this chapter is as follows. Section 4.2 describes two steps, namely structure identification and parameter learning, in the proposed mapping. Section 4.3 illustrates the application of our proposed mapping on AMM and describes the modification in NN-AMM to create NND-AMM. Section 4.4 presents the two variants of VISH features. Experiments were conducted in Section 4.5 to test the performance of the proposed mapping and the proposed system. Section 4.6 summarizes our results.

4.2 Mapping Mechanism

A Bayesian network is a directed acyclic graph (DAG), denoted by $\mathscr{G}$, in which nodes are random variables and absence of edges represents conditional independence assumptions on random variables. It defines a family of probability distributions $p(X_1,\ldots,X_\mathfrak{n})$ that can be expressed as

$$p(X_1,\ldots,X_\mathfrak{n}) = \Pi_{i=1}^{\mathfrak{n}} p(X_i|Pa_{X_i}^{\mathscr{G}}), \tag{4.1}$$

where X_i's are random variables, $\mathfrak{n}$ is the number of random variables, $Pa^{\mathscr{G}}_{X_i}$ are parents of X_i in $\mathscr{G}$, and conditional probability distributions $p(X_i|Pa^{\mathscr{G}}_{X_i})$ are called factors. Factors are interpreted as local relationships, which represent causal, evidential and intercausal reasonings, among random variables.

To map a Bayesian network to a NN, we first represent a NN using a graph, denoted as $\mathscr{G}^n$. A graph $\mathscr{G}^n$ should encapsulate the structure and synaptic weights of a NN. We define $\mathscr{V}^n$ as a set of nodes that represent neurons in a NN. The i-th node is denoted as v^n_i. It is defined by the corresponding neuron type (input, hidden, or output) and activation function, which are denoted as $\mathfrak{l}_i$ and $\mathfrak{a}_i$, respectively. We define $\mathscr{E}^n$ as a set of edges that represent synaptic connections and weights in a NN. The i-th edge is denoted as e^n_i. It is defined by two nodes, denoted as $\mathfrak{s}_i$ and $\mathfrak{t}_i$ that the i-th edge connects, and its synaptic weight, denoted as $\mathfrak{w}_i$. Since we can define a graph $\mathscr{G}^n$ by defining both $\mathscr{V}^n$ and $\mathscr{E}^n$, we write $\mathscr{G}^n = (\mathscr{V}^n, \mathscr{E}^n)$, where $\mathscr{V}^n = \{v^n_1, \ldots, v^n_{|\mathscr{V}^n|}\}$, $v^n_i = (\mathfrak{l}_i, \mathfrak{a}_i)$, $\mathscr{E}^n = \{e^n_1, \ldots, e^n_{|\mathscr{E}^n|}\}$, and $e^n_i = (\mathfrak{s}_i, \mathfrak{t}_i, \mathfrak{w}_i)$.

Using the graphical representation of a NN, the realization process involves defining a graph $\mathscr{G}^n$ according to a Bayesian network. The process is divided into two parts: structure identification and parameter learning. In the structure identification, we identify $\mathscr{V}^n$ and $\mathscr{E}^n$ except the synaptic weight $\mathfrak{w}_i$ for every $e^n_i \in \mathscr{E}^n$. As we will show later, a NN structure that is created by the identification process is feedforward. During the identification process, we can interpret a feedforward NN as a collection of interconnected feedforward NNs with semantic meaning. Details can be found in Section 4.2.1. In the parameter learning, we learn $\mathfrak{w}_i$ for every $e^n_i \in \mathscr{E}^n$. We present a part-based approach to learn parameters in a feedforward NN by decomposing it into semantic parts. Details can be found in Section 4.2.2.

4.2.1 Structure Identification

In order to realize a Bayesian network by a feedforward NN, the structure should be designed such that observed variables in a Bayesian network correspond to neurons in the input layer of a feedforward NN (input neurons), hidden variables correspond to neurons in

the hidden layers (hidden neurons), and target variables correspond to neurons in the output layer (output neurons). Synaptic connections should represent local relationships (factors) in a Bayesian network.

We propose a bottom-up approach to build a feedforward NN by constructing a group of feedforward NNs called modules. First, we represent a relationship among random variables in a factor by a module. As a consequence, each module represents only a relationship among random variables in one factor but not others. Thus, conditional independence assumptions in a Bayesian network are preserved. Figures 4.2 and 4.3 show examples of causal, evidential and intercausal relationships, and their modules. Once the structure of each module is defined, a NN is built by merging common neurons from different modules.

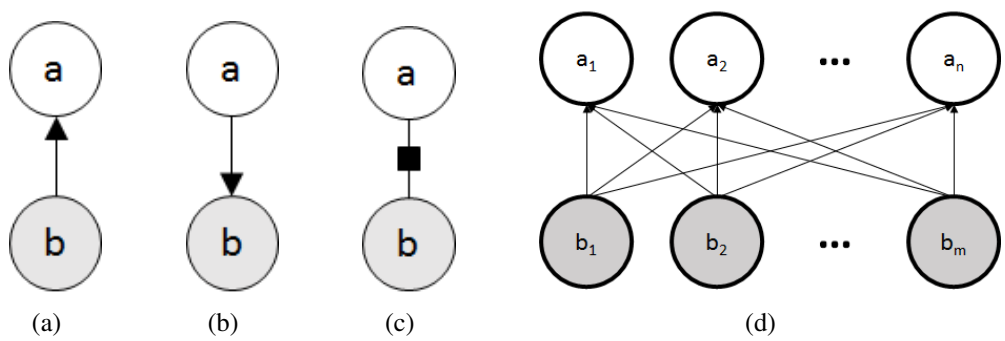


Fig. 4.2. Examples of causal and evidential relationships between an observed random variable b (shaded) and a target random variable a. (a) Causal relationship between random variables a and b. (b) Evidential relationship between random variables a and b. (c) Factor graph of the causal or evidential relationship. (d) A neural network representing the causal or evidential relationship. $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$ are possible values of random variables a and b when the random variables are discrete. When the random variables are continuous, one neuron is used with its value equal to the value of each random variable. Hidden layers can be added in the neural network if necessary.

We create $\mathscr{V}^n$ and $(\mathfrak{s}_i, \mathfrak{t}_i)$ in $\mathscr{E}^n$ by scanning each factor in a factor graph and identifying its corresponding subgraph in a DAG of a Bayesian network. If a random variable in a factor is discrete, we create one neuron in $\mathscr{V}^n$ for each possible value. The activation function of a neuron should be the softmax function because it represents a probability. If a random variable in a factor is continuous, we create one neuron in $\mathscr{V}^n$ to represent a real value. The activation function of a neuron depends on the range of a random variable.

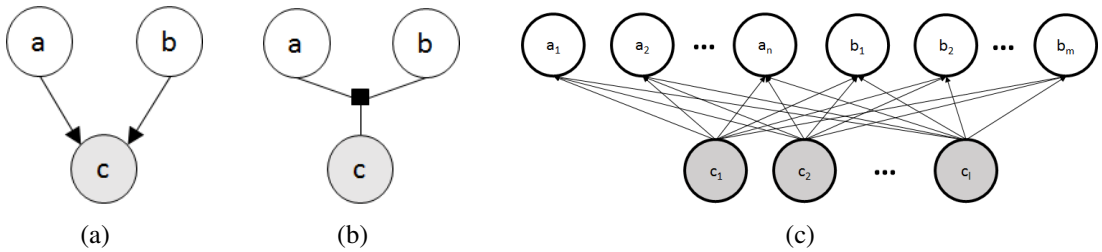


Fig. 4.3. Example of an intercausal relationship between target random variables a and b. An observed random variable c is shaded. (a) Intercausal relationship between target random variables a and b. (b) Factor graph of the intercausal relationship. (c) A neural network representing the intercausal relationship. $a_1, \ldots, a_n$, $b_1, \ldots, b_m$, and $c_1, \ldots, c_l$ are possible values of random variables a, b, and c when the random variables are discrete. When the random variables are continuous, one neuron is used with its value equal to the value of each random variable. Hidden layers can be added in the neural network if necessary.

After neurons are defined, we define synaptic connections between them by considering connections in a subgraph of each factor in a Bayesian network. There are six types of connections. They are connections between (1) observed variables, (2) an observed variable and a hidden variable, (3) an observed variable and a target variable, (4) hidden variables, (5) a hidden variable and a target variable, and (6) target variables. For the first type, the synaptic connection between input neurons is not created because neurons are observed. For the second type (the third type), we always create a synaptic connection from an input neuron to a hidden neuron (an output neuron) in order to represent the causal or evidential relationship between an observed and a hidden variables (a target variable). For the fourth type (the sixth type), we create synaptic connections between hidden (output) neurons according to connections between hidden (target) variables in order to avoid creating feedback connections in a module. For the fifth type, we always create a synaptic connection from a hidden neuron to an output neurons to represent the causal or evidential relationship between a hidden variable and a target variable. When a module is created, hidden layers can be added in order to represent a complex relationship. Algorithm 1 summarizes the proposed bottom-up approach. A NN that represents a Bayesian network is created by merging all modules together.

We will prove by contradiction that a NN structure created by the proposed bottom-up approach is feedforward. Assume there is a feedback connection in a NN. Consider the shortest path that contains the feedback connection. The shortest path does not contain input neurons because, by construction, there is no synaptic connection entering input neurons. The shortest path does not contain both hidden and output neurons because, by construction, there is no synaptic connections from output to hidden neurons. The shortest path does not contain hidden (output) neurons because synaptic connections are established according to a DAG in a Bayesian network. Thus, the shortest path does not contain any neurons. It contradicts the assumption.

Algorithm 1 The proposed bottom-up approach for structure identification

INPUT: A DAG and a factor graph of a Bayesian network
OUTPUT: $\mathscr{G}^n = (\mathscr{V}^n, \mathscr{E}^n)$ except the synaptic weight $\mathfrak{w}_i$ for every $e_i^n \in \mathscr{E}^n$

```
𝒱^n = ℰ^n = ∅
for each factor in a factor graph of a Bayesian network do
  Identify the subgraph of a DAG that corresponds to a factor
  for each random variable in the subgraph do
    Add neurons to 𝒱^n according to the description in Section 4.2.1
  end for
  for each edge in the subgraph do
    Add synaptic connections to ℰ^n according to the description in Section 4.2.1
  end for
  if a relationship of a factor is complex then
    Add hidden neurons to 𝒱^n
    Add synaptic connections to ℰ^n
  end if
end for
```

4.2.2 Parameter Learning

In the previous section, a bottom-up approach is presented to identify $\mathscr{V}^n$ and $\mathscr{E}^n$ except synaptic weights. In this section, we focus on learning synaptic weights; that is, the weight $\mathfrak{w}_i$ of a synaptic connection $e_i^n \in \mathscr{E}^n$, where $i = 1, 2, \ldots, |\mathscr{E}^n|$. We utilize the backpropagation algorithm [6] to learn synaptic weights. We assume that training data contains desired

values of input and output neurons. Desired values of hidden neurons may or may not be available in training data. Since a feedforward NN, in general, may have many layers, applying the backpropagation algorithm directly to learn synaptic weights may suffer from the vanishing-gradient problem [5]. Based on the concept of layer-wise training [86] [87], we propose a part-based approach to learn synaptic weights by creating parts of a feedforward NN and training them sequentially using the backpropagation algorithm. Since a part is a feedforward NN that may consist of several layers, the proposed approach is considered as an extension of layer-wise training procedures. This extension allows each part to have semantic meaning.

A part is defined by a subset of $\mathscr{E}^n$ and neurons that are connected by each edge in a subset of $\mathscr{E}^n$. Each part in a sequence should contain synaptic connections in the previous part so that synaptic weights of synaptic connections in the previous part may be used as initial estimates for training the current part. Nodes in a part are identified by first traversing nodes in $\mathscr{V}^n$ from each node in the previous part until a node that contains a desired value from the training data is reached. The set of nodes in a part is then defined by the union of the set of nodes in the previous part and the set of nodes that are on traversing paths. The set of edges in a part is defined by a subset of $\mathscr{E}^n$ in which each edge connects nodes in the set of nodes in a part. The previous part of the first part is defined as a set of input neurons in $\mathscr{V}^n$ without any synaptic connections. Algorithm 2 summarizes the proposed part-based approach.

4.2.3 Illustration of the Proposed Mapping

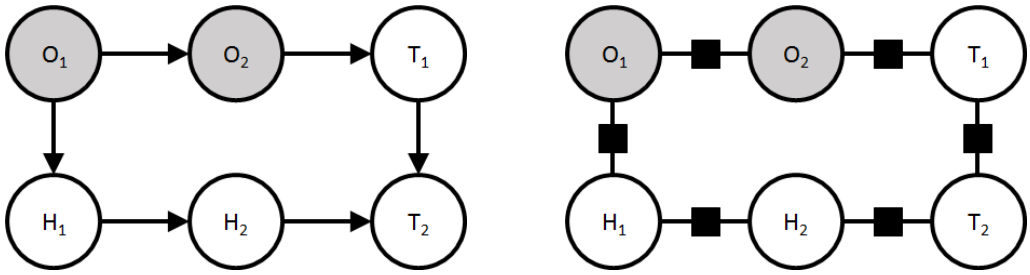
In this section, we will construct a Bayesian network and show the process of applying the proposed mapping on a Bayesian network. As the proposed mapping performs differently depending on the type of connections that are described in Section 4.2.1, we create a simple Bayesian network that contains all types of connections. The Bayesian network is shown in Figure 4.4. In the Bayesian network, observed variables are O_1 and O_2. Hidden variables are H_1 and H_2. Target variables are T_1 and T_2. In order to keep the illustration

Algorithm 2 The proposed part-based approach for parameter learning

INPUT: The structure $\mathscr{G}^n = (\mathscr{V}^n, \mathscr{E}^n)$ of a neural network
OUTPUT: Synaptic weights $\mathfrak{w}_i$ of the i-th edge in $\mathscr{E}^n$, where $i = 1, 2, \ldots, |\mathscr{E}^n|$

$Part_0 = (\mathscr{V}_0^n = \{v_i^n : v_i^n \in \mathscr{V}^n \wedge \mathfrak{l}_i = \text{input}\}, \mathscr{E}_0^n = \emptyset)$
$idx = 0$
repeat
  $idx = idx + 1$
  Create $\mathscr{V}_{idx}^n$ according to the description in Section 4.2.2
  $\mathscr{E}_{idx}^n = \{e_i^n : e_i^n \in \mathscr{E}^n \wedge \mathfrak{s}_i \in \mathscr{V}_{idx}^n \wedge \mathfrak{t}_i \in \mathscr{V}_{idx}^n\}$
  $Part_{idx} = (\mathscr{V}_{idx}^n, \mathscr{E}_{idx}^n)$
until $Part_{idx} = \mathscr{G}^n$
train $Part_0$ using random initial values of synaptic weights
for $i = 1$ to idx **do**
  train $Part_i$ using initial values of synaptic weights in $Part_{i-1}$
end for

straightforward, we assume that all random variables in the Bayesian network are binary. We also do not add additional hidden layers in the NN that is created by the proposed mapping.



(a) DAG. (b) Factor graph.

Fig. 4.4. The Bayesian network that is used for illustrating the proposed mapping. Observed variables are shaded.

In structure identification, we scan each factor in the factor graph as shown in Figure 4.4(b) and create a module for each factor. Since there are six factors, there are six modules. As each random variable is binary, two neurons are created to represent two possible values (zero and one) of each random variable. Each neuron is denoted by the name of a random variable with a superscript indicating the value of a random variable. For example, the neuron that represents the zero value of the random variable O_1 is denoted as O_1^0. Based on the Algorithm 1, modules are created and shown in Figure 4.5. In the figure, each factor is shown on the left and the corresponding module is shown on the right. Note that when we scan the factor between observed variables O_1 and O_2, we do not create any synaptic connections between input neurons because no inference is performed on observed variables. A NN that represents the Bayesian network is created by merging all modules together, and is shown in Figure 4.6.

In parameter learning, we consider that the training set contains desired values of input and output neurons except hidden neurons. The first part is created by traversing the NN from each input neuron until a neuron that contains a desired value is reached. Figure 4.7 shows the first part. The second part is created by traversing the NN from every neuron in the first part until a neuron that contains a desired value is reached. Figure 4.8 shows the

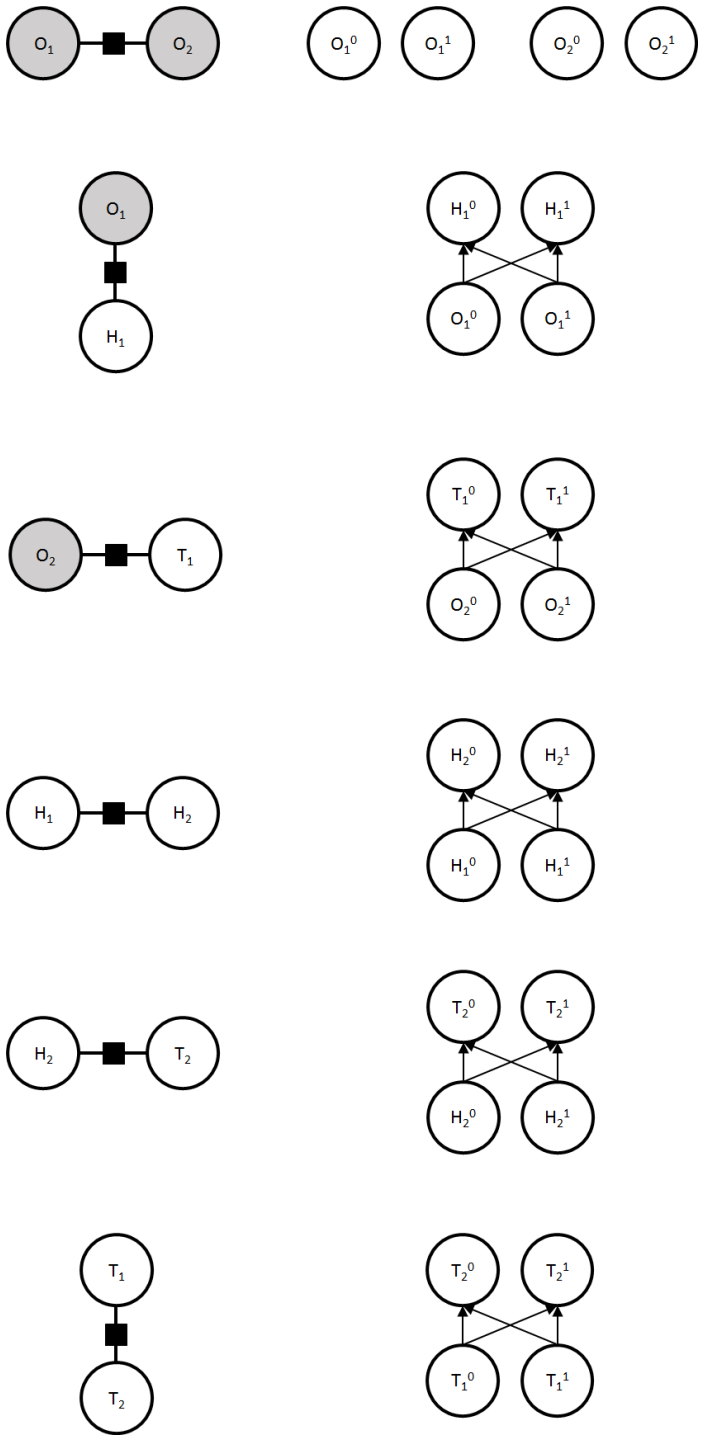


Fig. 4.5. Factors from the Bayesian network in Figure 4.4 and their corresponding modules that are created by the proposed mapping. In each row, a factor is shown on the left and its corresponding module is shown on the right.

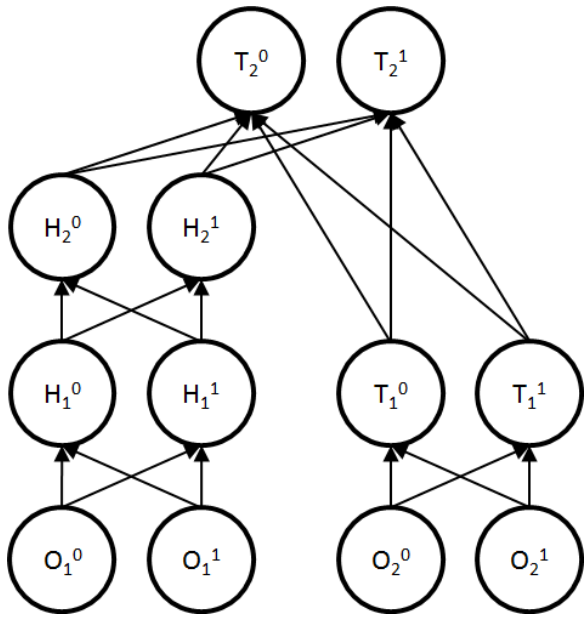


Fig. 4.6. The NN that is created by merging all modules in Figure 4.5.

second part. As the second part is the NN created in structure identification, no further part is created. Parameters in the NN are learned in two steps. First, parameters in the first part are learned. Then, they are used as initial values to train the second part.

This illustration not only shows the application of the proposed mapping on a general Bayesian network, but also demonstrates the main advantage of using the proposed mapping. The main advantage is that, instead of executing the laborious process of manually designing the six factors, we can utilize NNs to design factors automatically from training data.

4.3 Application of the Proposed Mapping on AMM

We will first describe AMM and apply our proposed mapping on AMM to create a feedforward NN called NN-AMM. Then, as the number of possible human poses in NN-AMM could be large, we will modify NN-AMM based on the concept of distributed representation to build a scalable feedforward NN called NND-AMM in the proposed NN-based HPE system.

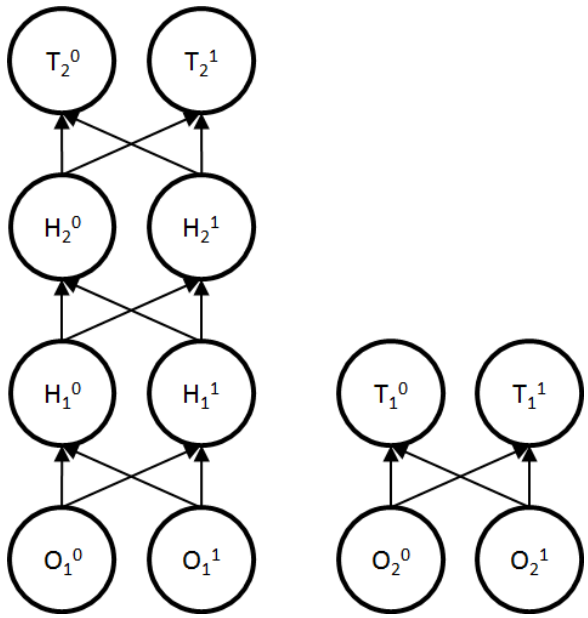


Fig. 4.7. The first part of a part sequence in the process of parameter learning. It is created by traversing every input neuron.

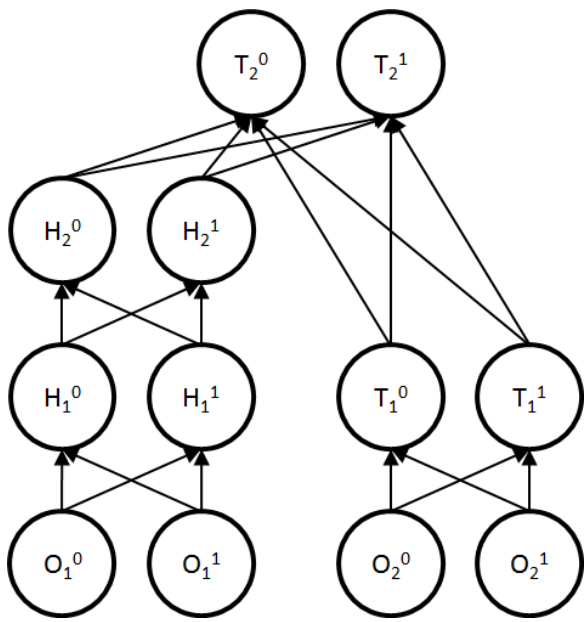


Fig. 4.8. The second part of a part sequence in the process of parameter learning. It is created by traversing every neuron in the first part in Figure 4.7.

4.3.1 Action-Mixture Model

An action-mixture model (AMM) is a Bayesian network. It is designed for HPE. In AMM, we design factors manually. As described in Chapter 3, factors of AMM are $p(a|\mathbf{x})$

and $p(y|a,\mathbf{x})$. The factor $p(a|\mathbf{x})$ is a probability mass function (pmf) of a VISH feature $\mathbf{x}$ belonging to an action a. It is commonly used to classify actions, and we use a typical approach called the bootstrap aggregating algorithm (i.e., bagging) [112] to train an action classifier. The factor $p(y|a,\mathbf{x})$ is computed in two steps: estimation and redistribution steps. The estimation step measures the likelihood of human poses spatially. The likelihood is defined based on the Euclidean distance between VISH features such that if the distance is small, human poses are close spatially. In addition, we increase the likelihood of all key poses (possible poses) from an action a. It is because a human pose corresponding to a VISH feature $\mathbf{x}$ should be similar to some key poses from an action a. The redistribution step measures the likelihood of human poses based on the frequency of actions. We define the frequency of actions as the portion of time a human pose in each action. If the frequency of an action is higher than other actions, that action is expected to be observed more often, and thus the likelihood of human poses in that action is higher. In the redistribution step, we consider that human poses could be similar in different actions. Thus, observing a human pose from an action changes not only the frequency of that action, but also the frequency of some other actions. This phenomenon is modeled using a continuous-time Markov chain [116] [117].

4.3.2 Neural-Network-Based Realization of Action-Mixture Model

When designing factors manually as described in Section 4.3.1, it is laborious to quantify the spatial relationship about human poses and the frequency of actions. Also, manually designed factors may not represent underlying probability distributions and may not be applicable to different situations such as different input features. Thus, in this section, we will construct a feedforward NN, denoted by NN-AMM, based on our proposed mapping procedure to realize AMM. First, we identify the NN-AMM structure. Based on the proposed bottom-up approach in Section 4.2.1, we build a module for each factor, and merge modules together to build the NN-AMM structure.

To build a module of the factor $p(a|\mathbf{x})$ in AMM, we assign an observed variable $\mathbf{X}$ to the input layer of a module, and a hidden variable A to the output layer of a module. Figure 4.9(a) shows the module. To build a module of the factor $p(y|a,\mathbf{x})$ in AMM, we assign an observed variable $\mathbf{X}$ to the input layer of a module, and add synaptic connections from an observed variable to a hidden variable A and a target variable Y. In addition, we add a synaptic connection from a hidden variable to a target variable to represent their causal relationship. Figure 4.9(b) shows the module. The two modules are then merged together to form the structure of NN-AMM, which is the same as the module in Figure 4.9(b). Note that the accuracy of HPE in NN-AMM is similar to the accuracy in AMM, whose factors are manually designed. Thus, we do not include any additional hidden neurons or hidden layers in the two modules. Details about the comparison between AMM and NN-AMM can be found in Section 4.5.

Synaptic weights in NN-AMM are learned by the proposed part-based approach in Section 4.2.2. As defining parts depends on training data, we first describe our training data. Our training data is the same as the one used in testing AMM in [117] because we would like to compare the performance of AMM and NN-AMM. Each training sample contains desired values of a VISH feature of a human, a human action and a human pose. The first part in a part sequence is defined by scanning the layer directly connected to the input layer of NN-AMM. The layer contains the action and pose variables. As both desired values are available in training data, those variables, together with a VISH variable and its synaptic connections, are selected to form a part. The part is shown in Figure 4.10(a). This part has the meaning of action recognition, and pose estimation without using action information. The second part is defined by scanning the next layer in NN-AMM. It contains a synaptic connection from an action variable to a pose variable, in addition to the previous part. Thus, the second part corresponds to NN-AMM. Figure 4.10(b) shows the second part. Once a part sequence is defined, the first part is trained by the backpropagation algorithm followed by the second part.

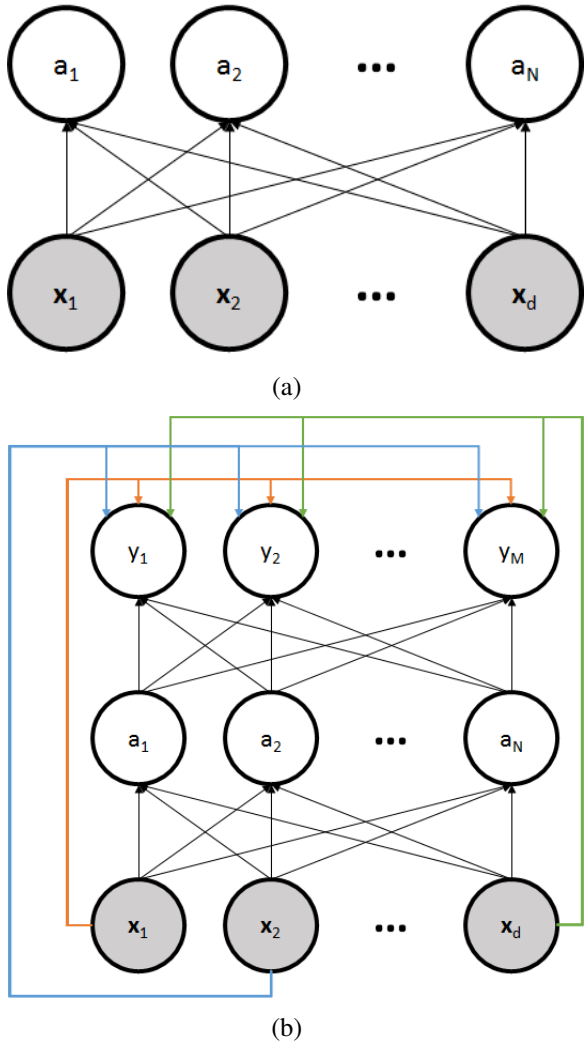
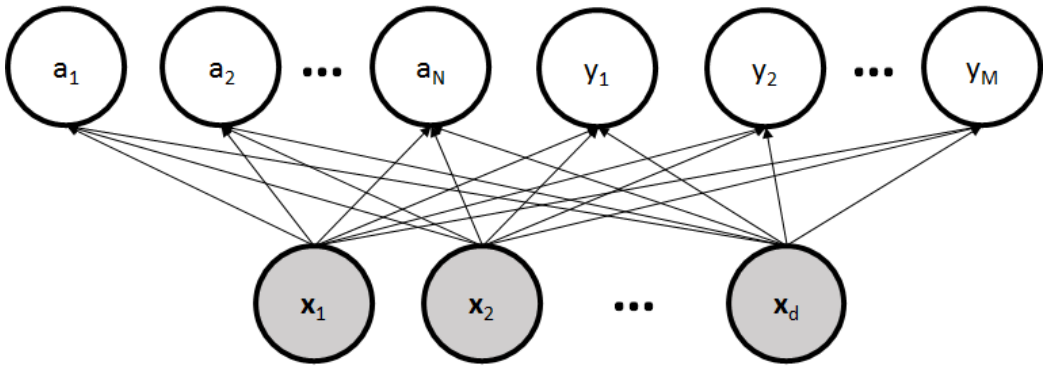


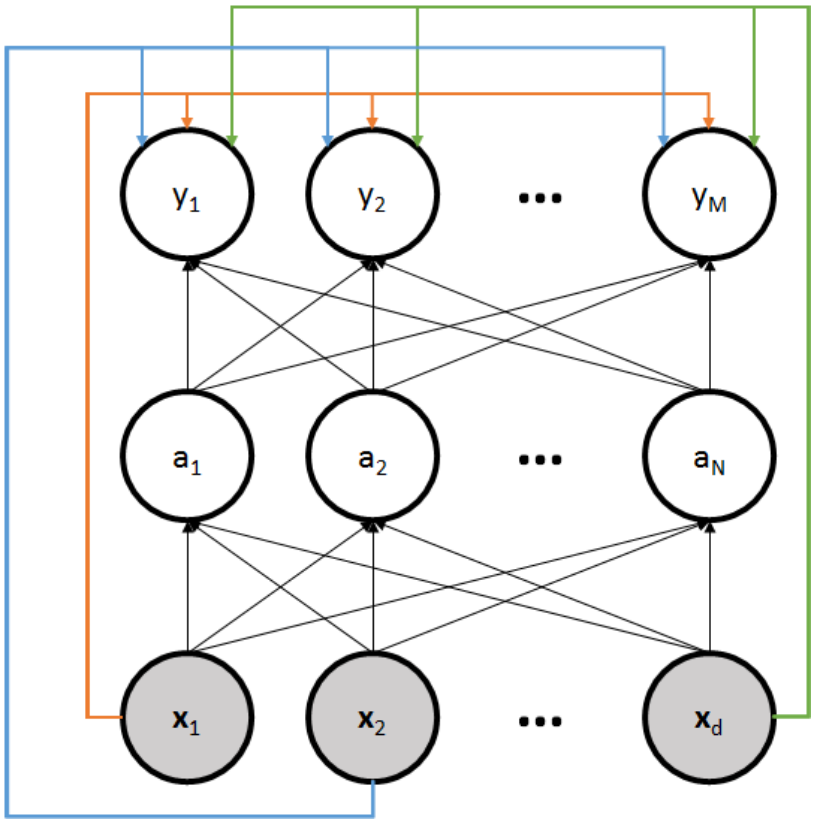
Fig. 4.9. Modules of factors (a) $p(a|\mathbf{x})$ and (b) $p(y|a,\mathbf{x})$. Observed variables are shaded. $\mathbf{x}_1,\cdots,\mathbf{x}_D$ are elements in a VISH feature $\mathbf{x}$. D is the dimension of a VISH feature. $a_1,\cdots,a_N$ and $y_1,\cdots,y_M$ are possible values of the random variables A and Y, respectively. N and M are the number of actions and key poses, respectively. In (b), the connection from the input layer to the output layer indicates that all neurons in the input layer are connected to every neuron in the output layer. The module in (b) is also the neural network NN-AMM that is built by applying the proposed mapping on AMM.

4.3.3 Scalable Neural-Network-Based Realization

With semantic meaning in NN-AMM, we recognize that each neuron y_i, where $1 \leq i \leq M$, represents a key pose. Thus, NN-AMM is not scalable because the number of



(a)



(b)

Fig. 4.10. (a) First part and (b) second part in a part sequence used in the proposed part-based approach. Notations are the same as those in Figure 4.9.

key poses could be large. Utilizing the concept of distributed representation, we can reduce the number of neurons (key poses) in NN-AMM. Instead of confining a human-pose estimate to be one key pose, we consider a human-pose estimate to be a linear combina-

tion of key poses. Thus, we can eliminate key poses that are combination of other key poses. The process of creating and eliminating key poses can be formulated as dictionary learning [121] [123] [124]. In this work, we use the hierarchical-clustering method [125], which is a common technique in dictionary learning, by considering cluster prototypes as key poses. A feedforward NN that is modified based on NN-AMM and the concept of distributed representation is denoted by NND-AMM. Figure 4.11 shows the NND-AMM structure.

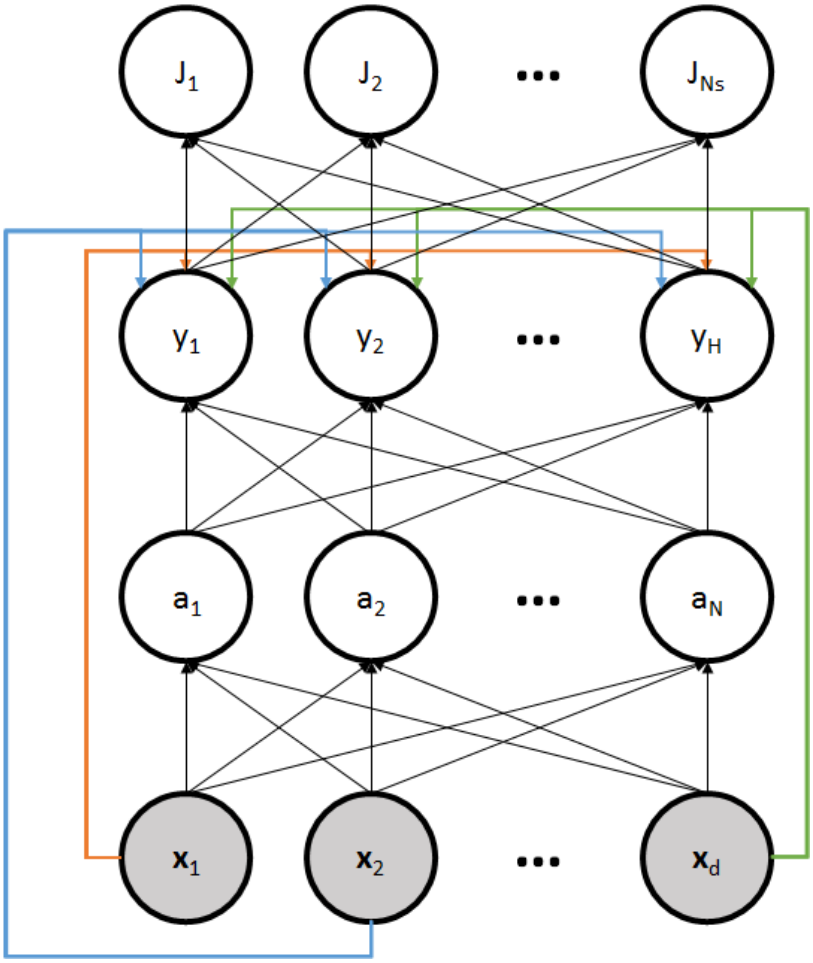


Fig. 4.11. The feedforward neural network NND-AMM that is built by modifying NN-AMM using distributed representation. Notations are the same as those in Figure 4.9. H is the number of key poses after performing the hierarchical-clustering method, and is typically less than M. $J_1, \ldots, J_{N_s}$ are the estimated 3D human-joint locations. N_s is the number of human joints in a 3D human-pose estimate.

In Figure 4.11, neurons $y_1, \cdots, y_H$ in a layer that is called a key-pose layer represent cluster prototypes computed by the hierarchical-clustering method. In the hierarchical-clustering method, we iteratively group two elements that could be key poses or clusters with the smallest distance to form a new cluster, until there is only one cluster. Since the grouping is repeated until all elements are clustered, it can be represented by a binary tree. The grouping depends on a predefined distance metric between key poses and clusters, and we define the distance of a cluster as the distance (according to a predefined distance metric) between its children. To identify clusters to be used in the key-pose layer, we compute a clustering-inconsistency coefficient [125] for each cluster (node) in a binary tree by

$$\frac{d_c - m_c}{\sigma_c}, \tag{4.2}$$

where d_c is the distance of a cluster c, m_c is the average distance of a cluster and its children, and σ_c is the standard deviation of distances of a cluster and its children. The coefficient is used to determine natural cluster divisions by measuring distances between elements in a cluster. The closer the elements are, the smaller the coefficient is. We specify a clustering-inconsistency-coefficient threshold and traverse a binary tree from the root node in a breadth-first manner. If the clustering-inconsistency coefficient of a cluster is less than or equal to a predefined threshold, the cluster will be selected and its descendants will not be traversed. For each selected cluster, a cluster prototype is computed by averaging key poses belonging to a selected cluster. Since each cluster prototype can be interpreted as a human pose, we consider a cluster prototype as a key pose. Thus, the number of key poses (cluster prototypes) in the key-pose layer of NND-AMM is, in general, less than the number of key poses in NN-AMM. In the output layer, each neuron represents a human joint and is computed by linearly combining a human joint of all cluster prototypes in the key-pose layer.

4.4 Variants of VISH features for adaptability testing

Two variants of VISH features are created to evaluate the adaptability of the proposed NN-based HPE system. To create the first variant, we partially occlude an input data, which is a 3D point cloud, and extract a VISH feature from the data. The feature is called an occluded VISH feature. To create the second variant, we use a continuous random variable to represent a reconstructed VISH feature. The reconstructed VISH feature should be close to a non-occluded VISH feature that is defined as the VISH feature extracted from the input data without occlusions. We assume that the probability density function (PDF) of the random variable is represented by a linear model with additive Gaussian noise. Then, we will show that the linear model is equivalent to a feedforward NN under certain types of inference. Mathematically, we will show that a two-layer feedforward NN in which the activation function of each neuron at the output layer is linear computes the expectation of the linear model. Finally, the model is used to infer a non-occluded VISH feature from an occluded VISH feature. The inferred VISH feature is the reconstructed VISH feature.

Note that the process of handling occlusions is complex. The complexity mainly comes from two parts, namely partially-occluded-human detection and pose estimation. In the literature, they are commonly resolved by building models that are capable of determining whether body parts are visible or occluded. In order to determine if a body part is visible or occluded, those models are trained from occluded training sets. Occluded training sets are built by manually occluding parts of humans in observations. Occlusions are simulated by removing parts of an observation in the training sets. For example, the partially-occluded-human-detection model [126] that extends the Deformable Part Model (DPM) [127] is trained by manually removing parts that are occluded. Figure 4.12 shows an example of visible and occluded parts in a training set. A partially-occluded-HPE model [128] is trained by removing parts of images in the HumanEva dataset [129]. Figure 4.13 shows some occluded images in the dataset.

Following the concept of building occluded training sets in the literature, we create an occluded VISH feature by removing parts of an input data at a time. As our goal of

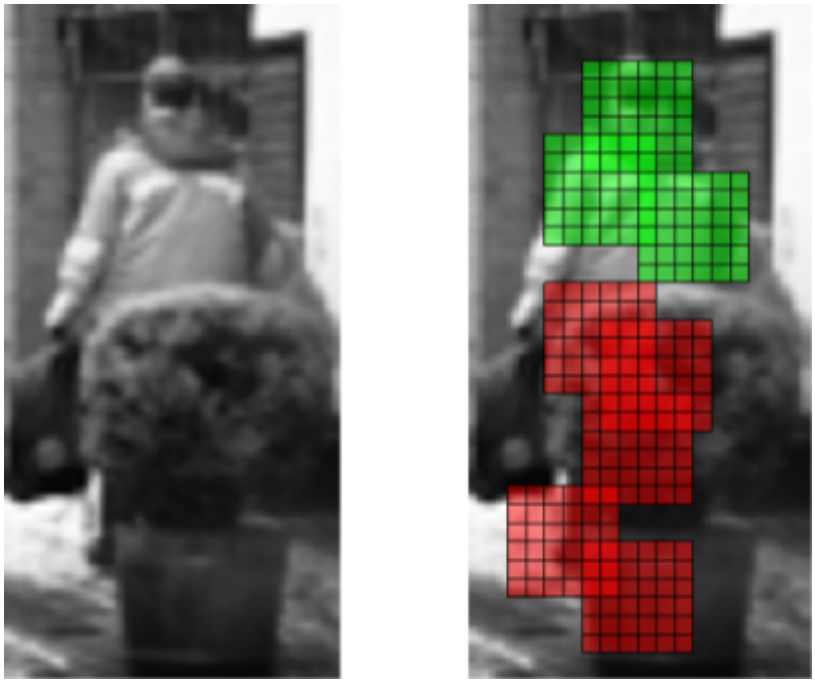


Fig. 4.12. An example of visible (green) and occluded (red) parts in an observation in a training set.

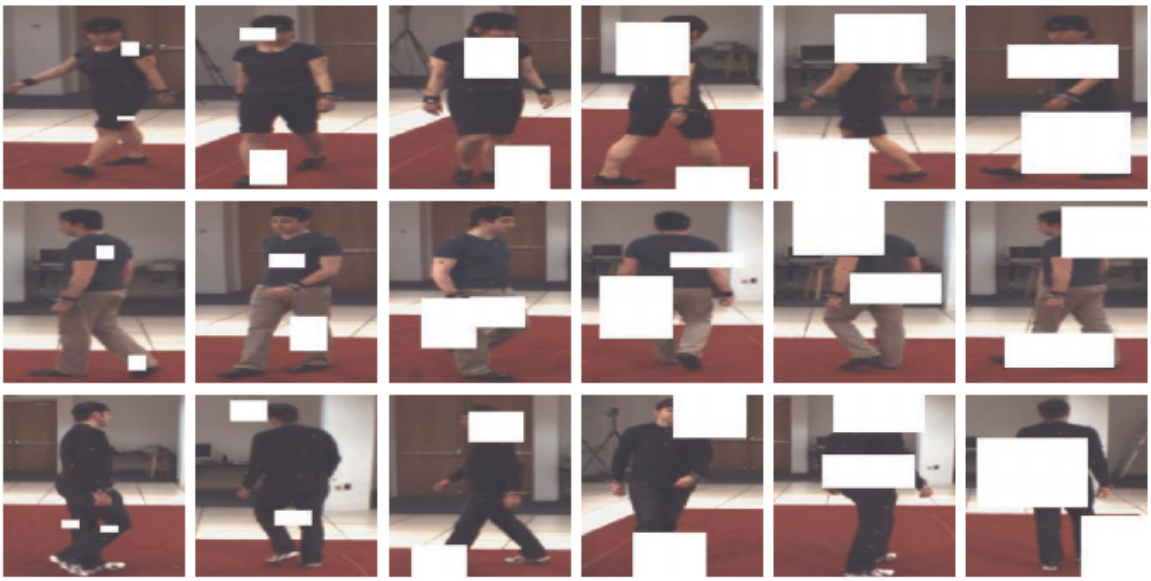


Fig. 4.13. Examples of images in an occluded training set for HPE.

creating an occluded VISH feature is to test the adaptability of the NN-based HPE system, we simplify the process of simulating occlusions by removing one cuboid at a time (details

can be found in Section 4.4.1). This simplification does not directly produce real-world occlusion scenarios because, in reality, more than one cuboid may be occluded at a time. Also, we may lose visible information in a cuboid. For example, if only one human joint is occluded but the cuboid that contains the human joint contains other visible human joints, both visible and occluded human joints in that cuboid will be removed. However, we can leverage this problem by defining fine-grained cuboids. Since we do not aim at solving the occlusion problem, which is complex, in this thesis, we do not attempt to handle the two cases described above. In addition, the inference process of creating a reconstructed VISH feature is simplified because we want to show that there exists a model that is equivalent to a feedforward NN.

Three types of features will be considered as observed random variables in AMM. When it is not important to distinguish the three types, we use the notations $\mathbf{X}$ and $\mathbf{x}$, which are defined in Section 4.3.1, to denote a random variable and its value representing either one of them, respectively. When it is necessary to distinguish them, we denote random variables representing non-occluded, occluded and reconstructed VISH features by $\mathbf{X}_n$, $\mathbf{X}_o$, and $\mathbf{X}_r$, respectively. Their values are denoted by $\mathbf{x}_n$, $\mathbf{x}_o$, and $\mathbf{x}_r$, respectively.

In the following, we will describe the two variants in details. Details of extracting VISH features can be found in [104]. In addition to the notations defined in [104], we use $\mathscr{P}_H^{i,j}$ to denote the j-th node at i-th level in a tree created in the process of hierarchical structuring.

4.4.1 Occluded VISH Feature

The process of extracting occluded VISH features is derived based on the non-occluded VISH feature extraction [104]. During the extraction, an occlusion step is added before the feature-extraction step. In the occlusion step, for each pre-processed 3D point cloud $\mathscr{P}_H$, one leaf node, which is to be occluded, is chosen at random (with probability $1/N_c$) from the set $\{\mathscr{P}_H^{L-1,0}, \mathscr{P}_H^{L-1,1}, \ldots, \mathscr{P}_H^{L-1,N_c-1}\}$ in a tree of a pre-processed 3D point cloud, where L is the number of levels in a tree and N_c is the number of leaf nodes ($L = 2$ and $N_c = 6$ in this work). Let $\mathscr{P}_H^{L-1,k}$ be the chosen leaf node. 3D points in the chosen leaf node

$\mathscr{P}_H^{L-1,k}$ are then removed to simulate the absence of 3D points when the corresponding cuboid is occluded, i.e., $\mathscr{P}_H^{L-1,k} = \emptyset$. During the feature-extraction step, both VFH and shape features do not exist in the chosen leaf node $\mathscr{P}_H^{L-1,k}$ because $\mathscr{P}_H^{L-1,k}$ is empty. They are thus set to zeros. Figure 4.14 shows examples of 3D point clouds after the occlusion step.

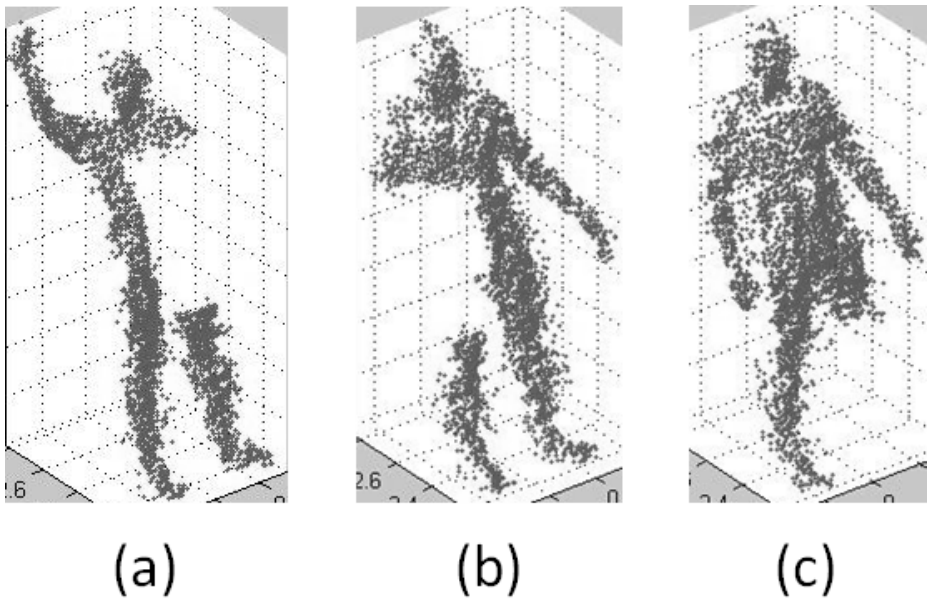


Fig. 4.14. Occluded 3D point clouds. (a) Middle right region is occluded. (b) Middle left region is occluded. (c) Lower right region is occluded.

When extracting non-occluded and occluded VISH features from a pre-processed 3D point cloud $\mathscr{P}_H$, they are different not only in the VFH and shape features from the chosen leaf node $\mathscr{P}_H^{L-1,k}$. Their VFH and shape features computed from ancestors of the chosen leaf node $\mathscr{P}_H^{L-1,k}$ are also different because portions of 3D points in ancestors are occluded (empty). In addition, VFH and shape features from nodes that are 8-connected neighbors of the chosen leaf node $\mathscr{P}_H^{L-1,k}$ may be changed because those features may depend on 3D points in an 8-connected neighborhood.

4.4.2 Reconstructed VISH Feature

A linear model with additive Gaussian noise, which is a Bayesian network, is built to reconstruct non-occluded VISH features from occluded VISH features, and illustrate that it is the same as a feedforward NN under certain inferences. The linear model contains two random variables $\mathbf{X}_r$ and $\mathbf{X}_o$ representing reconstructed and occluded VISH features, respectively. A directed edge is established to connect the random variable $\mathbf{X}_o$ to the random variable $\mathbf{X}_r$ because the value of random variable $\mathbf{X}_r$ is estimated from the value of the random variable $\mathbf{X}_o$. Figure 4.15 shows the DAG and the factor graph of the model.

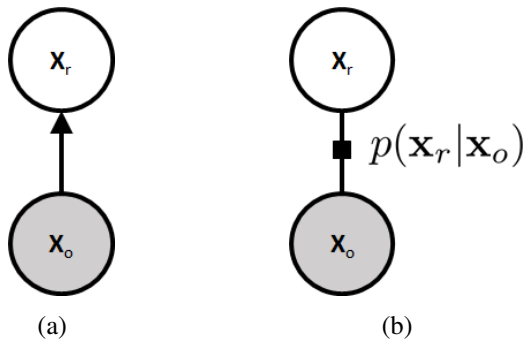


Fig. 4.15. (a) DAG and (b) factor graph of the linear model. The observed variable $\mathbf{X}_o$ is shaded.

In the linear model, the random variable $\mathbf{X}_r$ is Gaussian distributed and depends linearly on $\mathbf{x}_o$; that is,

$$\mathbf{x}_r = \mathbf{W}\mathbf{x}_o + \psi, \tag{4.3}$$

where $\mathbf{W}$ is a square matrix with dimension the same as VISH features, and ψ is Gaussian distributed with a zero mean vector and a covariance matrix R. The value of the random variable $\mathbf{X}_r$ is estimated by the value with the highest probability, denoted by $\hat{\mathbf{x}}_r$. It can be shown that the value is given by

$$\hat{\mathbf{x}}_r = \arg\max p(\mathbf{x}_r|\mathbf{x}_o) = \mathbf{W}\mathbf{x}_o. \tag{4.4}$$

In Eq. (4.4), we deduce that the estimate $\hat{\mathbf{x}}_r$ can be computed by a feedforward NN. The feedforward NN consists of an input layer and an output layer. The input layer represents an occluded VISH feature. Each neuron in the input layer corresponds to an element in the feature. The output layer represents an estimate of a reconstructed VISH feature. Each neuron in the output layer corresponds to an element in the estimate. The two layers are fully connected and the activation function in the output layer is linear (see Figure 4.16).

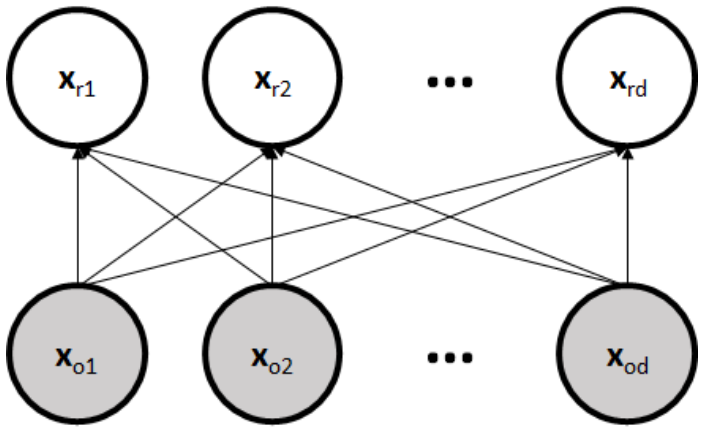


Fig. 4.16. Neural network that computes the estimate of the linear model. Neurons in the input layer are shaded. $\mathbf{x}_{oi}$ is the i-th element of an occluded VISH feature $\mathbf{x}_o$, for $1 \leq i \leq D$. Similarly, $\mathbf{x}_{ri}$ is the i-th element of a reconstructed VISH feature $\mathbf{x}_r$. D is the dimension of an occluded or reconstructed VISH feature.

4.5 Experimental Results

Experiments were conducted to evaluate the ability of designing factors using the proposed mapping, the effectiveness of distributed representation, and the adaptability of using feedforward NNs to realize AMM factors in the NN-based HPE system. The Stanford TOF Motion Capture Dataset [70], which is a benchmark dataset for 3D HPE, was used throughout the experiments. In the dataset, there are 28 video sequences. Each video sequence corresponds to one action. Each frame has a human pose that has 15 degrees-of-freedom (joints), namely head, neck, left/right shoulder, left/right elbow, left/right wrist, hip center, left/right hip, left/right knee and left/right ankle. The number of frames of the video se-

quences is shown in Table 4.1. In the dataset, a subject performed different actions such as kicking and rotation, and was captured by a Swissranger SR4000 TOF sensor. The range images were captured at 25 frames per second with a resolution of 176×144 pixels. The ground-truth 3D human-joint locations were recorded by a commercial motion-capturing system.

Table 4.1 Total number of frames in each video sequence from the dataset [70].

Number of frames	Video index
100	0-5,8,9,14,16,18,
400	6,7,10-13,15,17,19,20-27

Following the experimental setting in [117], the dataset was divided into 20% for establishing the human-pose database that contains key poses, 30% for validation, 40% for training and 10% for testing. To reduce the bias in dividing the dataset, 10 trials were created by randomly dividing the dataset with different random seeds. Each trial was tested using 5-fold cross-validation. Parameters of AMM were set according to [117]. In the hierarchical-clustering method, the distance metric between two key poses was defined as the Euclidean distance, and the distance between two clusters was defined as the shortest distance between elements from each of the two clusters.

When evaluating the HPE performance, frames with missing ground-truth 3D joint locations were ignored. Two metrics were used to quantify the performance in two aspects. The first metric, which is called average-joint-error metric, is a widely used metric that measures the average error on each joint and is denoted by ζ_1. It is defined as

$$\zeta_1 = \frac{1}{N_f} \sum_{s=1}^{N_f} \frac{1}{N_s} \sum_{i=1}^{N_s} \left\| \mathbf{j}_{s,i} - \tilde{\mathbf{j}}_{s,i} \right\|_2 , \tag{4.5}$$

where N_f is the number of frames of the video sequence for testing, N_s is the number of 3D human-joint locations measured by the motion-capturing system in the s-th frame, $\mathbf{j}_{s,i}$ and $\tilde{\mathbf{j}}_{s,i}$ are the ground-truth and the estimated 3D location of the i-th human-joint in the s-th frame, respectively, and $\| \cdot \|_2$ is the Euclidean norm.

The second metric, which is called average-pose-accuracy metric, measures the proportion of human-pose estimates that have all joints within an error less than a certain Euclidean distance. It is an extremely challenging metric that allows us to easily identify the number of human-pose estimates within an Euclidean distance from their corresponding ground-truth human poses. We call human-pose estimates that have all joints within a certain error true human-pose estimates, and other human-pose estimates false human-pose estimates. The metric is denoted by ζ_2, and is defined as

$$\zeta_2 = \frac{1}{N_f} \sum_{s=1}^{N_f} \prod_{i=1}^{N_s} \mathbb{1}(\|\mathbf{j}_{s,i} - \tilde{\mathbf{j}}_{s,i}\|_2 < d), \tag{4.6}$$

where $\mathbb{1}(\cdot)$ is an indicator function that equals to 1 when the input argument is correct and 0 otherwise, and d is the maximum Euclidean distance between a joint in a true human-pose estimate and its corresponding ground-truth joint position. For the results below, we plot the change of ζ_2 over a range of d. In particular, we include ζ_2 when $d = 0.2$m because we believe it is the maximum distance to determine true human-pose estimates that are useful in most interactive systems.

4.5.1 Ability of designing factors using the proposed mapping

We analyzed the performance of each model (AMM, NN-AMM, and NND-AMM). As factors in AMM were designed manually based on non-occluded VISH features, the features were used as input to the three models.

Comparison based on the average-joint-error metric ζ_1

Table 4.2 shows average joint errors and standard deviations of HPE incurred in AMM, NN-AMM, and NND-AMM when the clustering-inconsistency-coefficient threshold was set to 0.6. Bar charts of the errors and standard deviations in different trials are shown in Figures 4.17 and 4.18. When comparing AMM with NN-AMM, their average joint errors and standard deviations were similar. The errors and standard deviations among 10 trials

were different because of the random division of the dataset. The overall performance of the 10 trials showed that NN-AMM had lower errors than AMM. The result showed that the automatic design of factors using the proposed mapping was effective. When we compared NND-AMM with AMM and NN-AMM, the number of key poses was reduced from 1214 to 836 (about 31% reduction in the number of key poses). Both the errors and standard deviations incurred in NND-AMM were the lowest because estimating combination coefficients was relatively accurate when the number of key poses was decreased, and key poses could be combined to generate human poses in the human-pose database. Thus, the result showed that the distributed representation in NND-AMM could reduce the number of key poses while having the lowest average joint error and standard deviation among the three models.

Table 4.2 Average Joint Errors (in meters) of HPE. Numbers on the left and in the parentheses are the average joint errors and standard deviations of HPE, respectively.

Trial	AMM	NN-AMM	NND-AMM
1	0.0276 (0.0338)	0.0259 (0.0310)	0.0234 (0.0261)
2	0.0267 (0.0309)	0.0269 (0.0332)	0.0241 (0.0279)
3	0.0263 (0.0277)	0.0256 (0.0268)	0.0229 (0.0224)
4	0.0267 (0.0342)	0.0261 (0.0314)	0.0239 (0.0264)
5	0.0257 (0.0288)	0.0253 (0.0292)	0.0223 (0.0218)
6	0.0281 (0.0351)	0.0272 (0.0306)	0.0240 (0.0241)
7	0.0272 (0.0325)	0.0257 (0.0300)	0.0229 (0.0263)
8	0.0263 (0.0329)	0.0265 (0.0304)	0.0238 (0.0254)
9	0.0272 (0.0372)	0.0260 (0.0328)	0.0241 (0.0290)
10	0.0270 (0.0352)	0.0257 (0.0314)	0.0235 (0.0262)
Overall	**0.0269 (0.0328)**	**0.0261 (0.0307)**	**0.0235 (0.0255)**

Comparison based on the average-pose-accuracy metric ζ_2

Figure 4.19 shows the change of proportions of true human-pose estimates when the maximum Euclidean distance (d) is changed from 0m to 0.5m with a step size 0.01m. The percentages of true human-pose estimates computed by the three models when $d =$ 0.2m are explicitly shown in the figure. When d was small (between 0m and 0.04m), the

Fig. 4.17. Average joint errors incurred in the three models in different trials.

Fig. 4.18. Change of standard deviations of human-pose estimates in different trials.

percentages of true human-pose estimates computed by AMM and NN-AMM were similar, and were higher than that computed by NND-AMM. When d increased from 0.04m to 0.35m, the percentage computed by NND-AMM was higher than that of NN-AMM. The performance of AMM was the worst among the three models. Finally, the percentage computed by each model converged to 1. Overall, the performance of NND-AMM was the best among the three models. The average of maximum joint errors of human poses estimated in NND-AMM was smaller than that in AMM and NN-AMM. It was because in a distributed representation, key poses with non-zero weights and high maximum joint errors could be combined with key poses with non-zero weights and low maximum joint errors to reduce the error. However, when a key pose with the lowest maximum joint error had the largest weight, using distributed representation would yield a larger error than using the key pose. Hence, NND-AMM performance was the worst when d was small.

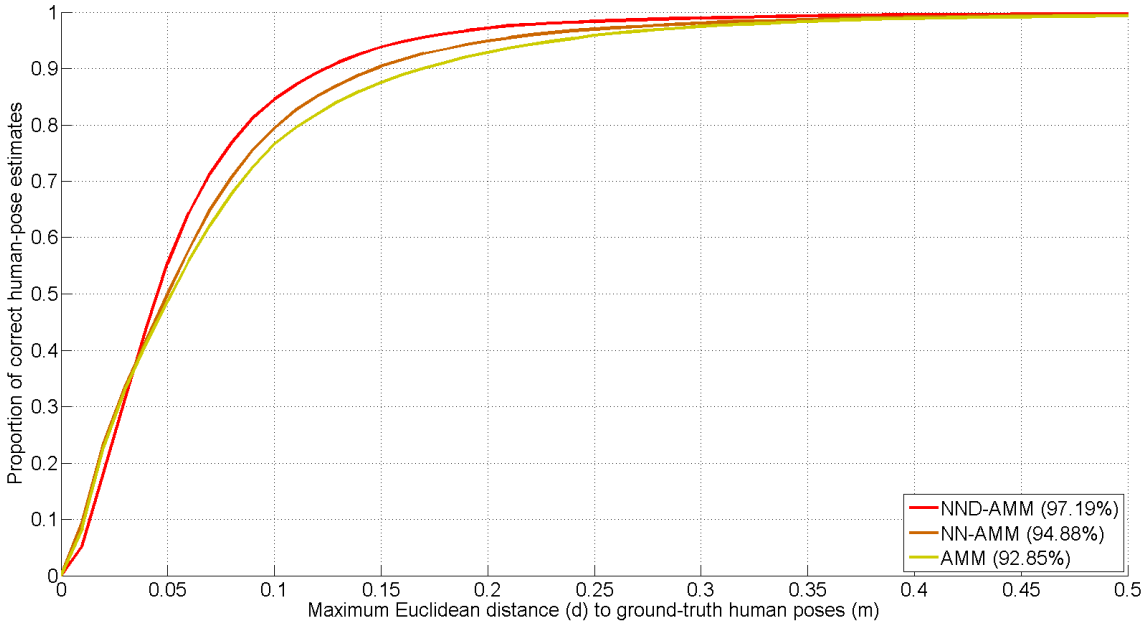


Fig. 4.19. Change of proportions of true human-pose estimates with different maximum Euclidean distances. Numbers in the legend are the percentages of true human-pose estimates computed by the three models when d = 0.2m.

4.5.2 Effectiveness of distributed representation in NND-AMM

We analyzed the performance of NND-AMM at different clustering-inconsistency-coefficient thresholds. When the threshold was changed, the number of neurons (key poses) in the key-pose layer of NND-AMM was changed. The variation is shown in Figure 4.20. When the threshold was zero, the hierarchical-clustering method was not performed. In other words, each key pose was a human pose in the human-pose database. The figure shows a decreasing trend in the number of key poses as the threshold increases. When the threshold was between 0.1 and 0.7, the number of key poses was almost the same. It showed that human poses in human-pose database contained natural clusters that were well-separated from each other.

In general, a large threshold could reduce the number of neurons (key poses) and hence the processing time in NND-AMM. However, errors in HPE would increase. In practice, an optimal threshold could be selected by computing the errors over a range of thresholds on a validation set, and compromising between the errors and the number of neurons.

Note that there was only one key pose when the threshold was bigger than or equal to 1.2. Thus, NND-AMM was only evaluated when the threshold was set between 0 to 1.1. Figure 4.21 shows two key poses (in red) and human poses (in blue) belonging to each key pose when the threshold was set to 1.

Comparison based on the average-joint-error metric ζ_1

Figure 4.22 shows the variation of average joint errors and their standard deviations incurred in NND-AMM when the threshold was changed from 0 to 1.1 with a step size equal to 0.1. The errors incurred in AMM were also included as a reference for comparison between AMM and NND-AMM. As AMM used human poses in the human-pose database instead of key poses computed in clustering, the errors in AMM were not affected by the threshold. For NND-AMM, the average joint error had an increasing trend when the threshold increased, except that it was decreased at first at the threshold 0.1 and was almost identical until at the threshold 0.7. The reduction of error was due to two reasons. First,

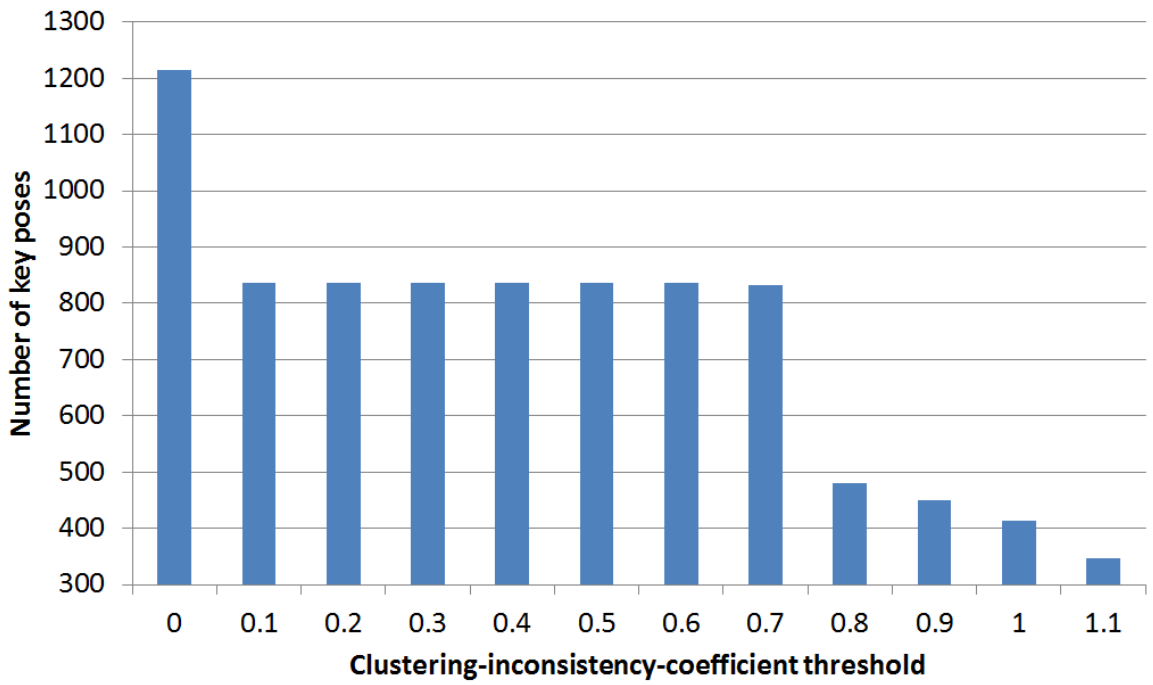
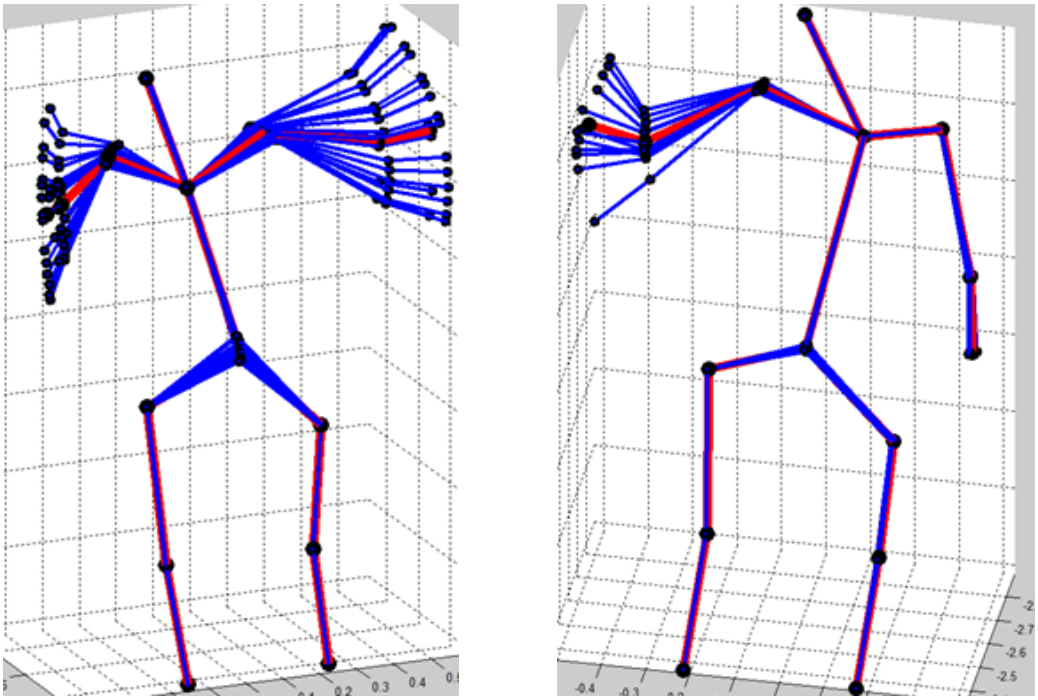


Fig. 4.20. Change of number of key poses in NND-AMM with different clustering-inconsistency-coefficient thresholds.

the number of key poses was decreased, and thus there was a higher chance to put more weights to key poses similar to ground-truth human poses. Second, the decrease of key poses did not affect key poses to represent human poses in the human-pose database. As the number of key poses was almost the same at a threshold between 0.1 and 0.7, the errors were similar (they were not identical because the numbers of key poses in different trials and cross-validation sets were different). Finally, as the number of key poses decreased further, key poses could not represent human poses in the human-pose database. Therefore, the error was increased. For the change of standard deviations, it was in general increased when the threshold increased because the variation of key poses was increased. Overall, the result showed that the distributed representation in NND-AMM could effectively reduce the number of key poses while keeping similar performance of AMM over a range of thresholds.



(a) (b)

Fig. 4.21. Two key poses (in red) and human poses belonging to each key pose (in blue) when the clustering-inconsistency-coefficient threshold is set to 1.

Comparison based on the average-pose-accuracy metric ζ_2

Figure 4.23 shows the performance change of NND-AMM at different thresholds. The percentages of true human-pose estimates computed at $d = 0.2$m are explicitly shown in the figure. Curves corresponding to thresholds from 0 to 0.7 were overlapped with each other, showing that their percentages of true human-pose estimates were similar. As the threshold continued to increase, the percentage was decreased. This was consistent with the result evaluated with the average-joint-error metric.

4.5.3 Comparison of our proposed models with existing works

We used the same error metric (average-joint-error metric ζ_1) that was used in some existing works [70] [49] [50] and compared their performance with our proposed models

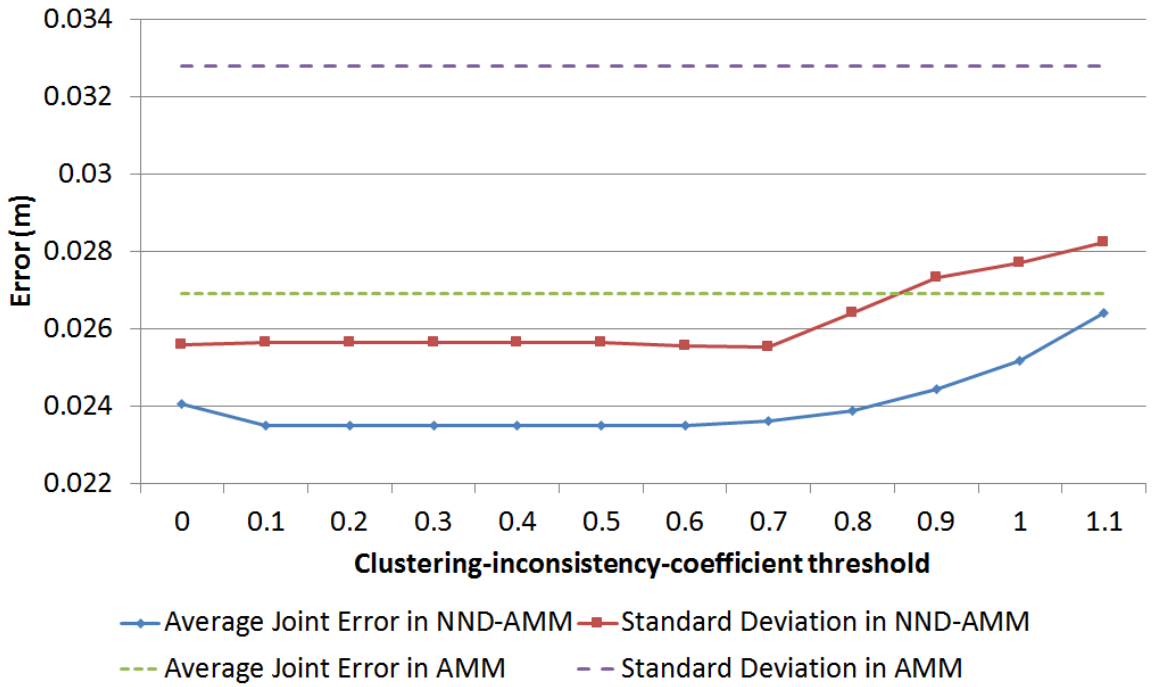


Fig. 4.22. Error change of human poses estimated by AMM and NND-AMM with different clustering-inconsistency-coefficient thresholds.

(AMM, NN-AMM and NND-AMM when the clustering-inconsistency-coefficient threshold was set to 0.6). Table 4.3 shows the errors and standard deviations of HPE incurred in the existing works and our proposed models. When we compared AMM and the existing works, AMM incurred the lower error and standard deviation, showing that using human actions could reduce errors in HPE. Both the errors and standard deviations incurred in NND-AMM were the lowest.

4.5.4 Adaptability of the three models with different features

The adaptability of a model can be measured by its performance with different model input because the underlying human-pose distribution depends on model input. Thus, we compared the performance of each model when its input was changed from non-occluded VISH features to occluded and reconstructed VISH features.

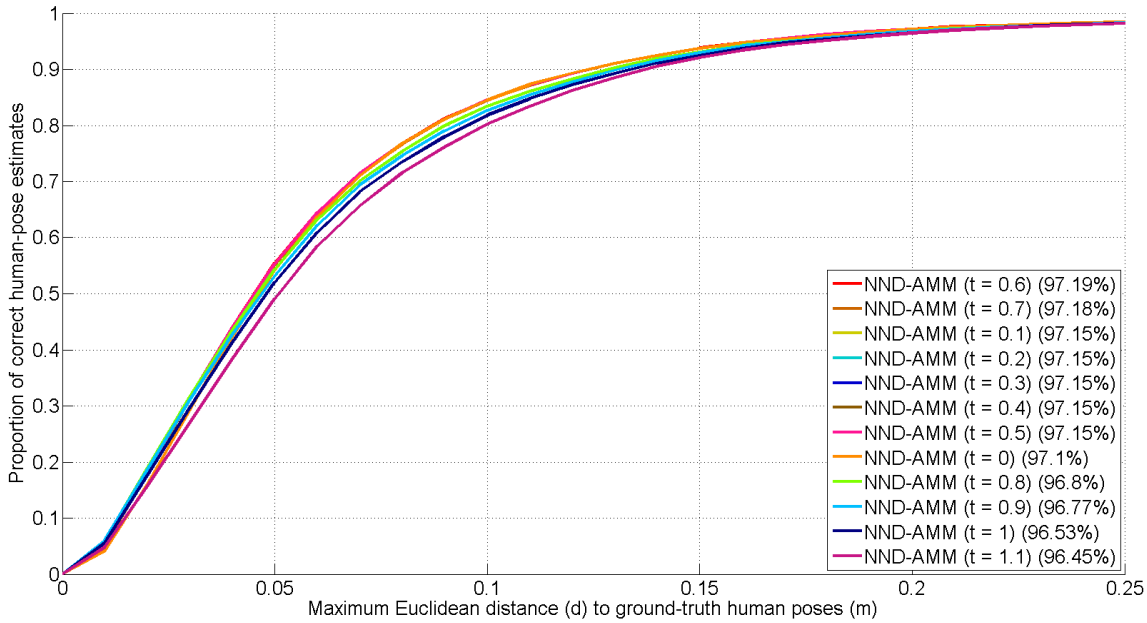


Fig. 4.23. Change of proportions of true human-pose estimates with different maximum Euclidean distances. The clustering-inconsistency-coefficient threshold is denoted by t. Numbers in the legend are the percentages of true human-pose estimates computed using different thresholds when d = 0.2m.

Table 4.3 The errors and standard deviations (S.D.) of HPE in the Stanford TOF Motion Capture Dataset [70]. The errors and standard deviations of the existing works were obtained from their papers.

	Error (m)	S.D. (m)
HC and EP Method [70]	0.1	N.A.
Data-driven Hybrid Method [49]	0.0618	0.0424
Exemplar Method [50]	0.038	N.A.
AMM	0.0269	0.0328
NN-AMM	0.0261	0.0307
NND-AMM	0.0235	0.0255

Comparison based on the average-joint-error metric ζ_1

Table 4.4 shows the performance of each model based on the average joint error ζ_1. Each column shows the errors incurred in the three models with a specific type of VISH features, namely, non-occluded, occluded and reconstructed VISH features. In each model, the error corresponding to the occluded VISH features was the highest compared with

errors corresponding to non-occluded and reconstructed VISH features because occlusions caused missing features. Reconstructed VISH features could infer the missing features and thus yielded lower errors compared with occluded VISH features. For each type of features, AMM performed the worst and NND-AMM performed the best. It was because one factor in AMM was designed manually to measure the Euclidean distance between two VISH features. When VISH features were occluded, the factor couldn't adapt itself and the Euclidean distance between features was not a reasonable measure. Thus, errors in AMM were increased almost two times when the input was changed from non-occluded to occluded features. On the other hand, factors in NN-AMM and NND-AMM could be adapted automatically. Thus, errors of the two models were increased about 1.5 times.

Table 4.4 Average Joint Errors (in meters) of HPE with different VISH features. Numbers on the left and in the parentheses are the average joint errors and standard deviations of HPE, respectively.

Model	Non-occluded	Occluded	Reconstructed
AMM	0.0269 (0.0328)	0.0526 (0.0628)	0.0371 (0.0430)
NN-AMM	0.0261 (0.0307)	0.0390 (0.0446)	0.0366 (0.0427)
NND-AMM	0.0235 (0.0255)	0.0346 (0.0360)	0.0336 (0.0339)

Comparison based on the average-pose-accuracy metric ζ_2

Figure 4.24 shows the performance of each model with different types of features as input. The percentages of true human-pose estimates computed by the three models when $d = 0.2$m are explicitly shown in the figure. The result was consistent with the previous result evaluated on the average-joint-error metric ζ_1. Overall, the performance of AMM with occluded VISH features was the worst. Also, AMM improved the performance the most when its input was changed from occluded to reconstructed VISH features. This change showed the drawback of designing factors manually in AMM. On the other hand, the performance of NND-AMM was the best when using occluded and reconstructed VISH features.

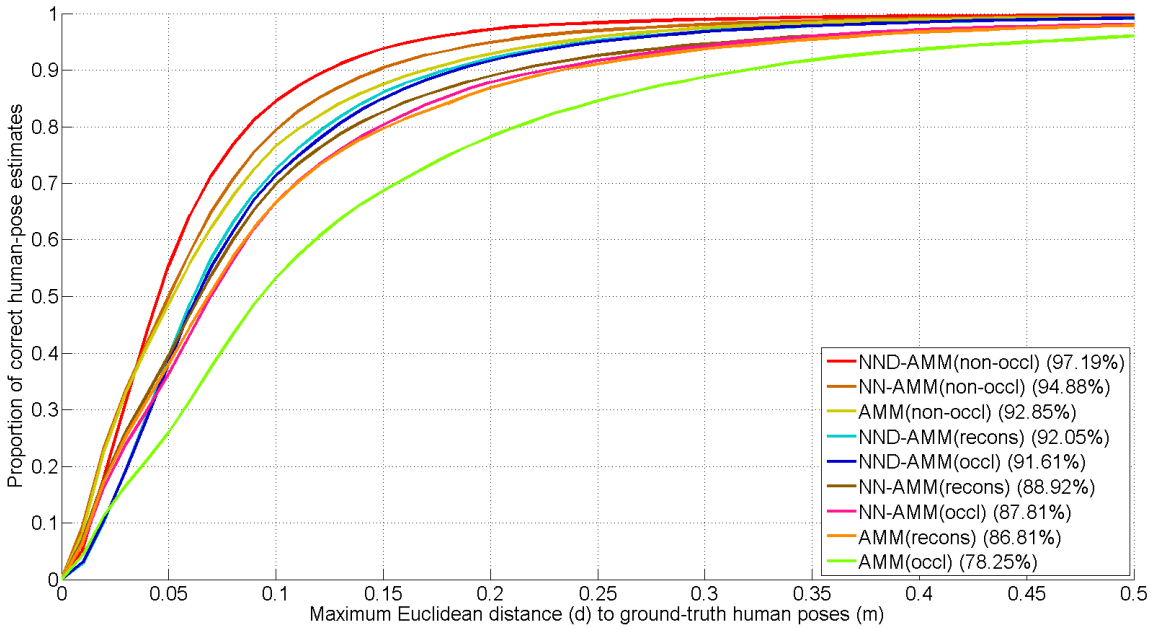


Fig. 4.24. Change of proportions of true human-pose estimates with different features. Non-occl, occl and recons are abbreviations for non-occluded, occluded and reconstructed VISH features, respectively. Numbers in the legend are the percentages of true human-pose estimates when d = 0.2m.

4.6 CONCLUSIONS

In this chapter, we have identified shortcomings of designing factors manually in a Bayesian network. The shortcomings are that (1) the design process is laborious and (2) factors may not represent the underlying probability distributions. These shortcomings exist in the action-mixture model (AMM), which is described in Chapter 3. Specifically, by considering AMM as a Bayesian network, factors in AMM are the pmf of action classification and base distributions. They are designed manually by incorporating human actions in HPE. Although AMM has been shown to outperform some existing works, they may still not represent the underlying distributions.

Being motivated by the shortcomings, we have proposed a mapping that realizes a Bayesian network by a feedforward NN. Based on the proposed mapping, factors can be designed automatically from training data. The proposed mapping eliminates the laborious step of designing factors manually. It consists of two steps, namely structure identifica-

tion and parameter learning. In the structure identification, a bottom-up approach has been presented to build a feedforward NN with modules that represent factors of a Bayesian network. In the parameter learning, a part-based approach has been presented to learn synaptic weights by decomposing a feedforward NN into parts. This way each part contains semantic meaning and synaptic weights in one part can be used as initial weights for training the following part in a part sequence.

Using the proposed mapping, we have extended our previous work on AMM and built NN-AMM. Through the realization, the learning capability and adaptability of NNs can be transferred to AMM. Also, the proposed mapping designs and interprets a feedforward-NN architecture systematically based on the semantic meaning of random variables in AMM. A NN-based HPE system has been presented by building a scalable NND-AMM based on NN-AMM. In the system, key poses are represented by cluster prototypes that are determined using the hierarchical-clustering method. Human poses are estimated using a linear combination of cluster prototypes.

The performance of each model (AMM, NN-AMM, and NND-AMM) was analyzed. Both the errors incurred in AMM and NN-AMM were similar, showing that the proposed mapping could effectively design factors of AMM automatically. The error incurred in NND-AMM was the lowest among our proposed models and some existing works, even though the number of key poses in NND-AMM was lower than that in AMM and NN-AMM. The adaptability of NND-AMM was tested by comparing its error with the errors incurred in AMM and NN-AMM using non-occluded, occluded and reconstructed VISH features. The results showed that both NN-AMM and NND-AMM could adapt to different features. This validated the adaptability of using feedforward NNs to design factors automatically.

5. CONCLUSIONS AND FUTURE RESEARCH

5.1 Summary and Conclusions

In this thesis, we focused on estimating human poses from observations that were captured by a stationary depth sensor. We explored methodologies for solving the HPE problem, and achieved contributions mainly in two aspects: feature extraction and human-pose modeling.

In the feature-extraction aspect, our research effort centered on reducing feature ambiguity. We used a depth sensor to capture an observation, which was a 3D point cloud. From a 3D point cloud, a 3D-point-cloud feature VISH, which utilized the geometric property of a 3D point cloud, was proposed to represent an observation of a person. The proposed feature could be considered as a 3D adaptation of HOG. It contained the geometric structure of a 3D point cloud by arranging 3D points into a tree structure, which preserved the global and local properties of 3D points. It was derived through the steps of 3D-point-cloud preprocessing, hierarchical structuring and feature extraction. In the pre-processing step, region-based thresholding and pseudo-residual were used to stabilize a 3D point cloud from a person. The stable 3D point cloud was then organized into a tree structure. The 3D orientation (pan, tilt and yaw angles) and shape features were extracted from each node in the tree to describe the geometric distribution of 3D points. VISH was derived by combining all the features and therefore preserved the spatial ordering of the stable 3D point cloud. The proposed feature was evaluated on a benchmark dataset and compared with two existing geometric features. Experimental results showed that the proposed feature had the lowest overall error in HPE, and the tree structure (spatial ordering) was particularly important to remove the ambiguity of symmetric 3D human poses.

When modeling human poses, we first defined factors manually in order to express our beliefs in HPE. Then, we automated the process of designing factors by using training

data. In the case of designing factors manually, we proposed a non-parametric action-mixture model (AMM) that estimated human poses using the result of action classification. Based on the concept of distributed representation, high-dimensional human-pose space was represented using low-dimensional manifolds. In AMM, human poses in each manifold were modeled using a base distribution without making stronger assumptions about the nature of the human-pose distribution compared with other parametric models such as an exponential family of distributions. Instead, a base distribution was estimated using an instance-based learning algorithm that measured the similarity of VISH features in the estimation step and frequency of actions in the redistribution step. The action associated with a VISH feature was then classified by the bootstrap aggregating algorithm (bagging) to determine weighting coefficients in combining low-dimensional manifolds. As human poses estimated by the proposed AMM were in discrete space, a kinematic model was used to model the spatial relationship between body parts in continuous space to reduce the quantization error in AMM. Computer-simulation results showed that using multiple low-dimensional manifolds could represent human-pose space and increase the accuracy and precision of human-pose estimates. The overall error and standard deviation were reduced compared with existing approaches without action classification.

As manually designing factors was laborious and yet, the designed factors might not represent the underlying probability distributions, we proposed to use a neural network to automatically design factors in AMM. AMM was extended by realizing it using NN-AMM. The realization introduced the neural-network adaptability to AMM so that factors of AMM could be designed automatically by using NN-AMM. It also introduced the semantic meaning of random variables in AMM to NN-AMM so that the network architecture of NN-AMM could be designed and interpreted systematically. The realization process consisted of structure identification and parameter learning. In the structure identification, a bottom-up approach was proposed to build NN-AMM with modules that represented factors of AMM. In the parameter learning, a part-based approach was proposed to learn synaptic weights by decomposing NN-AMM into parts. With this approach, parts contained semantic meaning, and synaptic weights in one part could be used as initial weights

for training the following part in a part sequence. As the number of key poses in NN-AMM could be large, NND-AMM was created by representing key poses using cluster prototypes that were determined using the hierarchical-clustering method. Based on the concept of distributed representation, human poses were estimated using a linear combination of cluster prototypes. Thus, NND-AMM was scalable. In the experiment, the performance of AMM and NN-AMM was similar, showing that the realization process could effectively design factors of AMM automatically. Human poses estimated by NND-AMM were more accurate than human poses estimated by AMM and NN-AMM even though the number of key poses in NND-AMM was less than that in AMM and NN-AMM. Existing works were compared with NND-AMM and the error incurred in NND-AMM was the lowest.

5.2 Future Research

The study of feature extraction and human-pose modeling using a stationary depth sensor can be extended to the case when observations are captured by a moving depth sensor. The extension has a broader impact on human-robot interaction as robots are often mounted with depth sensors. It is a challenging research topic. We believe there are three components along the direction of the extension. The three components are:

- HPE in the stationary case. When a moving depth sensor is capturing a scene, the viewpoint of the sensor in each frame can be considered as the viewpoint of a stationary depth sensor. Human poses can then be estimated using methods proposed in this thesis. However, the viewpoint of a moving depth sensor may be changing among frames. In some frames, the sensor may not be facing towards the person of interest. To test how different viewpoints affect HPE, we could mount a depth sensor on a mobile/humanoid robot and move the robot randomly around the person of interest. The sensor will then be at different distances and angles from the person. We expect that the body size of the person in the 3D point cloud captured by the sensor will be altered when the distance between the sensor and the person changes. Also, in some frames, the person may not be captured by the sensor. Proposed methods in this the-

sis could be extended to handle the two cases above or determine the viewpoint of the sensor such that the two cases above can be avoided.

- Human-pose prediction. After the current human pose is estimated, a human pose at a future time should be predicted so that a robot with a depth sensor will have enough time to move to the position and orientation determined by the component of best-viewpoint determination below. If the future time is short, the predicted human pose should be close to the current human pose. However, the prediction problem is, in general, a difficult problem because human is highly articulated. For example, if the future time is long enough, a person can move through a wide range of human poses. We could consider the human motion, such as walking, of a person is known in advance so that human poses could be predicted by matching the current human pose in the motion.

- Best-viewpoint determination. Since the viewpoint of a moving depth sensor can be changed over time, we expect that the viewpoint can be used to increase the accuracy of HPE. In this aspect, we proposed a framework [130] that determined the best viewpoint by directly mapping a human pose to the best viewpoint without human-body reconstruction. The proposed framework consisted of four stages: 3D-point-cloud pre-processing, viewpoint instantiation, feature extraction and pose estimation, and viewpoint evaluation. The 3D point cloud of a human captured by depth sensors was first extracted and filtered. The viewpoints of the depth sensors were instantiated using the finite camera model. VISH features were then extracted from depth images generated from instantiated viewpoints. Each viewpoint was evaluated for every human pose estimated by k-NN based on the matching of the VISH features. Experimental results showed that different viewpoints affected the accuracy of human-pose estimates. The maximum reductions of the mean errors (standard deviations) for two subjects were about 30% (46%) and 23% (41%), respectively. Although the proposed framework could reduce the mean errors, the possible viewpoints were computed in advance in the training phase and they were in discrete space. The relationship be-

tween the angle of a person facing and the best viewpoint of the sensor should be investigated so that the possible viewpoints could be computed on-the-fly and the mapping can be extended from discrete space to continuous space.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Y. S. Delahoz and M. A. Labrador, "Survey on fall detection and fall prevention using wearable and external sensors," *Sensors*, vol. 14, no. 10, pp. 19806–19842, 2014.

[2] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, second ed., 2004.

[3] R. Duda, D. Stork, and P. Hart, *Pattern classification and scene analysis. Part 1, Pattern classification*. Wiley, second ed., Nov. 2000.

[4] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Boston, MA, USA: PWS Publishing Co., 1996.

[5] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, pp. 107–116, Apr. 1998.

[6] P. Werbos, "Backpropagation: past and future," in *IEEE International Conference on Neural Networks*, vol. 1, pp. 343–353, Jul. 1988.

[7] T. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, pp. 90–126, Nov. 2006.

[8] R. Poppe, "Vision-based human motion analysis: An overview," *Computer Vision and Image Understanding*, vol. 108, pp. 4–18, Oct. 2007.

[9] D. Hogg, "Model-based vision: a program to see a walking person," *Image and Vision Computing*, vol. 1, no. 1, pp. 5–20, 1983.

[10] S. Wachter and H. Nagel, "Tracking of persons in monocular image sequences," in *IEEE Nonrigid and Articulated Motion Workshop*, pp. 2–9, Jun. 1997.

[11] M. Brand, "Shadow puppetry," in *IEEE International Conference on Computer Vision*, vol. 2, pp. 1237–1244, 1999.

[12] J. MacCormick and M. Isard, "Partitioned sampling, articulated objects, and interface-quality hand tracking," in *Proceedings of the 6th European Conference on Computer Vision-Part II*, ECCV '00, pp. 3–19, London, UK, UK: Springer-Verlag, 2000.

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

[14] 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 on Computer Vision and Pattern Recognition*, vol. 2, pp. II–669–II–676 vol.2, 2001.

[15] J. Mitchelson and A. Hilton, "Simultaneous pose estimation of multiple people using multiple-view cues with hierarchical sampling," in *Proceedings of the British Machine Vision Conference*, BMVA Press, 2003.

[16] R. Plänkers and P. Fua, "Articulated soft objects for multiview shape and motion capture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1182–1187, Sep. 2003.

[17] M. Lee and I. Cohen, "Proposal maps driven mcmc for estimating human body pose in static images," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. II–334–II–341 Vol.2, Jun. 2004.

[18] D. Anguelov, D. Koller, H. Pang, P. Srinivasan, and S. Thrun, "Recovering articulated object models from 3D range data," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, UAI '04, (Arlington, Virginia, United States), pp. 18–26, AUAI Press, 2004.

[19] R. Kehl, M. Bray, and L. van Gool, "Full body tracking from multiple views using stochastic sampling," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 129–136, Jun. 2005.

[20] R. Kehl and L. van Gool, "Markerless tracking of complex human motions from multiple views," *Computer Vision and Image Understanding*, vol. 104, pp. 190–209, Nov. 2006.

[21] J. Rodgers, D. Anguelov, H. Pang, and D. Koller, "Object pose detection in range scan data," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2445–2452, 2006.

[22] Y. Zhu, B. Dariush, and K. Fujimura, "Controlled human pose estimation from depth image streams," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8, Jun. 2008.

[23] Y. Zhu and K. Fujimura, "Bayesian 3D human body pose tracking from depth image sequences," in *Asian Conference on Computer Vision*, vol. 5995 of *Lecture Notes in Computer Science*, pp. 267–278, Springer Berlin / Heidelberg, 2010.

[24] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H. Seidel, "Motion capture using joint skeleton tracking and surface estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1746–1753, Jun. 2009.

[25] Y. Su, H. Ai, T. Yamashita, and S. Lao, "Human pose estimation using exemplars and part based refinement," in *Proceedings of the 10th Asian conference on Computer vision - Volume Part II*, ACCV'10, pp. 174–185, Berlin, Heidelberg: Springer-Verlag, 2011.

[26] 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*, pp. 1–8, Jun 2010.

[27] N. Lehment, M. Kaiser, and G. Rigoll, "Using segmented 3d point clouds for accurate likelihood approximation in human pose tracking," in *IEEE International Conference on Computer Vision Workshops*, pp. 406–413, Nov. 2011.

[28] J. Charles and M. Everingham, "Learning shape models for monocular human pose estimation from the microsoft xbox kinect," in *IEEE International Conference on Computer Vision Workshops*, pp. 1202–1208, Nov. 2011.

[29] Y. Wang, D. Tran, and Z. Liao, "Learning hierarchical poselets for human parsing," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1705–1712, Jun. 2011.

[30] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1385–1392, Jun. 2011.

[31] Y. Liu, C. Stoll, J. Gall, H. Seidel, and C. Theobalt, "Markerless motion capture of interacting characters using multi-view image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1249–1256, Jun. 2011.

[32] G. Pons-Moll, A. Baak, J. Gall, L. Leal-Taixe, M. Muller, H. Seidel, and B. Rosenhahn, "Outdoor human motion capture using inverse kinematics and von mises-fisher sampling," in *IEEE International Conference on Computer Vision*, pp. 1243–1250, Nov. 2011.

[33] K. Duan, D. Batra, and D. Crandall, "A multi-layer composite model for human pose estimation," in *Proceedings of the British Machine Vision Conference*, pp. 116.1–116.11, BMVA Press, 2012.

[34] N. Howe, "Silhouette lookup for automatic pose tracking," in *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pp. 15–22, Jun. 2004.

[35] 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*, vol. 2, pp. 882–888, Jun. 2004.

[36] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas, "Learning to reconstruct 3d human motion from bayesian mixtures of experts. a probabilistic discriminative approach," tech. rep., University of Toronto, Oct 2004.

[37] L. Taycher, G. Shakhnarovich, D. Demirdjian, and T. Darrell, "Conditional random people: Tracking humans with crfs and grid filters," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 222–229, Jun. 2006.

[38] R. Poppe, "Evaluating example-based pose estimation: Experiments on the humaneva sets," in *CVPR 2nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation*, 2007.

[39] D. Ramanan, "Learning to parse images of articulated bodies," in *Advances in Neural Information Processing Systems 19* (B. Schölkopf, J. Platt, and T. Hoffman, eds.), pp. 1129–1136, Cambridge, MA: MIT Press, 2007.

[40] A. Fathi and G. Mori, "Human pose estimation using motion exemplars," in *IEEE Conference on Computer Vision*, pp. 1–8, Oct. 2007.

[41] R. Okada and S. Soatto, "Relevant feature selection for human pose estimation and localization in cluttered images," in *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pp. 434–445, Berlin, Heidelberg: Springer-Verlag, 2008.

[42] V. Ferrari, M. Marin-Jimenez, and A. Zisserman, "Progressive search space reduction for human pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, Jun. 2008.

[43] M. Hofmann and D. Gavrila, "Multi-view 3D human pose estimation combining single-frame recovery, temporal integration and model adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2214–2221, Jun. 2009.

[44] S. Wang, H. Ai, T. Yamashita, and S. Lao, "Combined top-down/bottom-up human articulated pose estimation using Adaboost learning," in *IEEE International Conference on Pattern Recognition*, pp. 3670–3673, Aug. 2010.

[45] S. Sedai, M. Bennamoun, and D. Huynh, "Localized fusion of shape and appearance features for 3D human pose estimation," in *Proceedings of the British Machine Vision Conference*, pp. 51.1–51.10, British Machine Vision Association, 2010.

[46] X. Zhao, Y. Fu, H. Ning, Y. Liu, and T. Huang, "Human pose regression through multiview visual fusion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, pp. 957–966, Jul. 2010.

[47] G. W. Taylor, L. Sigal, D. J. Fleet, and G. E. Hinton, "Dynamical binary latent variable models for 3d human pose tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 631–638, Jun. 2010.

[48] 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*, pp. 3108–3113, May 2010.

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

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

[51] 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 Conference on Computer Vision and Pattern Recognition*, pp. 1297–1304, Jun. 2011.

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

[53] Y. Tian, L. Sigal, H. Badino, F. De la Torre, and Y. Liu, "Latent gaussian mixture regression for human pose estimation," in *Proceedings of the 10th Asian conference on Computer vision - Volume Part III*, ACCV'10, (Berlin, Heidelberg), pp. 679–690, Springer-Verlag, 2011.

[54] M. Straka, S. Hauswiesner, M. Rüther, and H. Bischof, "Skeletal graph based human pose estimation in real-time," in *Proceedings of the British Machine Vision Conference*, pp. 69.1–69.12, BMVA Press, 2011.

[55] C. Stoll, N. Hasler, J. Gall, H. Seidel, and C. Theobalt, "Fast articulated motion tracking using a sums of gaussians body model," in *IEEE International Conference on Computer Vision*, pp. 951–958, Nov 2011.

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

[57] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon, "The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 103–110, Jun 2012.

[58] Y. Tian, C. Zitnick, and S. Narasimhan, "Exploring the spatial hierarchy of mixture models for human pose estimation," in *Computer Vision ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), vol. 7576 of *Lecture Notes in Computer Science*, pp. 256–269, Springer Berlin Heidelberg, 2012.

[59] X. Chen and A. Yuille, "Articulated pose estimation by a graphical model with image dependent pairwise relations," in *Advances in Neural Information Processing Systems*, pp. 1736–1744, 2014.

[60] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler, "Learning human pose estimation features with convolutional networks," in *International Conference on Learning Representations*, Apr. 2014.

[61] S. Li and A. B. Chan, "3d human pose estimation from monocular images with deep convolutional neural network," in *Asian Conference on Computer Vision*, Nov. 2014.

[62] S. Li, Z.-Q. Liu, and A. Chan, "Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network," *International Journal of Computer Vision*, pp. 1–18, 2014.

[63] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653–1660, Jun. 2014.

[64] T. Pfister, K. Simonyan, J. Charles, and A. Zisserman, "Deep convolutional neural networks for efficient pose estimation in gesture videos," in *Asian Conference on Computer Vision*, 2014.

[65] W. Ouyang, X. Chu, and X. Wang, "Multi-source deep learning for human pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2329–2336, Jun. 2014.

[66] J. Tompson, A. Jain, Y. Lecun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Advances in Neural Information Processing Systems*, pp. 1799–1807, 2014.

[67] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard, "Tracking loose-limbed people," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–421–I–428, Jun. 2004.

[68] 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*, vol. 98, pp. 15–48, May 2012.

[69] A. Gupta, T. Chen, F. Chen, D. Kimber, and L. Davis, "Context and observation driven latent variable model for human pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, Jun 2008.

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

[71] J. Gall, A. Yao, and L. van Gool, "2D action recognition serves 3D human pose estimation," in *Proceedings of the 11th European conference on computer vision*, (Berlin, Heidelberg), pp. 425–438, Springer-Verlag, 2010.

[72] J. Gall, B. Rosenhahn, T. Brox, and H. Seidel, "Optimization and filtering for human motion capture," *International Journal of Computer Vision*, vol. 87, pp. 75–92, Mar. 2010.

[73] A. Yao, J. Gall, G. Fanelli, and L. van Gool, "Does human action recognition benefit from pose estimation?," in *Proceedings of the British Machine Vision Conference*, pp. 67.1–67.11, BMVA Press, 2011.

[74] A. Yao, J. Gall, and L. van Gool, "Coupled action recognition and pose estimation from multiple views," *International Journal of Computer Vision*, vol. 100, no. 1, pp. 16–37, 2012.

[75] J. Gall, A. Fossati, and L. van Gool, "Functional categorization of objects using real-time markerless motion capture," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1969–1976, Jun 2011.

[76] S. Sedai, D. Huynh, and M. Bennamoun, "Supervised particle filter for tracking 2D human pose in monocular video," in *IEEE Workshop on Applications of Computer Vision*, pp. 367–373, Jan. 2011.

[77] S. Zuffi, O. Freifeld, and M. Black, "From pictorial structures to deformable structures," in *IEEE Conference on Computer Vision and Pattern Recognition*, (Washington, DC, USA), pp. 3546–3553, IEEE Computer Society, 2012.

[78] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893 vol. 1, 2005.

[79] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–511–I–518, 2001.

[80] K. Murphy, *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*. The MIT Press, Aug. 2012.

[81] P. Felzenszwalb and D. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, vol. 61, pp. 55–79, Jan. 2005.

[82] D. Sankoff and J. Kruskal, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Center for the Study of Language and Inf, Dec. 1999.

[83] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Numerical Analysis* (G. A. Watson, ed.), pp. 105–116, Berlin: Springer, 1977.

[84] M. Gupta and Y. Chen, "Theory and use of the EM algorithm," *Found. Trends Signal Process.*, vol. 4, pp. 223–296, Mar. 2011.

[85] R. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 2155–2162, Oct. 2010.

[86] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, pp. 1–127, Jan. 2009.

[87] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798–1828, Aug. 2013.

[88] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[89] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[90] C. Stewart, C. Tsai, and B. Roysam, "The dual-bootstrap iterative closest point algorithm with application to retinal image registration," *IEEE Transactions on Medical Imaging*, vol. 22, pp. 1379–1394, Nov. 2003.

[91] X. He and P. Niyogi, "Locality preserving projections," (Cambridge, MA), MIT Press, 2004.

[92] H. Ackley, E. Hinton, and J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, pp. 147–169, 1985.

[93] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," in *Parallel Distributed Processing*, vol. 1, pp. 194–281, MIT Press, 1987.

[94] R. Neal, "Asymmetric parallel boltzmann machines are belief networks," *Neural Computation*, vol. 4, pp. 832–834, Nov. 1992.

[95] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," *International Conference on Artificial Intelligence and Statistics*, vol. 15, pp. 29–37, 2011.

[96] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, Jul. 2006.

[97] B. Apolloni and D. de Falco, "Learning by asymmetric parallel boltzmann machines," *Neural Computation*, vol. 3, no. 3, pp. 402–408, 1991.

[98] B. M. Marlin, K. Swersky, B. Chen, and N. de Freitas, "Inductive principles for restricted boltzmann machine learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 509–516, May 2010.

[99] R. M. Neal, "Connectionist learning of belief networks," *Artificial Intelligence*, vol. 56, pp. 71–113, Jul. 1992.

[100] Y. Bengio and S. Bengio, "Modeling high-dimensional discrete data with multi-layer neural networks," in *Advances in Neural Information Processing Systems*, pp. 400–406, MIT Press, 2000.

[101] J. Gall, A. Yao, N. Razavi, L. van Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 2188–2202, Nov. 2011.

[102] L. Alboul and G. Chliveros, "A system for reconstruction from point clouds in 3D: Simplification and mesh representation," in *IEEE International Conference on Control Automation Robotics Vision*, pp. 2301–2306, Dec. 2010.

[103] W. Brink, A. Robinson, and M. Rodrigues, "Indexing uncoded stripe patterns in structured light systems by maximum spanning trees," in *Proceedings of the British Machine Vision Conference*, BMVA Press, 2008.

[104] K.-C. Chan, C.-K. Koh, and C. S. G. Lee, "A 3D-point-cloud feature for human-pose estimation," in *IEEE International Conference on Robotics and Automation*, pp. 1615–1620, May 2013.

[105] D. Kuan, A. Sawchuk, T. Strand, and P. Chavel, "Adaptive noise smoothing filter for images with signal-dependent noise," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 165–177, Mar. 1985.

[106] W. Zucchini and I. MacDonald, *Hidden Markov Models for Time Series: An Introduction Using R*. Chapman & Hall/CRC, 2009.

[107] G. Wasson, D. Kortenkamp, and E. Huber, "Integrating active perception with an autonomous robot architecture," *Robotics and Autonomous Systems*, vol. 29, no. 2-3, pp. 175–186, 1999.

[108] R. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *IEEE International Conference on Robotics and Automation*, pp. 1–4, May 2011.

[109] M. Andriluka, S. Roth, and B. Schiele, "Monocular 3d pose estimation and tracking by detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 623–630, Jun 2010.

[110] K. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[111] K. Shoemake, "Animating rotation with quaternion curves," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 245–254, 1985.

[112] L. Breiman, "Bagging predictors," in *Machine Learning*, vol. 24, pp. 123–140, 1996.

[113] J. F. C. Kingman, *Poisson processes*, vol. 3 of *Oxford Studies in Probability*. New York: The Clarendon Press Oxford University Press, 1993.

[114] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[115] B. Gawronski and F. Strack, *Cognitive Consistency: A Fundamental Principle in Social Cognition*. Guilford Publications, 2012.

[116] K.-C. Chan, C.-K. Koh, and C. S. G. Lee, "Using action classification for human-pose estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1176–1181, Nov. 2013.

[117] K.-C. Chan, C.-K. Koh, and C. S. G. Lee, "A 3D-point-cloud system for human-pose estimation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 11, pp. 1486–1497, 2014.

[118] C. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[119] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Networks*, vol. 13, pp. 411–430, May 2000.

[120] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *International Conference on Machine Learning*, ICML '07, (New York, NY, USA), pp. 759–766, ACM, 2007.

[121] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," in *Advances in Neural Information Processing Systems* (B. Schölkopf, J. Platt, and T. Hoffman, eds.), pp. 801–808, MIT Press, 2007.

[122] A. Jain, J. Tompson, Y. LeCun, and C. Bregler, "Modeep: A deep learning framework using motion features for human pose estimation.," in *CoRR*, 2014.

[123] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Transactions on Automatic Control*, vol. 26, pp. 17–32, Feb. 1981.

[124] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *SCIENCE*, vol. 290, pp. 2323–2326, 2000.

[125] W. Martinez, *Exploratory Data Analysis with MATLAB (Computer Science and Data Analysis)*. Chapman & Hall/CRC, Nov. 2004.

[126] K.-C. Chan, A. Ayvaci, and B. Heisele, "Partially occluded object detection by finding the visible features and parts," in *IEEE International Conference on Image Processing*, pp. 2130–2134, Sep. 2015.

[127] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, Jun. 2008.

[128] J.-B. Huang and M.-H. Yang, "Estimating human pose from occluded images," in *Asian Conference on Computer Vision*, vol. 5994, pp. 48–60, 2010.

[129] 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*, vol. 87, pp. 4–27, Mar. 2010.

[130] K.-C. Chan, C.-K. Koh, and C. S. G. Lee, "Selecting best viewpoint for human-pose estimation," in *IEEE International Conference on Robotics and Automation*, pp. 4844–4849, May 2014.

VITA

VITA

Kai-Chi Chan was born on June 7, 1986 in Hong Kong. He received his B.Eng. degree with first class honors in Computer Engineering and M.Phil. degree in Computer Science and Engineering from the Chinese University of Hong Kong in 2008 and 2010, respectively. Since August 2010, he has been working toward his Ph.D. degree in the School of Electrical and Computer Engineering at Purdue University. As a graduate student, he was a research assistant in the Assistive Robotics Technology Laboratory (ARTLab). He is a student member of IEEE and a member of the IEEE Robotics and Automation Society. His research interests are in the areas of computer vision and machine learning.