博士論文

Human Behavior Analysis for
Customer Interest Level Estimation using In-store Cameras
（店舗カメラを用いた顧客関心度推定のための
人物姿勢検知に関する研究）

東京大学大学院
情報理工学系研究科
電子情報学専攻

48-14645　**劉景文**

指導教員　**上條俊介**　准教授

© Copyright 2017, Jingwen Liu.
All rights reserved.

Abstract

The analysis of customer behavior is one of most important open topic for retailers, because the customer behavior information can reveal the customer interest level to the merchandise and is helpful to increase the commercial benefit. Conventionally, retailers can collect the records of cash registers or credit cards to analyze the customer buying behavior. However, this information cannot reveal the “not buying” behaviors of customer. The customer behaviors inside the retail store are still in the black box. Therefore, this dissertation applies surveillance camera to analyze customer behavior for customer interest level extraction.

In order to assess the customer interest level, this dissertation focuses on 3 main parts: head and body orientation detection, pose estimation and arm action classification. These 3 parts reveal increasing customer interest level. When the customer has an initial interest in some products, he or she probably looks at or turns to merchandise shelf. The head and body orientation can tell these behaviors. If the interest level becomes higher, the customer may bend over or squat down to look at the items carefully. Thus the pose information including the joint positions also needs to be estimated. Additionally, the customer arm actions show further information, especially for the interactions with products, such as touching, taking or returning. Therefore, the semantic arm action classification is essential.

In the customer head and body orientation detection part, the orientation is discretized into 8 classes to estimate whether the customer is looking or turning to the merchandise shelf. Conventionally, the orientation can be predicted by Supervised Learning (SL) method. However, the ground truth in retail store is usually difficult to obtain and thus the orientation annotations are usually decided subjectively. There may be some inaccurate orientation annotations in the training data. To solve this problem, a Semi-Supervised Learning (SSL) method is proposed to reduce the inaccurate annotations. In SSL, the training dataset is annotated as strongly labeled data and weakly labeled data. The training data with ambiguous orientation are firstly given a weak label, while other training data are given a strong label. The SSL optimizes the classifier by categorized the weakly labeled data in iteration. This dissertation also incorporates the state-of-art structure ResNet as the classifier. According to what the author knows so far,

there is no research which applies deep learning to estimate head and body orientation. Additionally, the temporal constraint and the human physical model constraint are jointly considered in body and head orientation estimation. The experiments show that the deep learning + SSL methods outperform the conventional feature based methods and feature + SSL methods.

In the pose estimation part, the essential idea for pose estimation is to incorporate the pose with body orientation, joint connections and visibility mask. Firstly, a customer pose estimation network integrating different useful information is proposed, named Integral Pose Network (IntePoseNet). This system firstly generates initial joint heatmaps using Fully Convolutional Networks (FCN). Based on these heatmaps, a series of body orientation based message passing layers construct a Deformable Parts Model (DPM) to model the local joint connections. At the output end, the joint positions and visibility mask are calculated from the joint heatmaps. Secondly, a multi-task neural network is proposed to simultaneously output the joint heatmaps, joint connection maps, body orientation and visibility mask. In the multi-task learning, the tasks can improve the performances each other. The experiments show that the multi-task neural network outperforms the conventional DPM method, deep learning + DPM method, the state-of -art deep learning methods and IntePoseNet.

In the arm action classification part, this dissertation defines 3 semantic arm actions: *touching*, *picking and returning* to shelf and *picking and putting* into basket, which reveal increasing customer interest to the product. As for the arm action classification, a novel Combined Hand Feature (CHF), which includes hand trajectory, tracking status and the relative position between hand and shopping basket, is proposed to classify different arm actions. The CHF sequence is given input into a Dynamic Bayesian Network (DBN) to classify the arm actions. The experiments prove that the CHF outperforms the traditional algorithm using HOG + SVM.

Acknowledgements

I would like to thank my supervisor Associate Professor Kamijo Shunsuke. Your foresight and wisdom construct the base of my research. I appreciate very much that you provided me the research guidance, resources and great platforms. The most important things that you taught me are the methodologies and philosophies about research. That will benefit all my life.

I would like to specially thank the researcher of Kamijo Laboratory, Doctor Yanlei Gu. You gave me quite a number of advices and helps, from paper writing, time arrangement, to problem analysis and experiments design.

Besides, the other members of Kamijo Laboratory also helped me a lot. I would like to thank Haitao Wang, Yongjie Liu and Qianlong Wang for their help for labeling data. I would like to thank Yongjie Liu, Qianlong Wang, Xu Liu and Doctor Lida Hsu. The fruitful discussions with you gave me many inspirations and improved this research much. I also would like to thank Feiyu Chen, Kenichi Sunagawa, Jiali Bao, Yuyang Huang, Yuichiro Wada, Yoriyoshi Hashimoto, Lijia Xie, Prarthana Bhattacharyya, Ehsan Javanmardi, Mahdi Javanmardi, Hirotsugu Kitamura, Dailin Li and Ya Wang. Thank all of you for your helps and accompaniments to my doctorate student life.

At last, I would like to thank my parents. Without your strong support, I cannot proceed with my studies.

Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1. Background

The analysis of customer behavior is one of most important open topic for retailers. The extracted customer behavior information allows to enhance customer experience, optimize store performance, reduce operational costs, and ultimately improve profitability. Traditionally, retailers use the records of cash registers or credit cards to analyze the buying behaviors of customers. However, this information cannot analyze the "not buying" behaviors of customer. The customer behaviors inside the retail store are still in the black box. For example, a customer stops on the front of the merchandise shelf but does not pick anything, or a customer picks an item but then returns it to the shelf. These behaviors can reveal the customer interest level to the merchandise. Therefore, the customer behavior analysis by using surveillance camera draws more and more attentions of researchers.

Conventionally, retailers hire manual labors or consulting companies to watch the surveillance video and extract useful marketing information, such as the number, frequencies, ages and the dwell time of customers. By observing the customer behaviors, the video analyzers are able to extract many useful information. However, watching the huge amounts of surveillance video is a very boring and time-consuming work. Therefore, how to automatically analyze customer behaviors becomes an important topic.

There are many commercial products or services for automatically customer analysis. For example, *ABEJA Inc.* [1], *IntelliVision* [2], *SEQ Security Surveillance Service Inc.* [3] provide the services about store heatmap analysis, customer counting, age analysis, dwell time analysis and so on. However, the above services still do not look individual customer in detail. The above information is not enough to reveal the customer interest level to the merchandises.

Then a natural question is what kind of information is needed for customer interest level estimation. Thinking the process of shopping, when the customer starts to be interested in some products, he or she probably stops on the front of the merchandise shelf and turns to the products. Head and body orientation estimation is a necessary part of the estimation system. If the interest level is higher, the customer may bend over or

squat down to look at the items carefully. Thus the pose information is also needed to be extracted. The pose information should contain each joint position of customer, such as head, shoulder, hand and foot etc. In addition, the actions of the arm and hand show further information, especially the interactions with products, such as touching and grabbing. Therefore, the hand and arm detection and tracking are supposed to be significant for shopping behavior analysis. Moreover, the purchase event probably happens when the customer puts products into the shopping basket. This interaction between the hand and basket needs to be recognized in the proposed method as well.

Therefore, this dissertation focuses on 3 main parts about customer behavior analysis: head and body orientation detection, pose estimation and arm action classification. The Chapter 1.2 will give an overall review of the researches about customer behavior analysis, including the relative works of above 3 main parts.

1.2. Related Works

Overall Customer Behavior Analysis

Generally, the customer counting and trajectory analysis are the most basic tasks of customer behavior analysis. Haritaoglu *et al*. [4] proposed a system for counting shopping groups waiting in checkout lanes and Leykin *et al*. [5] used swarming algorithms to group customers throughout a store into shopping groups. Senior *et al*. [6] analyzed the trajectory of individual customer from surveillance camera to find out the hot zone and dwell time of customers. Moreover, Popa *et al*. [7] found that trajectory analysis consisting of walking pattern and speed provided the first indication of the type of customer behavior. Those previous works indicated what valuable information in the surveillance video could be extracted by automatic technologies instead of human labor works.

However, the customer shopping behaviors are more diverse, such as: stopping before merchandise shelf, browsing, picking an item, reading the label, returning an item to the shelf or putting it into the shopping cart. Those different behaviors or the combinations of them show much richer marketing information. Hu *et al*. [8] proposed an system to detect the interactions between customer and the merchandise on the shelf.

Haritaoglu *et al*. [9] applied stereo camera in top view to recognize shopping actions of customers. Lao *et al*. [10] used one surveillance camera to recognize customer actions, such as pointing, squatting, raising hand and so on. Combining these actions, they could also recognize some events like robbery event. Haritaoglu *et al*. [11] also analyzed customer sight line to study whether they are watching advertisements on the billboard or the new product promotion.

The above researches focused on the applications using one surveillance camera. Besides using single surveillance camera, there are also many approaches which combine multiple sensors to recognize customer behavior in detailed. Migniot *et al*. [12] used a top-view camera and a side-view Kinect to track customer postures in 3D space. Popa *et al*. [7] and Sae-ueng *et al* [13] established an experimental ubiquitous shop respectively to recognize different types of customer behaviors. In the experimental shop, one ceiling fisheye camera, multiple shelf cameras and RFID readers or one microphones were combined to sense customer behavior. Stan *et al*. [14] proposed an intelligent store which can automatically send information to customers' PDA, according to customers' different positions. They applied RFID inside the store and GPS outside the store to locate the customers' positions.

From the above researches, obviously, by combining more and more sensors, customer behavior can be analyzed more and more detailed. However, these extra sensors mean extra costs. Thus this dissertation argues that the methods which use too many sensors are hardly widely applied in the existed retail stores. In order to save the cost in practical applications, this dissertation focuses on single surveillance cameras, which are already widely installed in retail stores, to analyze customer behaviors.

Head and Body Orientation Detection

Head and body orientation estimation is studied by many researchers in different scenes. Lee *et al*. [15] developed a system to extract the head and body part location for multiple people from surveillance camera. Abe *et al*. [16] detected pedestrians' head orientations from surveillance camera, even if the head region was very small in the video. Schulz *et al*. [17] [18] proposed a framework for head localization and orientation detection using sliding window method. Gandhi *et al*. [19] used Histogram of Oriented

Gradients (HOG) [20] and Support Vector Machines (SVM) to recognize the body orientations of pedestrians. Goffredo *et al*. [21] proposed a viewpoint independent gait recognition method by using in multi-camera surveillance. Chen *et al*. [22] proposed an approach that jointly detected body location and body pose in monocular surveillance video. In the above researches, Supervised Learning (SL) algorithm is widely employed in [15] [16] [17] [18] [19] [22].

However, the ground truths of training data in SL are usually difficult to obtain and the annotations are usually decided subjectively. There may be some inaccurate labels in the training data. In order to solve this problem, this dissertation proposes a Semi-Supervised Learning (SSL) method to detect the customer head and body orientation. The training data with ambiguous orientation will be firstly given a weak label, while other training data with clear orientation will be given a strong label. The SSL method will optimize the classifier by categorized the weakly labeled data in iteration. HOG + SVM is used as the classifier in the SSL. This dissertation also applies the deep neural network ResNet [23] as the classifier in SSL. The experiments show that the deep learning based SSL outperforms the HOG based SSL and conventional SL. As far as the author knows, there is no papers using deep learning method to detect orientation.

Pose Estimation

In order to estimate the human pose, researchers proposed many approaches based on different information. One of these approaches is using human silhouette to estimate the pose or action. From silhouette, model-based methods are commonly employed for pose estimation [24] [25] [26]. There are also methods which analyze silhouettes from multiple views to generate 3D human pose [27] [28] [29].

Another way for pose estimation is to utilize the temporal information in video. One kind of the methods is based on detection and tracking, which detects a rough pose in the initial frame and track it in every continuous frame [30] [31] [32] [33]. The spatio-temporal Markov Random Field (MRF) model is also generally applied for pose estimation in frame sequence [34] [35] [36]. This model connects not only the intra-frame joints but also the inter-frame joints. Based on the spatio-temporal MRF, Cherian *et al*. [37] split the pose into limbs and then recomposed them to pose after optimization.

A more challenging topic is to estimate human pose from a single image. It requires to detect human from variant background and estimate the pose in a large number of degrees of freedom. General techniques are using Pictoral Structures [38] [39] [40] or Deformable Part Models (DPM) [41] [42] [43]. In another way, Pishchulin *et al*. [44] introduced a poselet detector to the Pictorial Structure to establish a more flexible model.

In recent years, the Convolutional Neural Networks (CNN) [45] [46] achieved high performance in different tasks of computer vision. AlexNet [47], VGG [48], GoogLeNet [49] and ResNet [23] etc. achieved higher and higher accuracy in image classification task. In pose estimation task, a pioneering paper is submitted by Toshev *et al*. [50]. They proposed DeepPose to estimate human pose by multiple cascade CNNs. However, there are some limitations of their method. Firstly, if the initial estimation is far from the ground truth, it is difficult to correct the estimation in the cascade structure. Secondly, their method only gives one prediction per image. There is no candidate to improve the prediction. To address the first problem, Carreira *et al*. [51] and Haque *et al*. [52] applied feedback structures to optimize the pose estimation in 2D and depth image respectively. Their methods feed the estimated pose to the input end and gradually refine the result in iteration. To address the second problem, a solution is using the probabilistic heatmaps of joints. One kind of methods is to train the CNN with local image patches and tested by sliding window to generate the joint heatmaps [53] [54]. That means the joint estimation problem is converted to a pixel-wise classification problem. In these pixel-wise classification methods, the training and testing are time consuming. Besides, choosing the local training image patches is tricky. To cover this shortage, another method is to train the Fully Convolutional Networks (FCN) with full images [55] [56].

The above methods still have not considered the spatio-temporal restrictions of pose. Because the human joints cannot locate at arbitrary positions in practical, the local joint connections need to be modeled. Yang *et al*. [57] and Chu *et al*. [58] stacked multiple message-passing layers on top of the CNN to establish a Deformable Parts Model. On the other hand, in order to model the temporal relationship in sequence, Jain *et al*. [59] fused image patches with optical flow [60] to predict the pose in the next frame. Pfister *et al*. [61] proposed a temporal pooling method to optimize the pose estimation by combining both past and future heatmaps.

In order to handle the occlusion, Azizpour *et al*. [62] and Ghiasi *et al*. [63] learned templates for occluded versions of each body part. Umer *et al*. [64] incorporated the context information of occluding objects to predict the locations of occluded joints. Haque *et al*. [52] implicitly learned the occlusion through a deep neural network and gave the output of visibility mask of joints.

Differing from the above researches, the pose estimation in retail store has its own characteristics. In the retail store environment, the customer pose and body orientation are highly related with each other. For example, when the customer is stretching arm to pick an item, he or she must face to the merchandise shelf. On the other hand, when the customer body orientation is parallel to the shelf, this customer is probably just walking through. These facts imply that the customer body orientation information is helpful for the pose estimation.

Considering the researches above and the situations of retail store, this dissertation proposes to incorporate the customer pose estimation with body orientation, visibility mask and joint connections. One of the proposed system is a network integrating all of the useful information, called Integral Pose Network (IntePoseNet). This system firstly generates initial joint heatmaps using Fully Convolutional Networks (FCN). Based on these heatmaps, a series of body orientation based message passing layers construct a Deformable Parts Model to fine-tune joint heatmaps. Besides, this network includes the dense optical flow to improve the estimation of moving body parts. By observing the surveillance video of retail store, the main occluding object of customer is the shopping basket. Thus this system also incorporates the basket detection to improve the invisible joint detection. In addition, the system employs a Bidirectional Recurrent Neural Network (BRNN) [65] on top of the network to improve the accuracy by utilizing the forward and backward frame sequences. Therefore, in this system, the global body orientation, local joint connections, motion features, occluding objects and temporal pose continuity are all integrally considered.

Another thinking is to apply the multi-task learning [66] to learn these elements simultaneously. In recent year, many multi-task learning structures were proposed as deep neural networks. Zhang *et al.* [67] proposed to combined the face landmark detection with the detections of wearing glasses, gender, smiling and face orientation.

Tian *et al.* [68] jointly optimized pedestrian detection with semantic tasks, including pedestrian attributes (e.g. 'carrying backpack') and scene attributes (e.g. 'vehicle', 'tree', and 'horizontal'). Yim *et al.* [69] proposed a two tasks network. The main task is rotating face. The other auxiliary task is reconstructing the input image after the main task. Kao *et al.* [70] incorporated the aesthetic quality assessment task with semantic recognition task. Cao *et al.* [71] proposed to learn the joint heatmaps and joint connection vector fields for multi-person pose estimation. The authors adopted a kind of bottom-up method. They firstly generated the joint positions of all of the people in the image by finding the local maximums of joint heatmaps. Then they connected these joints gradually to construct the person pose structures. This method performs well for the visible joints. However, there are many false positives for the occluded joints, because of its aggressive searching of joint positions. Although the implementations are different above, all of the researches tried to combine the tightly related tasks together, because the implicit relationships between these tasks can improve each other in the multi-task learning.

Therefore, a multi-task neural network is proposed for pose estimation. This network simultaneously outputs the joint heatmaps, joint connection maps, body orientation and visibility mask. These 4 tasks share a backbone of FCN and produce the outputs from corresponding specific output layers. The 4 tasks can improve the performance each other in the multi-task learning. Compared with [71], this method reduces the false positives by introducing the visibility mask.

Action Classification

As for action classification, Zelnik-Manor *et al*. [72] and Schüldt *et al*. [73] proposed to classify simple actions including walking, waving and clapping etc. in controlled conditions and opened their datasets, which are well known as Weizmann dataset and KTH dataset respectively. Based on Weizmann dataset, KTH dataset and both of them respectively, Liu *et al*. [74], Niebles *et al*. [75] and Benmokhtar *et al*. [76] applied different spatio-temporal features to classify these actions. Besides these datasets, Trinh *et al*. [77] recognized the hand motion of cashiers in retail stores in top view to detect the fraudulent behaviors of cashiers. Ryoo *et al*. [78] [79] extracted 3D XYT features to detect behaviors between two people, such as hugging, hand shaking, fighting

and so on. The HOG + SVM methods are also widely used in action recognition [80][81][82]. Except for HOG, more complex features are also applied with SVM [83] [84]. Beside these features, Elmezain *et al*. [85] developed a trajectory-based hand gesture recognition system for Arabic numbers (0-9) by using Hidden Markov Models. Shechtman *et al*. [86] used a small video clip including one behavior to query the similar behavior in a large long video. This method does not need training data and prior learning. Chen *et al*. [87] combined both global and local cues, such as body inclination, contour and speeds of different body parts, to describe human behavior. Then they employed Dynamic Bayesian Network (DBN) [88] to classify human behaviors. Zhu *et al.* proposed to use both motion and context features + Conditional Random Field (CRF) methods for action classification [89].

This dissertation proposes to classify 3 types of arm actions in real retail store environment: *touching*, *picking and returning* to shelf and *picking and putting* into basket, which reveal increasing customer interest to the product. These actions are especially important for marketing analysis. Although detecting and tracking the picked item is the most direct method to classify these 3 actions, the picked item is sometimes too small to recognize and is easily occluded by customer body. Therefore, this dissertation proposes a novel Combined Hand Feature (CHF) instead, which includes hand trajectory, tracking status and the relative position between hand and shopping basket, to describe the arm action in each frame. Then a Dynamic Bayesian Network (DBN) is employed to classify these CHFs into different arm actions.

1.3. Dissertation Organization

The rest of dissertation is organized as below.

Chapter 2 introduces the framework of the customer behavior analysis system and gives the definitions of inputs and outputs.

Chapter 3 discusses the customer head and body orientation detection. A SSL method are proposed to solve the ambiguity problem of training data. Two kinds of classifiers are implemented in the SSL: HOG + SVM classifier and ResNet classifier. Using the classifier generated by SSL, the head and body orientation will be detected

and tracked in image sequence.

Chapter 4 proposes the IntePoseNet for customer pose estimation. For comparison, the structures of pose estimation systems are introduced from simple to complex. Firstly, the conventional DPM is introduced. The conventional DPM can be seen as a structure of Heatmap by HOG + Message Passing. Then a trivial method using deep learning is introduced. This kind of methods directly give the output of joint position through a deep neural network. The main disadvantage of this kind of methods is that there is only one prediction per input image. Once the result is inaccurate, it is difficult to optimize based on only one prediction. In order to increase the predicted candidates, a FCN is applied to generate joint heatmaps. Based on the heatmaps, each component of the IntePoseNet is incorporated to construct an organic whole.

Chapter 5 describes the solution of hand action classification. A novel Combined Hand Feature (CHF) is proposed to combine hand trajectory, tracking status and the relative position between hand and shopping basket, to describe the arm action in each frame. Then the feature sequence is given input into a DBN to classify into different arm actions.

Finally, Chapter 7 concludes the works of this dissertation and discusses the future works.

1.4. Privacy Declaration

In this dissertation, the customer images are not used for personal information identification. The faces of customers which appear in the figures are blurred for the purpose of privacy. The logos or texts in the retail store are also blurred to prevent identification. This research is permitted by the Compliance Committee of the University of Tokyo.

Chapter 2. Framework of Customer Behavior Analysis System

The framework of the customer behavior analysis system is shown as Figure 2.1. In the input end, taking the advantage of surveillance camera, the background of video can be easily subtracted by using Multi-Layer Background Subtraction algorithm **[90]**. In the rest of this dissertation, except for special notice, all of the algorithms are using the images after background subtraction, although some figures keep the background for clarity. Figure 2.2 illustrates the process of background subtraction.

Based on the images after background subtraction, the head and body orientation detection are applied. The orientation is divided into 8 classes for every 45° angle. Figure 2.2 shows the representations of head and body orientations. Mathematically, the orientation is an 8 × 1 vector. The *n*th element means the probability of the *n*th label.

On the other hand, the silhouette image will be used to detect whether there is a stretching arm. The stretching arm is a trigger of hand action classification. Once the stretching arm is detected, the hand area will be tracked and be extracted the Combined

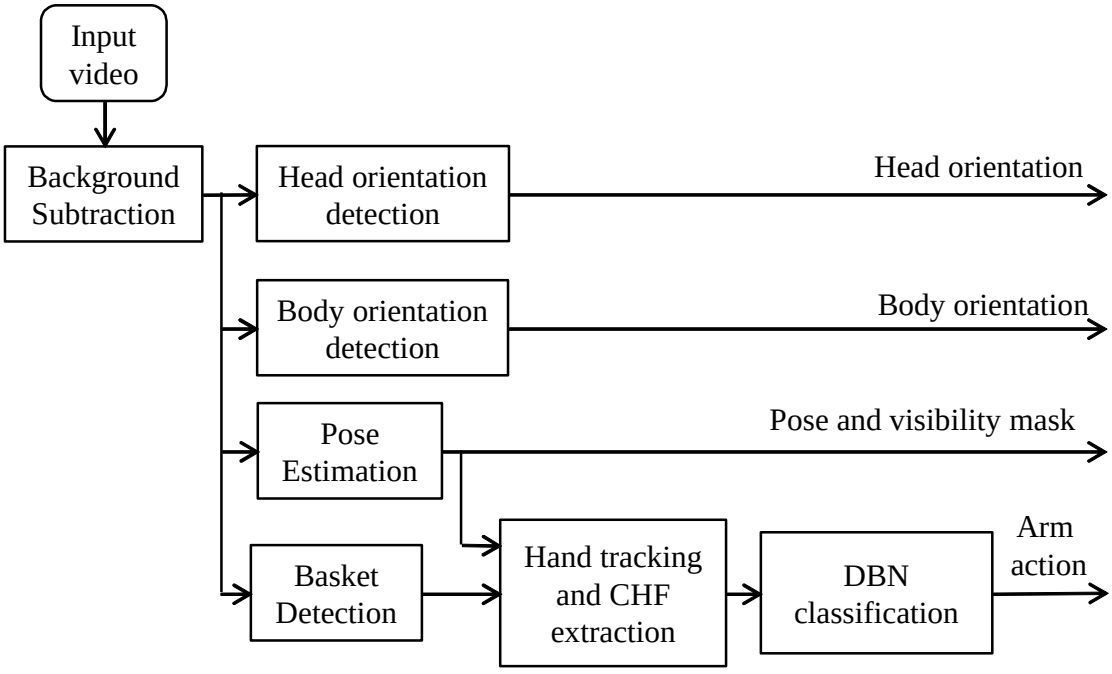
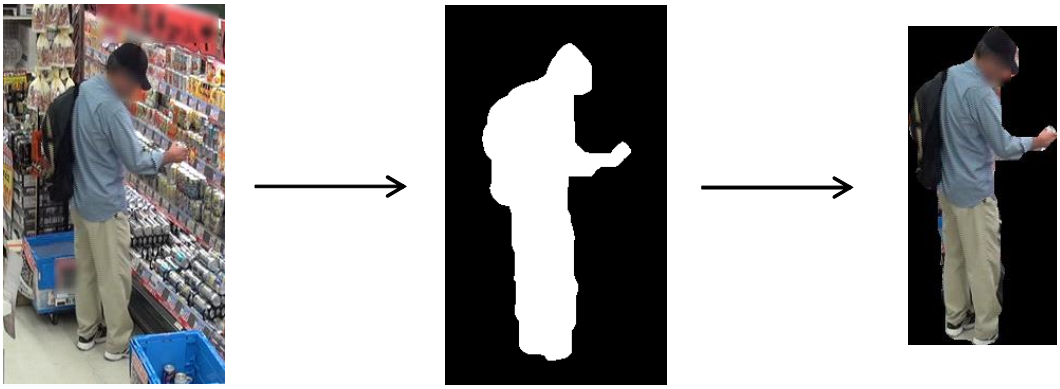


Figure 2.1 The framework of the customer behavior analysis system



(a) (b) (c)

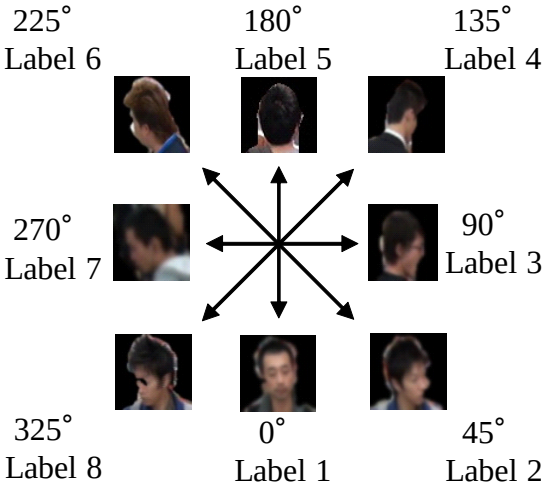
Figure 2.2 The process of background subtraction: (a) The input image; (b) The silhouette after background subtraction; (c) The customer image after background subtraction

Hand Feature (CHF). The hand position from the pose estimation would be the observation in the hand tracking. Considering the interaction between hand and shopping basket is an important clue for the hand action classification, the basket detection result is also used to calculate the CHF. The CHF is then given input into a DBN and is classified into 3 types arm actions: *touching*, *picking and returning* and *picking and putting*, which reveal increasing customer interest to the product. The definitions and examples of these 3 actions are shown in Table 2.1 and Figure 2.4 respectively.

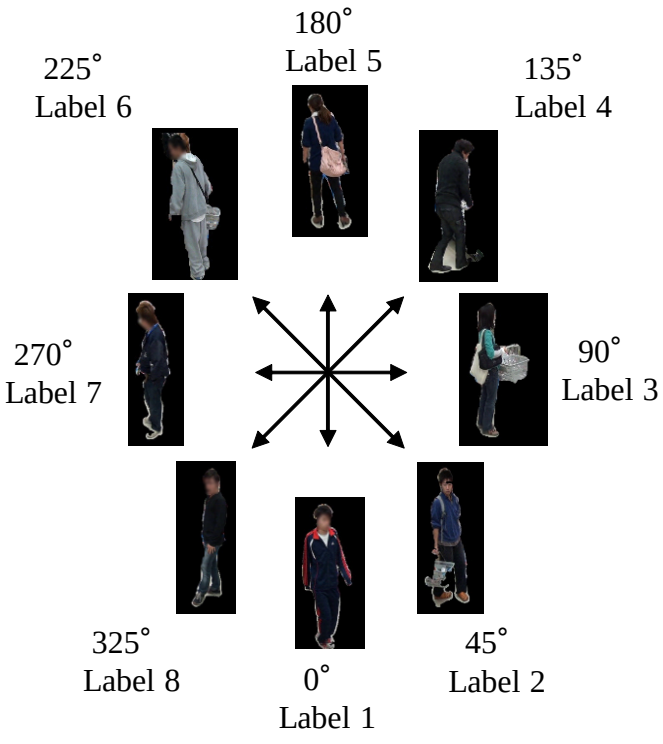
The pose estimation part receives the background subtracted image as inputs. The output is the positions and the visibility mask of 14 joints, which is shown as Figure 2.5. The 14 joints include: head top, head bottom, left/right shoulders, left/right elbows, left/right wrists, left/right hips, left/right knees and left/right ankles. Therefore, the pose is represented as a 28 × 1 vector: $[x_1, y_1, \dots, x_{14}, y_{14}]^T$, where x_n, y_n is the coordinate of the *n*th joint. The visibility mask is a 14 × 1 vector: $[v_1, v_2, \dots, v_{14}]^T$, where v_n is the probability of visibility of the *n*th joint. v_n = 1 means the joint is totally visible and v_n = 0 means the joint is totally occluded. Note that there are usually two types of annotations of pose: Observer-Centric (OC) annotation and Person-Centric (PC) annotation. In the OC annotation, the limbs in the left/right side of image mean the left/right limbs, while in the PC annotation, the left/right limbs of person are the left/right limbs. This dissertation chooses the PC annotation. According the PC annotation, the skeleton of Figure 2.5 is assumed in back view. That means the red arm and purple leg

are the left ones, the blue arm and cyan leg are the right ones.

After defining the inputs and outputs of the system, the following 3 chapters will discuss the head/body orientation detection, pose estimation and arm action classification respectively.



(a)



(b)

Figure 2.3 The representations of (a) head orientations and (b) body orientations

Table 2.1 The definitions of customer arm actions

Arm action	Definition
Touching	The customer touches a product but does not pick it from the shelf.
Picking and returning	The customer picks a product then puts it back to the shelf.
Picking and putting	The customer picks a product then puts it into the shopping basket.

(a)

(b)

(c)

Figure 2.4 Examples of different customer action actions: (a) Touching; (b) Picking and returning; (c) Picking and putting

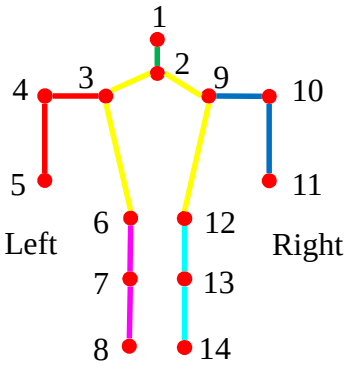


Figure 2.5 The representation of pose

Chapter 3. Customer Head and Body Orientation Detection

3.1. Overview

The customer head and body orientation reveal the customer initial interest level to merchandise. Generally, the positions of merchandise shelves are already known. Thus the head and body orientation can tell whether the customer looks at or turn to the shelves. In another word, for a given customer head image or a given customer full body image, the head or body orientation detection became a classification problem. As the discussion in Chapter 2, the customer full body image can be obtained by background subtraction. However, the head position still needs to be detected. Therefore, this chapter will discuss how to classify the head and body orientation for given head and body image, and then discuss how to jointly conduct the body orientation classification, head position detection and head orientation classification in the video.

In this chapter, firstly the conventional HOG + SVM method is introduced for classification. Secondly, the recent Deep Convolutional Neural Network (DCNN) based classification is introduced. The DCNN can automatically extract the features. Thus the features do not need to be specify in the DCNN. In recent year, the DCNNs are reported to outperform the conventional HOG + SVM method in various tasks. However, there is still a problem in the retail store environment. Whether the HOG + SVM or DCNN is a kind of Supervised Learning (SL) method. All of the training data needs to be label the ground truth by human. The problem is that, unlike in the controlled experimental scenes, orientation labeling of customers are no objective standard. There may be labeling error for the ambiguous orientation. Therefore, this chapter proposes a Semi-Supervised Learning (SSL) method to solve this problem. Moreover, the SSL algorithm is designed as the extension of in any kind of SL algorithm. This chapter proposes the SSL algorithm in both HOG + SVM and DCNN methods. After obtaining the classifier from the SSL training, this chapter proposes the joint head and body orientation detection and tracking. The physical constraints of body orientation and head orientation are considered in the joint detection. Finally, a series of experiments are conducted to evaluate the different methods. The experimental results show that the SSL based on DCNN achieves highest accuracy.

3.2. Head and Body Orientation Classification using Conventional Supervised Learning

3.2.1. HOG + SVM method

The HOG + SVM method is firstly proposed by Dalal and Triggs [20] for pedestrian detection. Then this method became widespread and was applied in different tasks [74] [75] [76]. This chapter will introduce the HOG + SVM in the head and orientation classification.

Histogram of Oriented Gradient (HOG) feature

The main idea of the Histogram of Oriented Gradient (HOG) feature is to concatenate the local gradient descriptor to represent the image. In order to calculate the HOG, the first step is to divide the image into small rectangle or radial connected regions, which are named cells. Then a histogram of gradient directions is calculated for each cell. At last the HOG feature is constructed by concatenating of these histograms. For improving performance, these histograms can be contrast-normalized by calculating the average of the intensity in a larger region of the image, which is named a block. Then all of cells in the block are normalize by this average value. The normalization can achieve better illumination or shadowing invariance.

The HOG feature becomes widely used because of its multiple key advantages comparing with other features. Because the histogram is calculated on local cells, it maintains the geometry invariance and illumination invariance, except the object orientation invariance. Note that the orientation here does not mean the head and body orientation like Figure 2.3, but the object rotation in the image. The rotation changes only appears in more global area. Additionally, as Dalal and Triggs [20] found that, HOG feature be invariant on individual body movement of pedestrians by applying coarse spatial area division, fine orientation division, and strong local illumination normalization, as long as the pedestrians maintain a roughly upright position. Therefore the HOG feature is particularly suited for human detection in images.

The HOG feature calculation starts from the gradient computation in cells. For many other feature, the first step for feature calculation is pre-processing. Generally, the

image will be normalized by color and gamma values. However, Dalal and Triggs [20] indicated that, the pre-processing can be omitted in HOG feature calculation. The reason is that the normalization has essentially been done in the local feature normalization in HOG calculation. Rather than pre-processing, the HOG feature calculation starts from the local gradient value computation. Mathematically, the local gradients are calculated by $[-1, 0, 1]$ and $[-1, 0, 1]^T$ filter kernels in the gray image.

In [20], the authors tested other more complex and larger filter kernels. However, all of these filter kernels provided lower accuracies in human detection. They also tested the image pre-processing like Gaussian smoothing at the beginning, but also cannot find any improvement in practice.

The local cell histograms calculation is the next step. For each pixel in the cell, a weighted voting is conducted to construct an orientation-based histogram. The shape of cells can be either rectangular or radial. The grids of histogram channels are the average division of 180 degrees for unsigned gradient, or 360 degrees for signed gradient. Thus the gradients can be into one of the histogram channels based on their orientations. In [20], the authors found that the best performance is achieved at 9 histogram channels for the unsigned gradients. Beside the weight of vote can be determined by either the magnitude of gradient, or other functions on the magnitude.

For the tolerance of the illumination changing and contrast changing, the gradient magnitudes need to be locally normalized. This requires dividing the cells into larger, spatially connected blocks. Then the HOG feature is concatenated all of the normalized cell histograms from all of the block regions. These blocks are generally overlapped, which means that each cell contributes more than one time to the final feature. Typically, there are two main block shapes: rectangular blocks named R-HOG and circular blocks named C-HOG. The R-HOG blocks are generally represented as square grids, where include three parameters: the number of cells per block, the number of pixels per cell, and the number of channels per cell histogram. From the experiments in [20], the optimal parameters were four 8 × 8 pixels cells per block, 2 × 2 pixels per cell and 9 channels per cell histogram. Moreover, the authors discovered that some minor improvements can be obtained by applying a Gaussian spatial filter in each block at first, because the gradients magnitudes are filtered less in the pixels far from the of the block center.

Besides, the calculations of R-HOG blocks are similar to the Scale-Invariant Feature Transform (SIFT) feature [91]. However, there are still differences. R-HOG blocks are calculated in dense local grids though the whole image at a constant scale without orientation invariance. On the other hand, SIFT features are generally calculated at some individual key points with scale invariance and orientation invariance. Additionally, the spatial information in R-HOG blocks are concatenated in order to describe the whole image, while SIFT features are generally used individually. The C-HOG blocks can be represented by four parameters: angular bins number, radial bins number, radius of the center bin and the expansion factor of the radial bins. According to [20], the optimal performance is archived when radial bins number is 2, angular bins number is 4, center radius is 4 pixels and an expansion factor is 2. Similarly, the Gaussian smoothing can be also omitted for C-HOG blocks. The C-HOG blocks and some shape context features [92] are constructed in similar forms. However, the cells in C-HOG includes multiple orientation channels. On the other hand, the calculation of shape contexts features only depends on the single edge presence.

There are 4 different methods for block normalization which were discussed in [20]. Assume *v* is the unnormalized vector including all histograms in a block, $\|v\|_k$ is its *k*-norm. Then the normalizations can be formulated as the following, where *e* is a small constant in order to avoid dividing zero.

L2-norm normalization: $v' = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$

L2-hys normalization: L2-norm normalization followed by limiting the maximum values of v' to 0.2 and one more the L2-norm normalization.

L1-norm normalization: $v' = \frac{v}{\|v\|_1 + \zeta}$

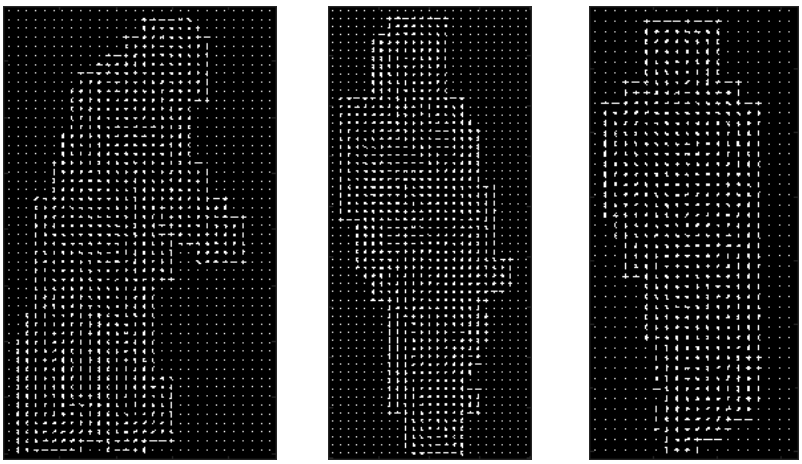
L1-square root normalization: $v' = \sqrt{\frac{v}{\|v\|_1 + \zeta}}$

In the experiments in [20], the authors found that the L2-hys, L2-norm and L1-square root normalization produced similar accuracies. Only the L1-norm produced slightly worse accuracies. Even though, all of the four methods significantly outperformed over the non-normalized data. The examples of the visualized HOG features after normalization are shown in Figure 3.1. The HOG features are visualized

in cell grids. In each cell, a polar diagram shows the line in each orientation, where the length of line represents the proportion of the cell histogram in that orientation.

The final step in object classification using HOG features is to give the features input into some pre-trained classification system, e.g. the Support Vector Machine (SVM) classifier. After training on the data including a certain specific object with HOG feature, the SVM classifier can distinguish whether the target appears in the additional test images. The SVM also can be easily extended to multiple object classification, such the head and body orientation.

(a)



(b)

Figure 3.1 The examples of visualized HOG features. (a) The input image for HOG feature calculation; (b) The corresponding visualized HOG features

Support Vector Machine (SVM)

Support Vector Machines (SVMs) are a kind of supervised learning algorithms which analyze data in classification task and regression task. As a simplest example, given training data which only include two categories, an SVM classifier can be trained to group new examples to one of the categories. This means that SVM is a kind of non-probabilistic 2-class classifier. For multiclass classification, the problem can reduce into multiple binary classification problems and then apply the SVM. An SVM model can be also seen as a plane in the training point space, which separates the different categories as wide as possible. Figure 3.2 shows an example of SVM in 2-D space. The classifier becomes a line in 2-D space. If the points is distributed in 3-D space, the classifier is a plane. For higher dimension, the classifier is also named hyperplane. As the HOG feature, the dimension of data could be thousands. Then new examples can be classified into one of the categories according to which side of the classifier they fall.

More generally, a SVM constructs a hyperplane in a high dimensional space, which can be applied for classification, regression, or other tasks. Intuitively, as shown in Figure 3.2, a good classifier should be a hyperplane which has the largest distance to the nearest training data point of any class, because in general the larger the distance the lower the testing error of the classifier. The distance is also called functional margin.

Although the original problem is stated in a finite dimensional data space, it often happens that the data set is not linearly separable in that space. For this reason, the

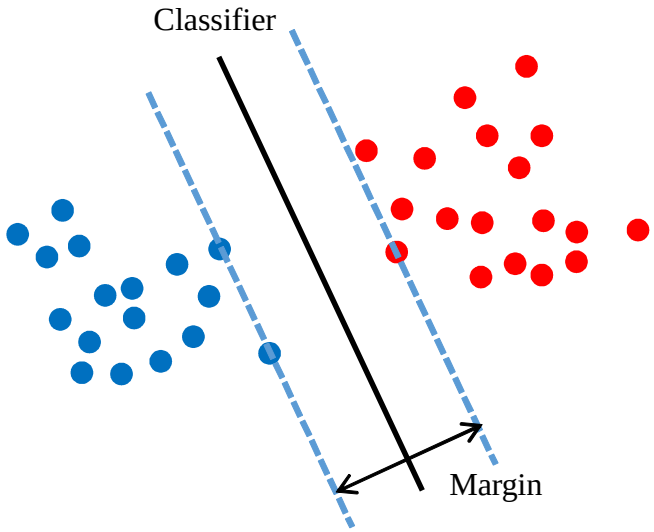


Figure 3.2 An example of SVM classifier in 2-D space

original finite dimensional data space is generally mapped into a much higher dimensional space, in order to making the separation linear in that space. To maintain the computational cost reasonable, the mappings used by SVM are designed to ensure that dot products can be calculated easily in terms of the variables in the original space, by defining them through a kernel function. The kernel function ensures that the hyperplane does not need to be calculated in the higher dimensional space directly, but only is represented as the set of points whose dot product with a vector in that space is a constant. The vectors which compose the hyperplanes can be chosen to be a linear combination with parameters α_i of the feature points x_i in the training data. Mathematically, with the optimal hyperplane, a testing point *x* in the feature space which are mapped into the hyperplane can be defined by the constraint $\sum_i \alpha_i k(x_i, x) = \text{constant}$, where the $k(x_i, x)$ is the kernel function. Note that the further *x* is away from x_i, the smaller kernel function is. This means that each term in the sum measures the degree of nearness of the point *x* to the corresponding point x_i in the training data set. In this form, the sum of kernels above can be applied to measure the relative closeness of the test point to the training data points resulting one of the two sets to be discriminated. Note the fact that the testing points set *x* which is mapped into any hyperplane can be quite complicated but allowing a discrimination between the categories, while the discrimination may be not convex in the original space.

Originally, the SVM is linear, which means the data do not need to be mapped into higher dimensional space by kernel function. Assume a training dataset is given as the forms $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, where the y_i is the category label with value 1 or -1, $\vec{x}_i$ is the feature vector with *p* dimension, for example the HOG feature. The target of linear SVM is to find the maximum margin hyperplane that divides the group of points $\vec{x}_i$ with label $y_i = 1$ from the group of points for with $y_i = -1$. In another word, the distance between the hyperplane and the closest point from either group is maximized. Therefore, the hyperplane can be written as below.

$$\vec{w} \cdot \vec{x} + b = 0 \tag{1}$$

where $\vec{x}$ is a set of points in the hyperplane, $\vec{w}$ is the normal vector of the hyperplane, *b* is the bias.

If the training data are linearly separable, there are two parallel hyperplanes can be constructed to just separate the two classes of data. From each hyperplane, the distance to the points in the other category is largest. As an example, these two hyperplanes can be seen as the dash line in Figure 3.2. The region which is bounded by these two hyperplanes is called as margin. Therefore, the maximum margin hyperplane is the hyperplane which locates the halfway between them, as the solid line in Figure 3.2. These two hyperplanes can be represented by the following equations.

$$\vec{w} \cdot \vec{x} + b = 1 \tag{2}$$

$$\vec{w} \cdot \vec{x} + b = -1 \tag{3}$$

The distance between these two hyperplanes can be easily calculated as $\frac{2}{\|\vec{w}\|}$. Therefore, in order to maximize the distance between these two hyperplanes, the target becomes to minimize $\|\vec{w}\|$. Moreover, to prevent the training data points from falling into the margin, there are following constraints.

$$\vec{w} \cdot \vec{x}_i + b \geq 1, \text{if } y_i = 1 \tag{4}$$

$$\vec{w} \cdot \vec{x}_i + b \leq -1, \text{if } y_i = -1 \tag{5}$$

These constraints ensure that each training data point must locate on the correct side of the margin. The constraints can be also rewritten as below.

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, \text{for all } 1 \leq i \leq n \tag{6}$$

Therefore, the learning target is to minimize $\|\vec{w}\|$ with the constraint equation (6). In the inference step, for a testing feature $\vec{x}$, the category label is determined as below.

$$\text{prediction} = \text{sign}(\vec{w} \cdot \vec{x} + b) \tag{7}$$

where the sign function is defined as below.

$$\text{sign}(x) = \begin{cases} 1, \text{if } x \geq 0 \\ -1, \text{if } x < 0 \end{cases} \tag{8}$$

From above discussion, geometrically, the max margin hyperplane is completely determined by those $\vec{x}_i$ which are closest to it. These $\vec{x}_i$ are also called support vectors.

To extend SVM to classify the data are not linearly separable, hinge loss function

as below can be applied for the soft margin.

$$\zeta_i = \max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i + b)\right) \tag{9}$$

This function is zero if the constraint equation (6) is satisfied. In other words, the hinge loss function does not strictly force the training data points falling in the correct side. If the point locates at the correct side of margin, the equation (9) is with equal meaning as constraint equation (6) . For data on the wrong side of the margin, the value of loss function is proportional to the distance from the margin.

Then the learning target is to minimize the following expression.

$$\left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i + b)\right)\right] + \lambda\|\vec{w}\|^2 \tag{10}$$

where the λ is a tradeoff parameter between increasing the margin size and ensuring that the $\vec{x}_i$ locates on the correct side of the margin. Therefore, if the value of λ is sufficiently small, the soft margin SVM becomes identical to the hard margin SVM if the training data are linearly separable. Therefore, more generally, here only introduces how to minimize the expression (10).

In order to minimize the (10), firstly substitute the expression (9) for expression (10) as following.

$$\frac{1}{n}\sum_{i=1}^{n} \zeta_i + \lambda\|\vec{w}\|^2 \tag{11}$$

The constraints becomes following.

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i \tag{12}$$

This is called the primal problem.

In order to solve the primal problem, Lagrangian dual is applied to transform the primal problem to a simplified dual problem. The dual problem is to maximize the below target.

$$f(c_1, \ldots, c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} c_i y_i \left(\vec{x}_i \cdot \vec{x}_j\right) c_j y_j \tag{13}$$

The constraints of the dual problem is shown as following.

$$\sum_{i=1}^{n} c_i y_i = 0 \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda}, \text{ for all } i \tag{14}$$

where the c_i is the learnable parameter and is determined by the below equation.

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \vec{x}_i \tag{15}$$

Because the dual maximization problem is a quadratic function of the c_i following linear constraints, it can be efficiently solved by quadratic programming algorithms. Furthermore, $c_i = 0$ means the $\vec{x}_i$ locates on the correct side of the margin, and $0 < c_i < \frac{1}{2n\lambda}$ when $\vec{x}_i$ exactly locates on the margin boundary. Therefore, the $\vec{w}$ is composed of a linear combination of the support vectors. Besides, the bias b can be calculated as following.

$$b = y_i - \vec{w} \cdot \vec{x}_i \tag{16}$$

Another way to deal with the not separable data is to apply the kernel function. This method is proposed by Boser *et al*. [93] The algorithm is formally similar as the original SVM, except that a nonlinear kernel function replaces every dot product. This allows the SVM to search the maximum margin hyperplane in a transformed feature space. This transformation may be nonlinear and the transformed space is generally with higher dimension. Although the classifier is a hyperplane in the transformed higher dimensional feature space, it may be nonlinear in the original training data space. However, it is noteworthy that the algorithm in the higher dimensional feature space increases the generalization error, although given enough training data can reduce the error in a degree [94].

There are some common kernel functions shown as below.

Polynomial (homogeneous): $k(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j)^d$.

Polynomial (inhomogeneous): $k(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d$.

Gaussian radial basis function: $k(\vec{x}_i, \vec{x}_j) = \exp\left(-\gamma \left\|\vec{x}_i - \vec{x}_j\right\|^2\right)$, for $\gamma > 0$.

Hyperbolic tangent: $k(\vec{x}_i, \vec{x}_j) = \tanh(\kappa \vec{x}_i \cdot \vec{x}_j + c)$, for some (not every) $\kappa > 0, c < 0$.

The essential thought behind the kernel trick is to avoid to calculating the transformation $\varphi(\vec{x}_i)$. The kernel function is generally the dot product in the transformed space $k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)$. Also, the $\vec{w}$ in the transformed space is

calculated as below.

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \varphi(\vec{x}_i) \tag{17}$$

where c_i is the learnable parameter. Therefore, the dot product between the $\vec{w}$ and feature point in the transformed space can be calculated as $\vec{w} \cdot \varphi(\vec{x}) = \sum_i c_i y_i \varphi(\vec{x}_i)\varphi(\vec{x}) = \sum_i c_i y_i k(\vec{x}_i, \vec{x})$. As a result, the transformation function $\varphi(\vec{x})$ actually does not need to be calculated.

Similar to the algorithm of soft margin, the learning target is to maximize the following formulation.

$$\begin{aligned} f(c_1, \ldots, c_n) &= \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} c_i y_i \left(\varphi(\vec{x}_i) \cdot \varphi\left(\vec{x}_j\right)\right) c_j y_j \\ &= \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} c_i y_i k\left(\vec{x}_i, \vec{x}_j\right) c_j y_j \end{aligned} \tag{18}$$

The constraints are also the same as equation (14).

$$\sum_{i=1}^{n} c_i y_i = 0 \text{ and } 0 \le c_i \le \frac{1}{2n\lambda}, \text{ for all } i \tag{19}$$

The parameters c_i can be solved by applying quadratic programming as before. Similarly, in the index i where $0 < c_i < \frac{1}{2n\lambda}$, the $\varphi(\vec{x}_i)$ just locates on the boundary of the margin in the transformed space. Then the bias b in the transformed space can be calculated as below.

$$\begin{aligned} b &= \vec{w} \cdot \varphi(\vec{x}_i) - y_i = \left[\sum_{k=1}^{n} c_k y_k \left(\varphi(\vec{x}_k) \cdot \varphi(\vec{x}_i)\right)\right] - y_i \\ &= \left[\sum_{k=1}^{n} c_k y_k k(\vec{x}_k, \vec{x}_i)\right] - y_i \end{aligned} \tag{20}$$

In recent research, the SVM algorithm introduces sub-gradient descent and coordinate descent. Both techniques have proven to provide significant advantages comparing with the conventional methods, when datasets are large and sparse. The sub-gradient method is especially efficient in the large training dataset, while the coordinate descent is effective when the dimension of the feature space is high.

The sub-gradient descent algorithm for the SVM works by following below expression.

$$f(\vec{w}, b) = \left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i + b)\right)\right] + \lambda\|\vec{w}\|^2 \tag{21}$$

Note that function f is a convex function of $\vec{w}$ and b. Therefore, the traditional Gradient Descent (GD) or Stochastic Gradient Descent (SGD) methods can be applied. A small difference from the GD is that the sub-gradient descent algorithm takes a step in the direction of a vector selected from the function's sub-gradient. The advantage of this algorithm is that, for certain implementations, the number of iterations are not affected by the number of data points [95].

The coordinate descent algorithm for the SVM works from the dual problem as equation (18) and (19). The parameter c_i is optimized in the direction $\partial f / \partial c_i$. Then, the result parameters $(c_1', \ldots, c_n')$ is aligned the closest parameters which meet the constraints (19). Generally the distances are represented by the Euclidean distances. The process is iterated until a near optimal vector of parameters is obtained. The advantage of this algorithm is extremely fast in practice, although there is few performance guarantees [96].

Above only discussed the SVM for the classification of two classes. For multiclass classification, there are also extension of the SVM, which named Multiclass SVM. The main idea for doing this is to divide the multiclass problem into multiple two-class classification problems.

There are two kinds of the divisions to deal with multiclass classification problem. One is creating two-class classifiers which distinguish between one label and the other all (one-versus-all). Another is creating two-class classifiers between every pair of classes (one-versus-one). In the previous case, for a new instance, the classification is done by a winner-takes-all strategy, where the classifier with the highest output score will assign the class. In the one-versus-one case, the classification is done by a max-wins voting strategy, where every classifier labels the instance to one of the two classes. Then the vote for the labeled class is increased by one vote. Finally the class with the most votes is treated as the classification result. Note that the one-versus-all strategy may suffer the unbalanced data problem, because the number of data in other classes are generally far more than the corresponding class. On the other hand, the one-versus-one strategy can solve this problem. However, the one-versus-one strategy may have extremely high computation cost when the number of class is large, because this strategy needs to build the classifier for every class pair. In the customer head and body

orientation classification tasks, the number of class is only 8. Thus this dissertation adopt the one-versus-one strategy for head and body orientation classification.

3.2.2. DCNN method

In recent year, the Deep Convolutional Neural Networks (DCNNs) are widely used in computer vision research and achieve higher performance than traditional HOG + SVM in different tasks [45] [46]. This chapter will introduce the basic concept of DCNN and the implementation of DCNN for orientation classification.

Compared with DCNN, most of the traditional machine learning algorithms are swallow (1-3 levels), such as SVM, Mixture of Gaussians (MoG), *k*-Nearest Neighbors (*k*-NN) and Principal Component Analysis (PCA). However, the mammal brain is organized in a deep architecture. Hubel *et al.* [97] discovered the receptive fields and their deep architecture in cat's vision system. Inspired by the architectural depth of the brain, researchers wanted for decades to train deep multi-layer neural networks. However, no successful attempts were reported before 2006, except the LeNet-5 proposed in 1998 [98]. On the other side, in 1992, Boser *et al*. [93] improved the SVM algorithm. After that, many researchers abandoned deep neural networks because SVMs worked better, and there was no successful attempts to train deep networks. The breakthrough of deep neural networks happened in 2006. Hinton *et al.* [99] and Bengio *et al.* [100] proposed the deep neural networks for image classification. Then, the deep learning drew more and more attentions.

The main advantage of DCNN is that it does not need the design a certain feature for it. The network is able to automatically learn the features, compared to the hand-craft feature like HOG. For example, Figure 3.3 shows a general framework of machine learning. The HOG + SVM discussed in the Chapter 3.2.1 also follows this framework. However, for DCNN, the feature extraction step is not needed. The feature is automatically learned in the DCNN.

In order to construct a DCNN, the general approach is to connect multiple basic layers. For example, Figure 3.4 shows the architecture of LeNet-5 [98]. There are 7 layers in the network. The subsampling layer in Figure 3.4 is often called pooling layer

nowadays. Using LeNet-5 as an example, here will introduce the basic layers in DCNN.

Convolutional layer

The convolutional layer is the most important layer in a DCNN. It contains a set of convolutional filters, whose parameters need to learn in the training. These filters are generally with a small size, but have the same channels number as the input. The effect output is constructed in the convolution layer. Therefore, every channel in the output can also be treated as an output of a neuron that focuses on small receptive field in the input and parameters of neurons are shared in the same channel.

There are two reasons for sharing parameters. One is that, when the input has large channel number (e.g. the RGB image has 3), it is too time consuming to calculate the connections to all of the neurons. The fully connected architecture cannot analyze the local spatial structure of the data. The second reason relies on an assumption that, if a

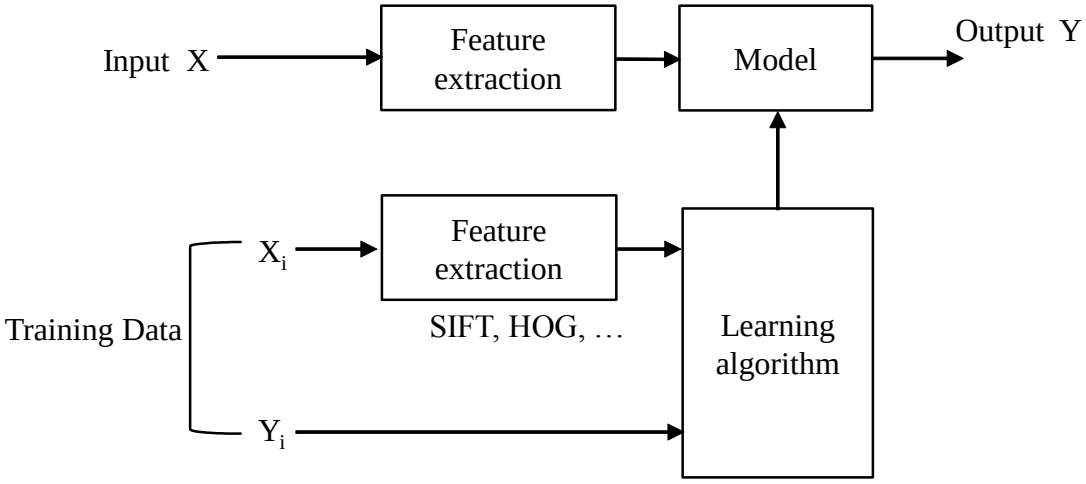


Figure 3.3 A general framework of machine learning. However, for DCNN, the feature extraction step is not needed

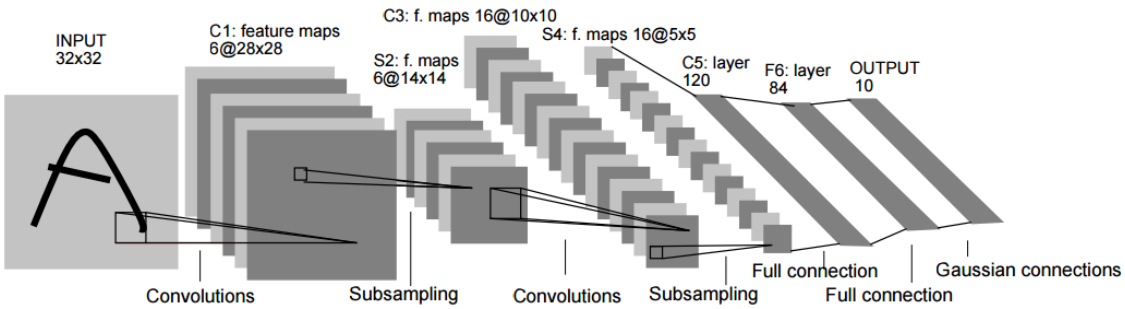


Figure 3.4 The architecture of LeNet-5 [98]. The image is directly sourced from [98]

local feature is important to calculate at certain positions, it should be also important at any other positions. Therefore, the convolutional layer enforces the local spatially correlation by limiting connection size and sharing the parameters. This small region can be also thought as the receptive field in this layer. This design make sure that the learnable convolutional filters can generate the strong activation in any position for specific input pattern.

There are 3 hyper-parameters which determine the size of the output data. They are the number of filters, stride of sliding window and the size of zero-padding.

The channel number of output data is determined by the filter number of this layer. These filters learn to give strong responds for some certain patterns in the input. For example, a raw image is given into a convolutional layer. Then different filters may generate in different types of oriented edges or circles.

The stride determines the sliding length of the filter. Because the filter size is generally much smaller than the input data, the filters needs to scan the input area in order to calculate the convolution across the input data. When the stride is 1 then the filters move 1 pixel per step.

The zero-padding size determines the how many zeros will be padded on the border of the input in convolutions. Padding provides a methods to adjust the size of output data. Specifically, the zero-padding can be applied to generate the exactly same size of input data in the output.

Therefore, the size of the output data can be determined by the above 3 hyper-parameters. Assume that the input data has size L, the filter size convolutional layer is F, the stride of filters is S, and the number of zero-padding is Z. Then the output size can be calculated as $(L - F + 2Z)/S + 1$. The output size may be not integer when the stride and the zero-padding number are not set correctly. Thus the filters cannot tightly fit the input data. Particularly, letting the number zero-padding $Z = (F - 1)/2$ ensures that the output data have the same spatially size with the input data when $S = 1$. Although it is generally not a requirement to keep the size of the previous layer, for example, one can use just a portion of padding.

Pooling layer

Another important layer of DCNN is pooling layer, which is named subsampling layer in Figure 3.4. The pooling layer conduct a non-linear down sampling to the input data. There are serval non-linear functions to conduct the pooling. In details, the input data is scanned by a window, whose stride must be equal to the window size. Then for each window position the maximum will be output in the max pooling. The essential thought of pooling layer is that the local feature in a specific location is less important than spatial relationship with other features. Usually, there are multiple pooling layer in the DCNN to gradually make size of the data smaller and smaller. It means the calculation time is also reduced. Moreover, this reduction can also control overfitting, because the pooling operation cam be considered as another kind of geometry invariance. Generally, pooling layers are periodically inserted between two successive convolutional layers in a CNN architecture.

The pooling layer operates independently on every channel of the input and down samples it spatially. The most common pooling size is 2×2. It means a 2×2 window scans with a stride of 2 along both width and height at every channel and down samples the input data. This pooling generates an output with 1/4 size of the input. The number of channels does not change through the pooling layer.

Because of the rapid decrease of the data size, the convolution layers after the pooling layer actually involve a lagers area in the raw image scale. This allows the succeeding convolution layers to learn more complex pattern from a large spatial scale, even their filters sizes do not increase. On the other hand, the pooling layer may result in the loss the detail information in the raw image. For example, if the target of network is to detect some small objects in the image, the pooling layers should avoid to being used to much in the network.

ReLU layer

The Rectified Linear Units (ReLU) works by following function $a(x) = \max(0, x)$. It is easy to see that the ReLU layer does not change the data size. The ReLU layer usually follows a convolutional layer although it is not used in Figure 3.4. It increases the nonlinear properties of the convolutional layer without affecting the receptive fields of the convolution layer. As a result, the nonlinearity is enforced for the

whole network.

Fully connected layer

The Fully Connected (FC) layers are generally set as the final layers. In FC layer, each filter has the same size as the input data, which means the neurons have connections to all elements of the input data. Therefore, the output of FC layer can be calculated as a multiplication of matrix between the input data and the filters.

Loss layer

The loss layer determines how to measure the difference between the predictions and ground truths. It is commonly the last layer in the training step and could be removed in the trained network. There are many loss functions can be used for different tasks. For example, softmax loss is used for predicting a single class from K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probabilistic values from 0 to 1. Euclidean loss is used for regressing to real value labels $(-\infty, +\infty)$.

Dropout layer

The dropout layer randomly sets a proportion of the input data to zero. The aim of dropout layer is to reduce overfitting. The DCNN is prone to overfitting when layers are too many or the number of parameters are too large. At each training batch, the dropout layers "drop out" the net with probability 1 - p, where the p is the dropout rate. The connections between the input data and output data are also deleted. Only the remained network is updated in one training mini-batch. Then the weights of deleted nodes are reset to their values. In every training mini-batch, the dropout nodes are randomly selected. In the prediction stage, the dropout layer does not dropout anything or can be directly removed.

In the training stages, the drop rate is generally set as 0.5. However, for dropout layers which near to the first input layer, the drop rate may adjust to a lower value, in order to avoid to losing too much information at the beginning.

The dropout layers avoid overfitting in the training. They also significantly reduce the training time. Although the dropout layers seem to reduce node connections, they

actually force the network to obtain more robust parameters that produce better results in new data.

Batch Normalization layer

The batch normalization layer shifts the input data to a certain mean and variance. It is proposed by Ioffe and Szegedy [101]. The essential though behind the batch normalization is to reduce the internal covariate shift.

The normalization operation is often used as the pre-processing in the traditional machine learning, such as SIFT and HOG. As in the DCNN, the values are automatically adjust in the network. However, sometimes it makes the data too big or too small again. This condition is called an internal covariate shift in [101]. By normalizing the data in each training mini-batch, this problem can be significantly avoided.

Basically, rather than just performing normalization once in the beginning, the batch normalization layer adjust the data to a better distribution in every training mini-batch. Assume $x_i \in \{x_1, \dots, x_m\}$ is one the data in a training mini-batch. Then the mean μ and variance σ^2 of this mini-batch can be calculated as below.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{22}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \tag{23}$$

Then the layer normalizes the data to zero-mean and unit variance as below.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{24}$$

where ϵ is a small value to avoiding dividing a zero. Finally, the layer applies a scale and shift

$$y_i = \gamma \hat{x}_i + \beta \tag{25}$$

where y_i is the output data, γ and β are the learnable parameters. Commonly, in the first training mini-batch, γ and β are initialized as μ and σ of this this mini-batch. It means the layer exactly normalizes the first input data to zero-mean and unit variance. With the training process going on, the values of γ and β are updated. Therefore, they can learn that any other distribution might be better.

By introducing the batch normalization layer, a higher learning rate can be applied to accelerate the training process. Generally, learning rate for DCNN is set as a small value, so that only a small portion of gradients corrects the weights. The reason is that one does not want the outlier data affect the whole network. By batch normalization, these outlier data are reduced. Therefore a higher learning rate can be adopted to improve the learning speeding.

Implementation of ResNet for head and body orientation

In recent year, the DCNN become deeper and deeper. The layer number increase from 8 layers of AlexNet [47], to 19 layers in VGG [48] and 22 layers in GooLeNet [49], until 152 layers in ResNet [23]. The performance is also improved with the increasing depth of network. These network are originally used for classification task. That means one can slightly modify some top Full Connection (FC) layers to adapt different tasks.

This dissertation implements the ResNet for head and body orientation classification, which is champions of classification task in ImageNet Large Scale Visual Recognition Competition in 2015.

ResNet is a very deep neural network, which is constructed by many small residual structures. The structure of ResNet is shown as Figure 3.5. Usually the deep neural network suffers the vanishing gradient problem. That means when updating the parameters of network using error backpropagation algorithm, the gradient of error become smaller and smaller. Finally, the gradient would decrease to nearly zeros in the deep neural network. The vanishing gradient problem makes the deep neural network become difficult to train. Therefore simply increasing the depth of network will not produce better results. In order to solve this problem, the ResNet introduces a residual structure as Figure 3.5.

See the residual structure in Figure 3.5. $H(x)$ is arbitrary desired mapping of the residual structure. $F(x)$ is the residual mapping with respect to the identity *x*. The two weighted layers are hoped to fit $F(x)$. By introducing the bypass identity *x*, the gradient decreasing could become slower. This is the reason that ResNet can improve the performance.

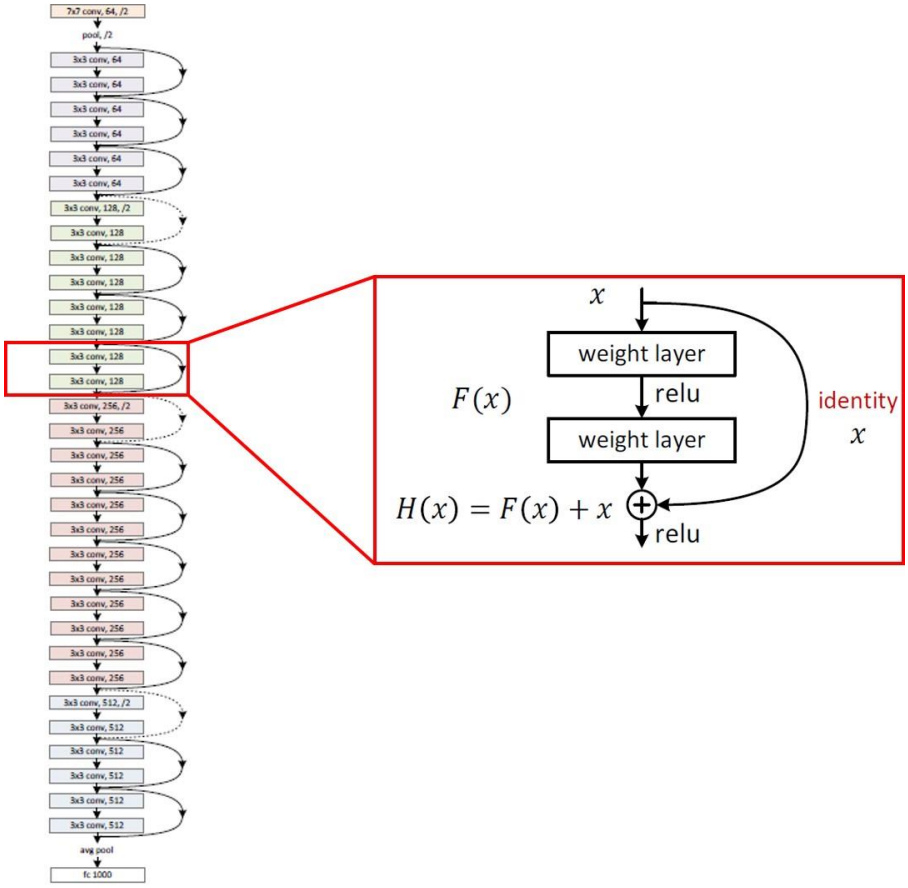


Figure 3.5 The structure of ResNet and its basic inception module. Except the red bounding box, the images are directly sourced from [23]

The authors of [23] propose 3 variations of ResNets with different depths: ResNet-50, ResNet-101 and ResNet-152, which mean they contain 50 layers, 101 layers and 152 layers respectively. These networks are originally used to classify 1000 classes object. This dissertation modifies the ResNet-50 for 8-class orientation classification.

In order to modify the ResNet to adapt the orientation classification, the last FC layers needs to output an 8-length vector. This modification can be achieved by change the filter size of FC layer from $1 \times 1 \times 2048 \times 1000$ to $1 \times 1 \times 2048 \times 8$. The detail of modification is listed in Table 3.1. The structure of modified ResNet is shown as Figure 3.6.

Similar to the original classification task of ResNet, the orientation classification also applies the cross entropy as the loss function. The orientation loss L_o and its gradient e_o are calculated as below, where *o* and *o'* are the predicted orientation vector and true orientation vector respectively.

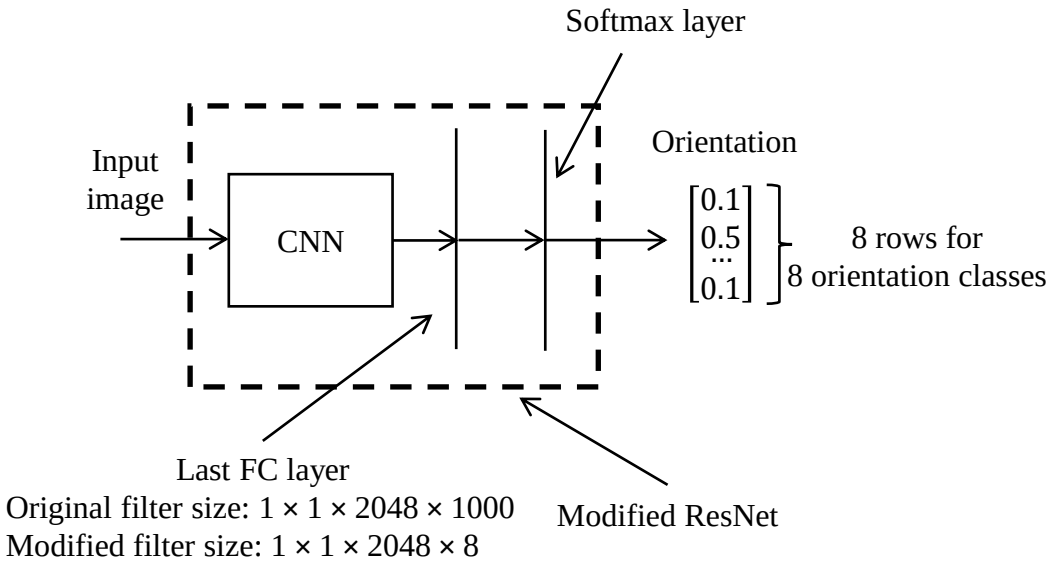


Figure 3.6 The structure of modified ResNet for orientation classification

Table 3.1 The modification of ResNet for orientation classification

	Input size of FC layer	Filter size of FC layer	Output size of FC layer	Output size of softmax layer
Original GoogLeNet	1 × 1 × 2048	2048 × 1000	1 × 1 × 1000	1 × 1 × 1000
Modification for orientation classification	1 × 1 × 2048	2048 × 8	1 × 1 × 8	1 × 1 × 8

$$L_o = -\sum_{i=1}^{8} o_i \log o_i' \quad (26)$$

$$e_o = o - o' \quad (27)$$

3.3. Head and Body Orientation Classification using Semi-Supervised Learning

3.3.1. HOG + SVM based SSL

In this dissertation, the head and body orientation of customer is detected in 8 directions as shown in Figure 2.3. Generally, the orientation can be predicted by using pre-trained orientation classifier by applying the standard Supervised Learning (SL)

algorithm [15] [16] [17] [18] [19] [22].

However, the ground truth is difficult to be obtained in practice. The training data labels are usually decided subjectively. Because the orientation in real world varies continuously, there are always some cases which have an orientation between two defined labels. These cases may result in errors in the training data. One of the solutions is to not forcibly give a label to these unclear cases, but give them a weak label. The weak label means a label between two adjacent defined classes. Meanwhile, the cases which are near the center of defined class (e.g. 45°, 90° and 135°) will be given a strong label. Figure 3.7 shows examples of strong and weak labels of customer body orientation. The labels {1, 2, …, 8} in Figure 3.7 mean the orientation classes {0°, 45°, …, 325°}.

By introducing the weak label, Semi-Supervised Learning (SSL) is used for training the orientation classifier instead of SL. The main idea of SSL is to use the labeled data to generate an initial classifier, then gradually classify the unlabeled data and add into

1 2 3 4 5 6 7 8

(a)

(1, 2) (2, 3) (3, 4) (4, 5) (5, 6) (6, 7) (7, 8) (8, 0)

(b)

Figure 3.7 (a) Strong label. The number under the image is the defined label in classifier. (b)Weak label. The pair of numbers under the image means it possibly belongs to one of two adjacent defined classes

the labeled data. Instead of unlabeled data, this dissertation uses weakly labeled data. Then the HOG features [20] are extracted for training. The Support Vector Machine (SVM) is used in training. The flowchart and example of SSL are illustrated as Figure 3.8 and Figure 3.9 respectively.

As shown in Figure 3.8, the SSL firstly trains the strongly labeled data to generate a classifier in *training* step. Then this classifier is used to estimate the labels for all of weakly labeled data in *testing* step. After this step, the SSL ranks the weakly labeled data in descending order by the likelihood. The *N* data with highest likelihood will be added into the strongly labeled data. Only the possible weak labels are considered in the ranking. For example, in Figure 3.9, if the SSL tries to add *N* weakly labeled data to the strongly labeled class 1, only the weak labels which contain label 1 are considered in the ranking for class 1. The SSL repeats these steps until no new strongly labeled data appear. In other word, the strongly labeled data generate the initial classifier, and the weakly labeled data refine the classifier in iteration.

There are two advantages of using weak label. Firstly, it solves the problem of inaccurate orientation labels. The strongly labeled data give strong positive label information to each class. The unclear weakly labeled data, which may bring negative label information if one forcibly labels them, do not attend the training at first. Secondly, this solution does not need to increase training data. The weakly labeled data are automatically added to the most closed class in the SSL process. This process guarantees

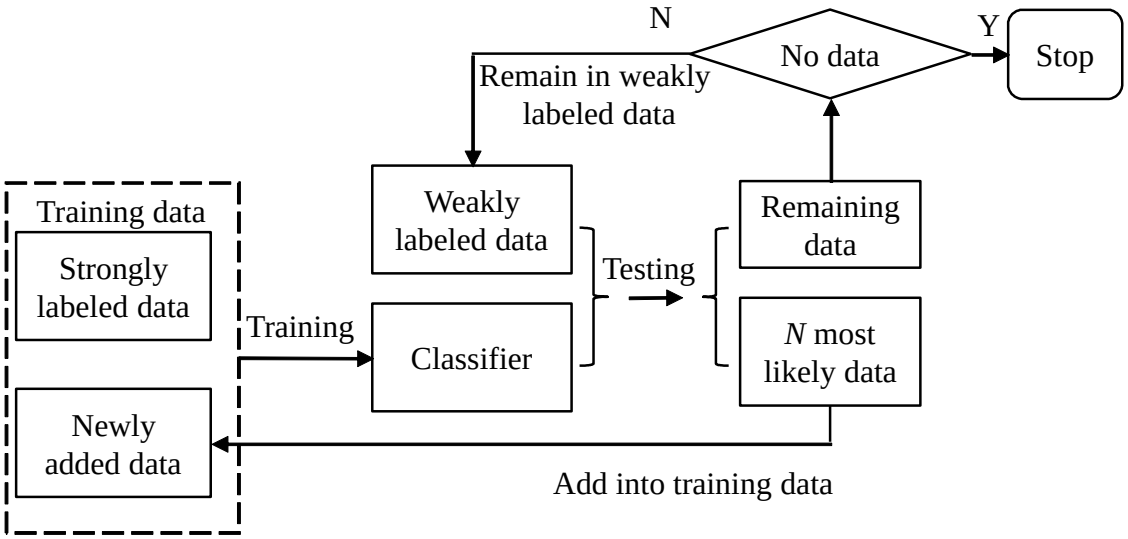
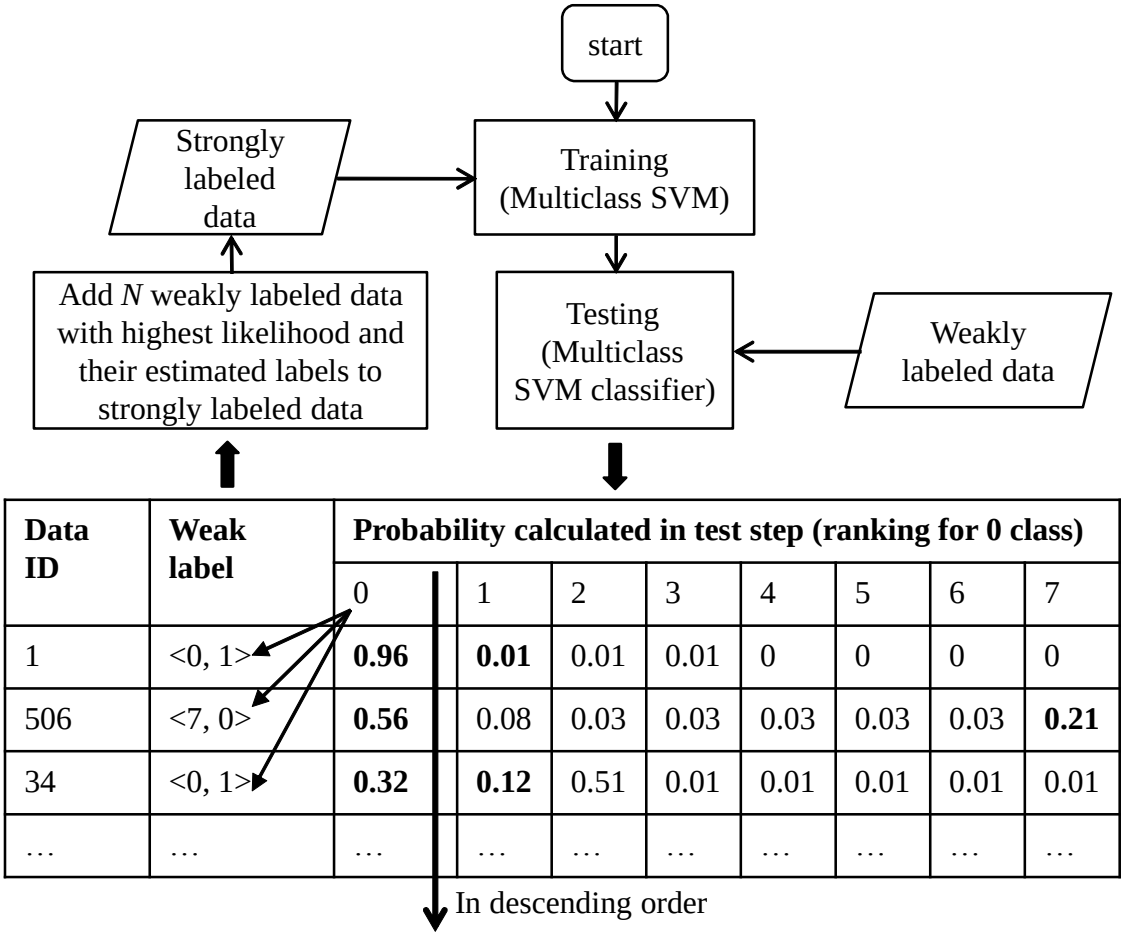


Figure 3.8 The flowchart of Semi-Supervised Learning



Data ID	Weak label	Probability calculated in test step (ranking for 0 class)							
		0	1	2	3	4	5	6	7
1	<0, 1>	**0.96**	**0.01**	0.01	0.01	0	0	0	0
506	<7, 0>	**0.56**	0.08	0.03	0.03	0.03	0.03	0.03	**0.21**
34	<0, 1>	**0.32**	**0.12**	0.51	0.01	0.01	0.01	0.01	0.01
…	…	…	…	…	…	…	…	…	…

Figure 3.9 An example of Semi-Supervised Learning using SVM

that each unclear case can produces largest positive label information. Therefore, there is no need to add extra training data to offset the negative effects of the unclear cases.

3.3.2. DCNN based SSL

The proposed SSL can adapt to any form of classifier, including the DCNN classifier. This chapter will introduce the details of the implementation of ResNet for SSL.

As discussed in Chapter 3.2.2, the output of softmax layer in ResNet is an 8-length probability vector. It could be directly fed into the SSL process which is discussed in the Chapter 3.3.1.

However, there are still some differences between the DCNN and SVM implementation. Firstly, unlike the SVM, one DCNN classifier is needed to train for

multiple epoch. Therefore, the setting of learning rate should follow these principle. For the initial network which is modified from the original one, the learning rate should be set a relative large value. Then the value decrease gradually in the training process. After obtain the first classifier, the SSL will add a batch of new training data from the weakly labeled data. In the following training, the learning rate should keep the small value, because the network has already been able to well classify the orientation to some degree. Secondly, there are several batch normalization layers in the ResNet. These layers normalize the inputs by the whole training dataset distribution. For the initial network which is modified from the original one, the parameters of these batch normalization layers should be reset, because these parameters are accumulated from the training data of original ResNet. Moreover, when every time the SSL add new training data from the weakly labeled data, the parameters of batch normalization layers need to be reset, because the distribution of training data has been changed.

From the above discussion, the training of the ResNet based SSL is quite time consuming comparing with the SVM based SSL. However, in the experiments, the ResNet based SSL show better performance than SVM based SL, ResNet based SL and SVM based SSL.

3.4. Head and Body Orientation Detection and Tracking in Sequence

After generating the classifiers of head and body orientation, they can be applied to detect and track the head and body orientation in image sequence. Unlike the body area which can simply detected by background subtraction, the head position needs to be detected and tracked. On the other hand, the orientation difference between human head and body could not be more than 90°, and the orientation also should not change dramatically in continuous frames. In order to obtain reasonable and stable detection result from frame sequences, the temporal constraint and the human physical model constraint need to be jointly considered. Figure 3.10 shows the flowchart of joint head and body orientation detection.

Firstly, a sliding window method is applied at the start time t_0 to get the position and orientation of head in the first frame. The probability of the head orientation of each

scanned block image is $P(d_H|C, x_H, y_H, s_H)$. Here, (x_H, y_H) is head position, s_H means head size, d_H denotes head orientation, C is the customer image. Moreover, the probability of body orientation can be described as $P(d_B|C)$, where d_B denotes body orientation. The estimation result is shown by rectangle and arrows on left-second image in Figure 3.10. The arrow connecting to a rectangle means head orientation and head position. The arrow from the center of body means body orientation. After the first frame, a particle filter is applied to track the head position, the head size, the head orientation and the body orientation. The particles are distributed around the estimation result of last frame. The state of one particle q is a vector including five elements: head position (x_H, y_H), head size s_H, head orientation d_H and body orientation d_B, which is described as (28).

$$q = [x_H, y_H, s_H, d_H, d_B]^T \tag{28}$$

The human physical model constraint is described as (29).

$$L_{model} = P(d_H|C, x_H, y_H, s_H) \cdot P(d_B|C) \cdot P_M(d_B - d_H, 0, k_{HB}) \tag{29}$$

where P_M denotes von Mises distribution, also known as the circular normal distribution. The k_{HB} is a constant value that decides the shape of von Mises distribution. Basically, the value of L_{model} is inversely proportional to $(d_B - d_H)$. The value of L_{model} is proportional to the probability of head orientation $P(d_H|C, x_H, y_H, s_H)$ and the probability of body orientation $P(d_B|C)$.

The temporal constraint $L_{temporal}$ is a multiplication of two parts: body temporal constraint $L_{temporal}^{body}$ and head temporal constraint $L_{temporal}^{head}$, which are represented as (30), (31) and (32).

$$L_{temporal}\big(q(t), C(t), q(t-1)\big) = L_{temporal}^{body}\big(q(t), C(t), q(t-1)\big) \cdot L_{temporal}^{head}(q(t), C(t), q(t-1)) \tag{30}$$

$$L_{temporal}^{body}\big(q(t), C(t), q(t-1)\big) = P\big(d_B(t)\big|C(t)\big) \cdot P_M(d_B(t) - d_B(t-1), 0, k_B) \tag{31}$$

$$L_{temporal}^{head}\big(q(t), C(t), q(t-1)\big) = P\big(d_H(t)\big|C(t), x_H(t), y_H(t), s_H(t)\big) \cdot P_M(d_H(t) - d_H(t-1), 0, k_H) \tag{32}$$

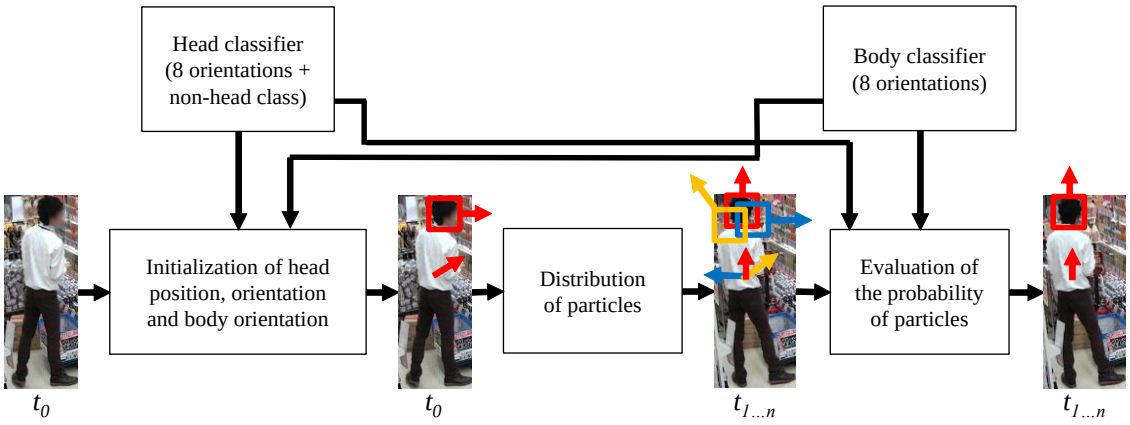


Figure 3.10 The flowchart of joint head and body orientation detection. The rectangle with arrow around head means head localization and orientation. The arrow which starts from the center of body indicates body orientation

The format of $L^{body}_{temporal}$ and $L^{head}_{temporal}$ are similar to L_{model}, the value of $L^{body}_{temporal}$ and $L^{head}_{temporal}$ is inversely proportional to the orientation difference in two frames. In the particle filter, the likelihood of one particle is represented as a multiplication of model constraint L_{model} and temporal constraint $L_{temporal}$. The particles are weighted by their likelihoods and generate joint orientation detection result in each frame.

3.5. Experiments

3.5.1. Dataset

The experiments are designed as two parts. One is for customer head and body orientation classification. Another is for orientational behavior classification in video. Therefore, there are two datasets corresponding to these two parts of experiments. They are named customer orientation dataset and customer orientational behavior dataset.

The data of customer orientation dataset and customer orientational behavior dataset are shot from real surveillance camera of retail stores. The surveillance videos are selected from the security camera near the merchandise shelf. The occlusion between customers is not included in the dataset.

The selected videos are from 4 surveillance cameras, which are installed in 4 different retail stores respectively. These cameras shoot customers from different view

angles, which are shown in Figure 3.11.

The details of customer orientational dataset is shown in Table 3.2. In the dataset, there are 9,703 images for training, and 3,250 images for testing. Each image includes the full body of a customer. Additionally, the head area of customer is cropped for the head orientation classification. Thus the numbers of head images and body images are the same. The head or body orientations which are near the center of defined class (e.g. 45°, 90° and 135°) will be given one strong label, otherwise the data will be given a pair of adjacent weak labels. The resolution of original images is 720 by 480. The body height of customer is around 300-400 pixel, the head size is around 40-50 pixel. The frame rate of videos is 29 frames per second. As the explanations in Chapter 2, all of the

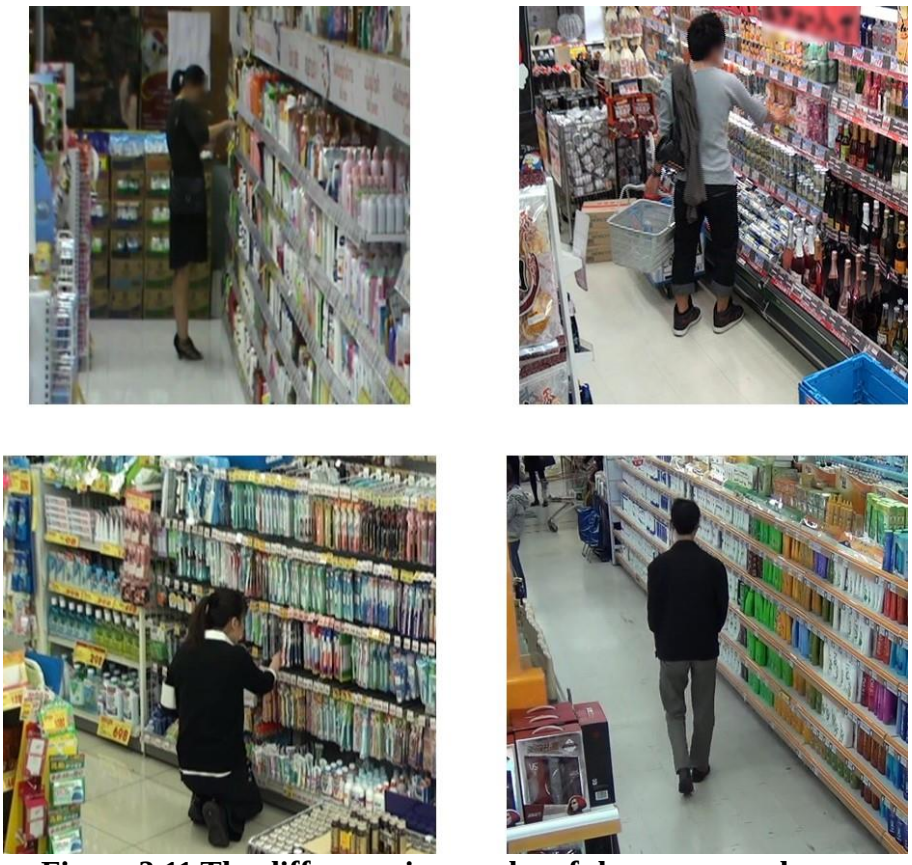


Figure 3.11 The different view angles of the customer dataset

Table 3.2 The details of customer orientation dataset

	Training data	Testing data	Total
Image number	9,703	3,250	12,953
Sequence number	58	21	79

images used in experiments are after background substation. The figures shown with background are just for clarity.

For retailers, they may not care the customer head and body orientation in one specific frame. They do care the customer orientational behavior in a certain sequence. Thus, 3 groups of sequences are picked up as the customer orientational behavior dataset. The sequences are grouped by the orientational behavior *no interest*, *looking at shelf* and *turning to shelf*, which are defined in Chapter 2. In this dataset, the customer keep the same head and body orientation at least 1 second (29 continuous frames). For example, in one *looking at shelf* sequence, the customer' head orientation will direct to the shelf for at least 1 second, and his or her body orientation will not direct to the shelf for at least 1 second. Unlike the customer orientation dataset, the customer orientational behavior dataset excludes the all of the customer arm actions. The details of the customer orientational behavior dataset is shown as Table 3.3.

3.5.2. Experiments of head and body orientation detection

Firstly, the experiments about the head and body orientation classification are conducted on the customer orientation dataset. The training data are augmented by left-right flipping in the experiments. The results of orientation detection are evaluated by the orientation accuracy, which is defined as the ratio of the number of correct detected orientations to the total number of orientations. Considering the ambiguity of orientation, the neighboring orientation accuracy is also applied, which means the accuracy allowing one neighbor label error.

The following experiments are conducted for head orientation classification:

(1) SL based on HOG + SVM [19].

(2) SL based on HOG + Sparse Representation based Classification (SRC) [102].

(3) SL based on Local Binary Patterns (LBP) + SVM [103]

(4) SL based on ResNet-50.

(5) SSL based on HOG + SVM.

(6) SSL based on HOG + SRC.

(7) SSL based on LBP + SVM.

Table 3.3 The details of customer orientational behavior dataset

Orientational behavior	Sequence number
No interest	20
Looking at shelf	20
Turning to shelf	29
Total	69

(8) SSL based on ResNet-50.

For the training of SL in experiments (1) ~ (4), each weakly labeled image is randomly given one of the pair of weak labels as its unique label. In each iteration in SSL, up to 50 weakly labeled data with highest likelihood would be added into each strongly labeled class. In the SL with ResNet, the network is trained for 1,000 epochs. The learning rate starts from 10^{-3} and then decreases by half every 100 epochs. In the SSL with ResNet, the network is firstly trained for 500 epochs on strongly labeled data. The learning rate also starts from 10^{-3} and then decreases by half every 100 epochs. In each iteration in SSL, the network is trained for 10 epochs with learning rate 10^{-5}. In these experiments, the head images are resized to 18 × 18 for HOG and LBP feature extraction. For ResNet input, the head images are resized to 224 × 224.

For the body orientation, the same experiments as (1) ~ (8) are conducted for the body data. The body images are resized to 64 × 128 for HOG and LBP feature extraction. For ResNet input, the body images are also resized to 224 × 224.

The results are shown in Table 3.4. In Table 3.4, the general trends are that SSL methods outperform the SL methods and deep learning methods outperform the hand-crafted features based methods. The combination of SSL and ResNet achieves the best performances.

Then these classifiers are used in the joint head and body orientation detection method. The results are shown in Table 3.5. The values in brackets indicate the improvements compared with that in Table 3.4. In Table 3.5, the joint head and body orientation detection shows the significant improvements of head and body orientation detection in the image sequence.

After generating the classifiers of head and body orientation, they can be also applied for orientational behavior classification by joint head and body orientation

detection. Here, the classifiers of SSL based on ResNet are chosen for the experiments on the customer orientational behavior dataset. In the sequence, if the predicted head and body orientations are correct (allowing one neighboring error) for more than 1 second (29 continuous frames), the orientational behavior detection is treated as correct. The behavior accuracy will be calculated as the ratio of the number of correct behavior detections to the total behavior number.

The experimental results are shown in Table 3.6. In Table 3.6, the behavior accuracy

Table 3.4 The orientation accuracy of SL and SSL

(a) The result of head orientation classification

Classifier	Orientation accuracy	Neighboring orientation accuracy
SL based on HOG + SVM [19]	43.7%	80.0%
SL based on HOG + SRC [102]	45.5%	84.1%
SL based on LBP + SVM [103]	45.6%	84.8%
SL based on ResNet	52.8%	85.7%
SSL based on HOG + SVM	45.9%	82.6%
SSL based on HOG + SRC	47.2%	86.0%
SSL based on LBP + SVM	47.4%	86.1%
SSL based on ResNet	**55.9%**	**89.5%**

(b) The result of body orientation classification

Classifier	Orientation accuracy	Neighboring orientation accuracy
SL based on HOG + SVM	48.1%	82.4%
SL based on HOG + SRC	50.2%	85.0%
SL based on LBP + SVM	47.9%	81.2%
SL based on ResNet	55.4%	85.4%
SSL based on HOG + SVM	51.3%	85.1%
SSL based on HOG + SRC	53.9%	88.5%
SSL based on LBP + SVM	51.0%	84.5%
SSL based on ResNet	**60.8%**	**90.3%**

Table 3.5 The orientation accuracy in joint head and body detection

(a) The result of head orientation detection

Classifier	Orientation accuracy	Neighboring orientation accuracy
SL based on HOG + SVM [19]	57.8% (+14.1%)	87.1% (+7.1%)
SL based on HOG + SRC [102]	58.6% (+13.1%)	89.4% (+5.3%)
SL based on LBP + SVM **[103]**	58.5% (+12.9%)	89.9% (+5.1%)
SL based on ResNet	64.7% (+11.9%)	90.7% (+5.0%)
SSL based on HOG + SVM	58.9% (+13.0%)	88.8% (+6.2%)
SSL based on HOG + SRC	59.2% (+12.2%)	90.8% (+4.8%)
SSL based on LBP + SVM	59.7% (+12.3%)	90.6% (+4.5%)
SSL based on ResNet	**67.0% (+11.1%)**	**92.6% (+3.1%)**

(b) The result of body orientation detection

Classifier	Orientation accuracy	Neighboring orientation accuracy
SL based on HOG + SVM	60.3% (+12.1%)	87.9% (+5.5%)
SL based on HOG + SRC	61.6% (+11.4%)	89.9% (+4.9%)
SL based on LBP + SVM	59.8% (+11.9%)	87.8% (+6.6%)
SL based on ResNet	65.2% (+10.2%)	90.7% (+5.3%)
SSL based on HOG + SVM	62.3% (+11.0%)	90.4% (+5.3%)
SSL based on HOG + SRC	63.8% (+9.9%)	91.6% (+3.1%)
SSL based on LBP + SVM	61.1% (+10.1%)	90.5% (+6.0%)
SSL based on ResNet	**70.5% (+9.7%)**	**93.2% (+2.9%)**

is higher than the orientation accuracy in Table 3.4. The reasons are: firstly, the joint head and body orientation detection and tracking correct some detection errors; secondly, some isolated errors do not affect the whole behavior detection results. Figure 3.12 shows the example results of head and body orientation detection in image sequences. The example results show that the customer head and body orientation can be successfully detected in wide angle range.

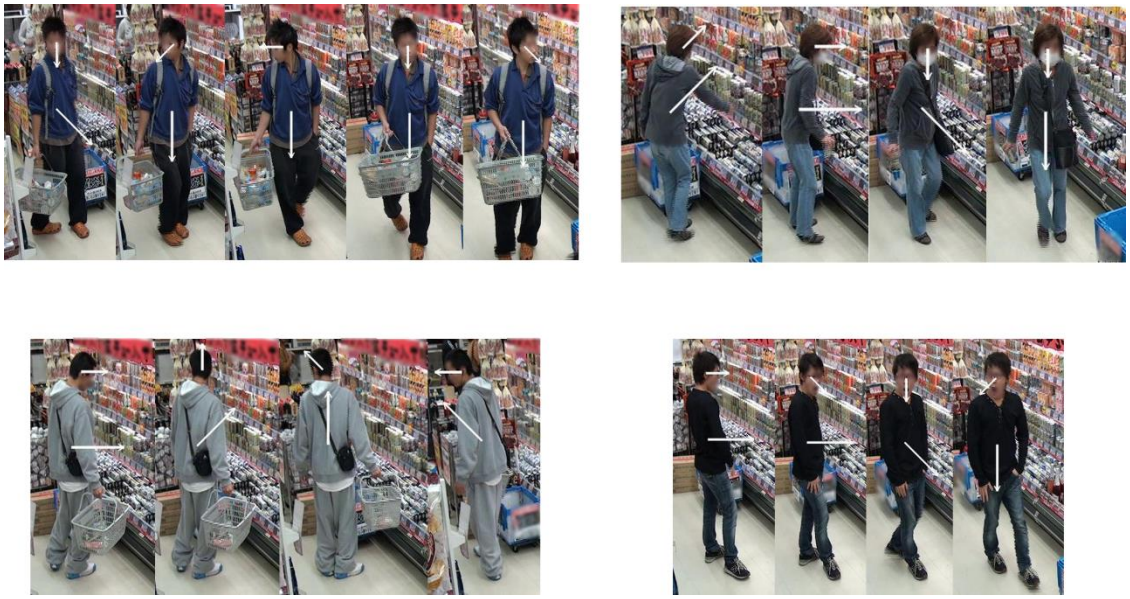


Figure 3.12 Example results of customer head and body orientation detection in image sequences

Table 3.6 The confusion matrix of customer orientational behavior classification

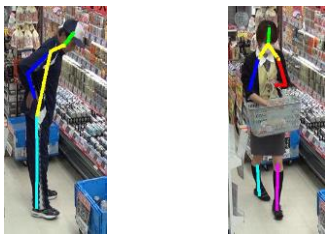
	No interest (estimated)	Looking at shelf (estimated)	Turning to shelf (estimated)
No interest (actual)	**95.0%**	**0.0%**	**5.0%**
Looking at shelf (actual)	**5.0%**	**90.0%**	**5.0%**
Turning to shelf (actual)	**6.9%**	**0.0%**	**93.1%**

Chapter 4. Customer Pose Estimation

4.1. Overview

The customer pose, which includes the positions of joints, can provide rich information about the customer interest level to the merchandise. For example, the hand position could tell whether the customer is taking an item. The foot position can reveal the distance between the customer and merchandise shelf. Some special pose configurations can also represent the squatting and bending over behaviors. These are the reason of that this dissertation treats the customer pose estimation is an indispensable part of customer interest level assessment system.

However, the 2D human pose estimation is not an easy task in computer vision. One of the difficulties is the occlusion problem, including both self-occlusion and inter-occlusion by other objects. The main reason of self-occlusion is the body orientation. For example, if a standing person is facing the right in the camera, his or her left body is probably occluded. On the other hand, the inter-occlusions may happen due to arbitrary objects. In retail store environment, the inter-occlusions are typically caused by the shopping baskets. Figure 4.1 shows the examples of the self-occlusion and the inter-occlusion by shopping basket. Another difficulty of pose estimation is the left-right similarity problem because of the symmetry of human body. For example, the left shoulder of a person in the back view is very similar as the right shoulder in front view. In order to address these problems, this chapter introduces the body orientation and visibility mask for each joint. Therefore, in order to solve these problem, this dissertation proposes to incorporate the pose estimation with body orientation, joint connections and



(a) (b)

Figure 4.1 The examples of occlusions in the retail store. (a) The left body part is self-occluded. (b) The right wrist, left hip and right hip are occluded by the shopping basket

joint visibility mask. These elements are tightly related no matter in physics or in the image.

The body orientation is a kind of global information of pose configuration. The body orientation is defined in 8 directions as Figure 2.3. The body orientation is useful for solving both self-occlusion problem and left-right similarity problem. For example, if one knows a person is facing to right, the body orientation indicates the occlusion of his or her left body. Similarly, if a person is facing to the camera, his or her right shoulder is probably at the left side of image. To this end, this dissertation proposes novel Orientational Message Passing (OMP) layers on top of a Fully Convolutional Network (FCN) to fuse the global body orientation and local joint connection. These layers can be seen as an extension of Deformable Parts Model on FCN structure [57] [58], which introduced the spatial constraints into the deep neural network. In other word, the OMP simultaneously incorporate the body orientation and the joint connections. Note that the orientation defined in Figure 2.3 is only applicable for some simple pose in the store, such as standing, walking, bending over or squatting down. For more complex poses, it is difficult to give a body orientation in the definition of 8 directions, such as the dancing, parkover or doing gymnastics.

The visibility mask is a binary vector where its element indicates the visibility of each joint. The term of visibility mask is introduced by Haque et al [52]. However, they only applied the visibility mask in the top view images for self-occlusions. This dissertation extends the application of visibility mask for more flexible view angles and for both self-occlusion and the inter-occlusion by objects. Other approaches such as [104] [64], also used a binary vector to represent the occlusion. Although these approaches modeled the occlusions in different ways, they did not consider the relationship between the joint occlusion and body orientation. This dissertation argues that the body orientation is an indispensable part of the occlusion model. Besides solving left-right similarity problem, the proposed OMP layers can also pass different occlusion information according to different body orientations. Additionally, this dissertation combines the occluding object detection in the system. It is an apparently clue of inter-occlusion for visibility mask prediction. From observation, the occluding objects are

mainly the shopping baskets in retail stores. (For the shopping cart, the mainly occluding part is also the basket on the cart.)

Furthermore, the temporal consistency in customer pose is also considered. The system includes the optical flow information to improve the estimation accuracy of some variable joints like wrists. A Bidirectional Recurrent Neural Network (BRNN) [65] is also employed on top of the network to improve the accuracy by utilizing both the forward and backward sequences.

Therefore, the pose, body orientation, joint connections and joint visibility mask are fused in one network. This dissertation names this network as Integral Pose Network (IntePoseNet). However, this network seems to be too heavy. All of the elements are piled up on one network. In order to make the system simple and efficient, this dissertation finally proposes a multi-task neural network to produce the 4 elements simultaneously.

The hypothesis behind the multi-task neural network does not change: the pose, body orientation, joint connections and joint visibility mask are tightly related. Therefore, these elements are expected to help each other to learning better in the one network. In the proposed multi-task neural network, the 4 tasks share an FCN backbone. On top of that, there are 4 task specific output layers to produce different outputs.

In the rest chapter, the Chapter 4.2 introduces the traditional DPM for pose estimation and gives an implementation to incorporate the DPM with body orientation. The Chapter 4.3 introduce two kind of deep learning methods for pose estimation. One is to directly predict the joint positions, and another is to produce the joint heatmaps and then calculate the joint positions. The proposed OMP layers are introduced in the joint heatmaps learning part. Then the Chapter 4.4 describes the proposed the Integral Pose Network, which incorporate the optical flow, basket heatmap and BRNN. The Chapter 4.5 discusses the proposed Multi-task Neural Network for pose estimation. Finally, the Chapter 4.6 conducts a series experiments to evaluate different methods.

4.2. Pose Estimation using Deformable Parts Model

4.2.1. Basic Knowledge of Deformable Parts Model

Deformable Parts Model (DPM) is initially proposed by Felzenszwalb *et al*. [105] The main idea is to treat a whole target as a series of connected parts. The detection of parts considers not only the local parts but also the connection between neighbor parts. For example, a person could be seen as the connections of head, torso, arms and legs, a horse could be seen as the connections of head, neck, body and legs. In recent years, DPM is already widely used in many classification, segmentation, pose estimation and action recognition tasks

The DPM method in [105] could be divided into roughly 3 steps. The first step is the detection of the objects using a lower-resolution global template. The aim is to roughly localize the target objects in the image. Secondly, a set of spatially flexible high-resolution 'part' templates are applied to detect the parts of objects. These templates capture the local appearance properties of each parts. The global and part templates are shown in Figure 4.2. Thirdly, the deformations are characterized by links connecting them to fine tune the positions of each parts. In this dissertation, the customer image is already obtained by the background subtraction. Thus just the second and third steps would be explained here.

For pose estimation, the pose can be represented as a set of vertexes and edges (*V*, *E*) as Figure 2.5. It is a tree structure and the head is the root of tree. Then the full score of a pose configuration given an input image *I* is as follow.

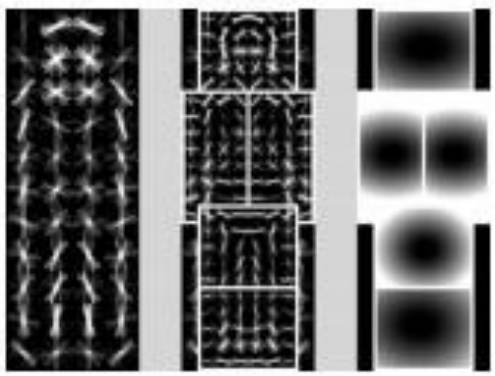


Figure 4.2 The global and part HOG templates in [105]. The image is directly sourced from [105]

$$score(l|I) = \sum_{i \in V} \varphi(l_i|I) + \sum_{(i,j) \in E} \psi(l_i, l_j|I) \quad (33)$$

where the $l = (x, y)$ is the pixel location. The first unary term $\varphi(l_i|I)$ provides the local confidence of the appearance of part *i*. It could be seen as the feature map of each human joint. It can be generated by a HOG detector [105] [41] [106], a sliding CNN [57], or a FCN [58], The second pair-wise term $\psi(l_i, l_j|I)$ model the compatibility two neighboring parts *i* and *j*. In [105] [41] [106] [57] , it is modeled by a parameterized Euclid square distance as (34), where $\omega_{i,j}$ is the weight parameters and the operator $\cdot$ means element-wise multiplication. In [58], it is written as a convolution with a filter as (35), where $w_{j,i}$ and $b_{j,i}$ are the filter and bias, operator $\otimes$ is convolution and *f*() is an activation function. The format of equation (35) is designed for using general CNN operations so the calculation can be implemented in CNN structures.

$$\psi(l_i, l_j|I) = \omega_{i,j} \cdot \left[x_i - x_j, (x_i - x_j)^2, y_i - y_j, (y_i - y_j)^2\right]^T \quad (34)$$

$$\psi(l_i, l_j|I) = f\left(\varphi(l_j|I) \otimes w_{j,i} + b_{j,i}\right) \quad (35)$$

The aim of DPM is to find the optimal pose configuration which maximizes the score in (33). Mathematically, equation (33) represents a discrete, pairwise Markov Random Field (MRF). The pair-wise term is a message which is sent from neighbor part. When (*V*, *E*) is tree-structured, one can compute maximum of score with dynamic programming.

The DPM in [105] improve the accuracy by modeling the relationships between parts. However, there is a limitation. The problem is the types of parts may be very various. For example, the human hand may be shown in multiple appearances and its relative position to elbow also has large degree of freedom. A single appearance model and a single relationship model are difficult to represent this variety. To solve this problem, Yang *et al*. [41] introducing the concept of Mixture-of-Parts.

In [41], each part is clustered into multiple "types" according to its relative position to its parent part. The clustering is applied K-means algorithm and the example results are shown in Figure 4.3. Therefore, the equation (33) is rewritten as follows, where $\phi(l_i, t_i|I)$ is the extracted HOG feature and $\omega_i^{t_i}$ is its weight, *t* denotes the type

of parts.

$$\text{score}(l,t|I) = \sum_{i\in V} \varphi(l_i,t_i|I) + \sum_{(i,j)\in E} \psi\left(l_i,l_j,t_i,t_j|I\right) \tag{36}$$

$$\varphi(l_i,t_i|I) = \omega_i^{t_i}\phi(l_i,t_i|I) \tag{37}$$

$$\psi\left(l_i,l_j|I\right) = \omega_{i,j}^{t_i,t_j} \cdot \left[x_i - x_j, \left(x_i - x_j\right)^2, y_i - y_j, \left(y_i - y_j\right)^2\right]^T \tag{38}$$

In order to evaluate the maximum of equation (36), a message-passing algorithm is employed. They iterated over all parts starting from the leaves and moving "upstream" to the root part (head) as follows, where the $\text{kids}(i)$ denotes the kid parts of part *i*.

$$\text{score}(l_i,t_i) = \varphi(l_i,t_i) + \sum_{k\in\text{kids}(i)} m_k(l_i,t_i) \tag{39}$$

$$m_i\left(l_j,t_j\right) = \max_{l_i,t_i}\left[\text{score}(l_i,t_i) + \psi\left(l_i,l_j,t_i,t_j\right)\right] \tag{40}$$

After the upstream pass, the final score of root part is determined. Then a "downstream" pass starts from the root part. By keeping track of the argmax indices, one can backtrack to find the location and type of each maximal configuration. In the downstream pass, the equation (39) and (40) can be rewritten by simply replace the $\text{kids}(i)$ with $\text{parents}(i)$.

In learning process, they prepared positive training images $\{I_n, l_n, t_n\}$ and negative images $\{I_n\}$ containing no person. Then they defined a structured prediction objective function similar to those proposed in [105] and [107] as follow, where $\omega = \left(\omega_i^{t_i}, \omega_{i,j}^{t_i,t_j}\right)$ ξ_n is the slack variables, *C* is a constant.

$$\arg\min_{\omega,\xi_n\geq 0} \frac{1}{2}\omega\cdot\omega + C\sum_n \xi_n \tag{41}$$

$$\text{S.T.}\ \forall n \in \text{pos}\ \ \text{score}(l_n,t_n) \geq 1 - \xi_n$$

$$\forall n \in \text{neg}, \forall z\ \ \text{score}(l_n,t_n) \leq -1 + \xi_n$$

In equation (41), the scores of positive examples are supposed to be larger than 1, while the scores of negative examples are supposed to be less than -1. ξ_n is applied to penalized the violation of the constraints. The training is done by the dual coordinate descent algorithm, which is proposed by [108]

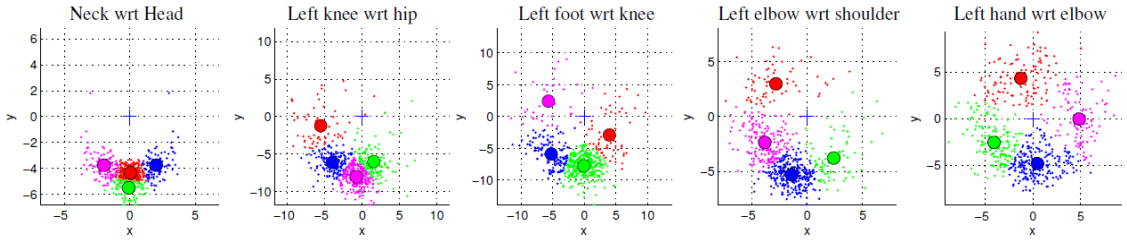


Figure 4.3 The clustering of parts in [41]. The image is directly sourced from [41]

4.2.2. Deformable Parts Model Combined with Body Orientation

In this Chapter, the DPM in [41] is extended by combining the body orientation. The frame of the extension is shown as Figure 4.4.

In Figure 4.4, the steps (b) and (c) are done in previous chapters. Based on the detected body orientation, a deformable parts model is used to estimate the articulated pose. The model is pre-trained for each orientation and will be chosen according to the

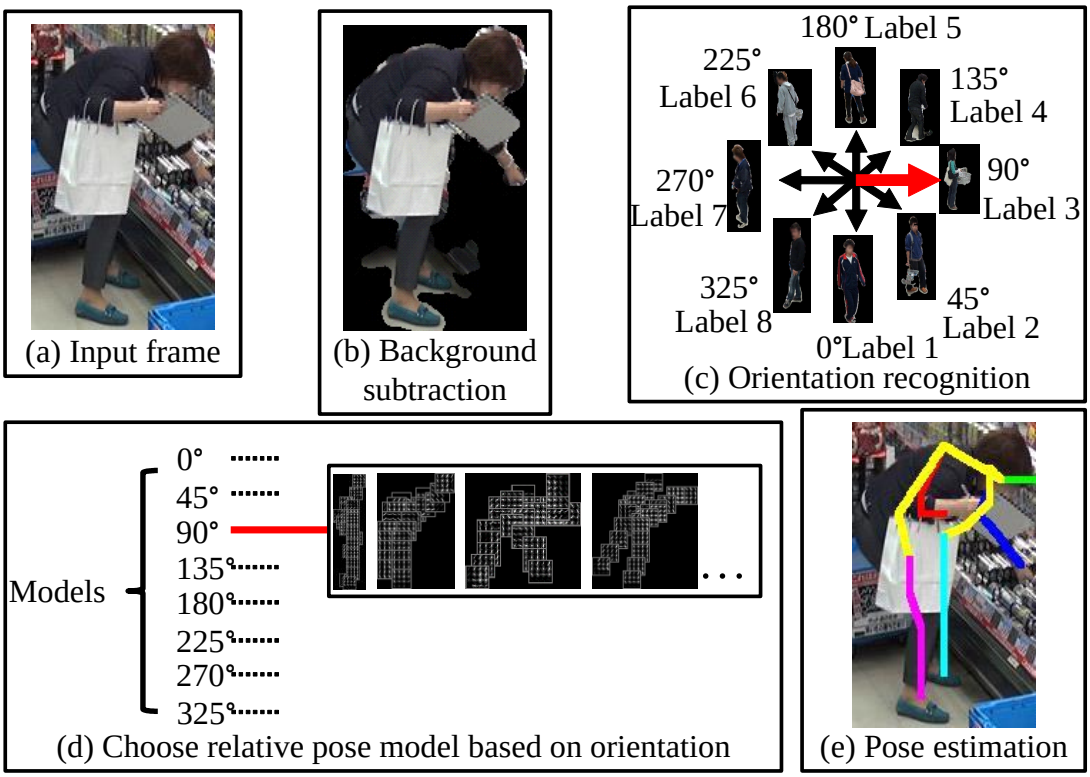


Figure 4.4 Framework of DPM estimation system combined with body orietation. (a) The input frame (b) background subtraction (c) orientation detection (d) choose relative pose model based on orientation (e) pose estimation result

body orientation recognition result. Thus the dataset can be write as $\{I_n, l_n, t_n, o_n\}$, where $o_n \in \{1, 2, \dots, 8\}$ is the orientation label. The equation (37) and (38) should be rewritten as follows.

$$\varphi(l_i, t_i|I) = \omega_i^{t_i,o} \phi(l_i, t_i|I) \tag{42}$$

$$\psi(l_i, l_j|I) = \omega_{ij}^{t_i,t_j,o} \cdot \left[x_i - x_j, (x_i - x_j)^2, y_i - y_j, (y_i - y_j)^2\right]^T \tag{43}$$

The ω in equation (41) also becomes $\omega = \left(\omega_i^{t_i,o}, \omega_{i,j}^{t_i,t_j,o}\right)$.

The typical pose model of each orientation is shown as Figure 4.5, while the shelf is at the right side. In Figure 4.5, one can roughly recognize the orientation from the models, such as the models of 0°, 90° and 270°. It shows the constraints of pose estimation based on orientation.

4.3. Pose Estimation using Deep Learning

In the Chapter 4.2, DPM method for pose estimation is introduced. The DPM method is based on the hand-crafted feature (HOG). In recent year, the deep neural networks are proved to outperform the hand-crafted feature based methods in different computer vision tasks. Thus this chapter will introduce the pose estimation using deep learning.

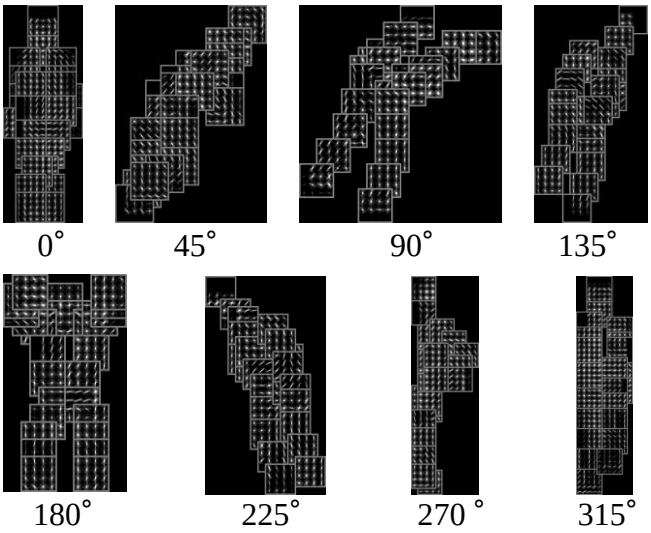


Figure 4.5 Typical pose model of each orientation

4.3.1. Deep Joint Position Learning

In recent year, some outstanding deep neural networks are proposed for classification task, such as AlexNet [47], VGG [48], GoogLeNet [49] and ResNet [23]. For pose estimation, a trivial method is applying these networks to directly produce the output of joint positions and visibility mask. In Chapter 3.3.2, the ResNet-50 shows its ability to detection the head and body orientation. In this chapter, it will be modified to estimate the customer pose.

The framework of this method is shown as Figure 4.6. The ResNet is modified to adapt the output of joint positions and visibility mask. Firstly, the original softmax layer at the output end of ResNet is removed. Secondly, the last FC layer is modified to output a vector with length 56. This vector contains two parts of information. One is the *x*, *y* coordinates of 14 joints, whose length is 14 × 2 = 28. Another is the [visibility, invisibility] scores of each joint, whose length is also 14 × 2 = 28. The [visibility, invisibility] scores are given into an additional softmax layer to produce the probabilities of visibility and invisibility. For the ground truth visibility mask, the pair of [visibility, invisibility] is [1, 0] for visible joints and [0, 1] for occluded joints. Therefore, the filter

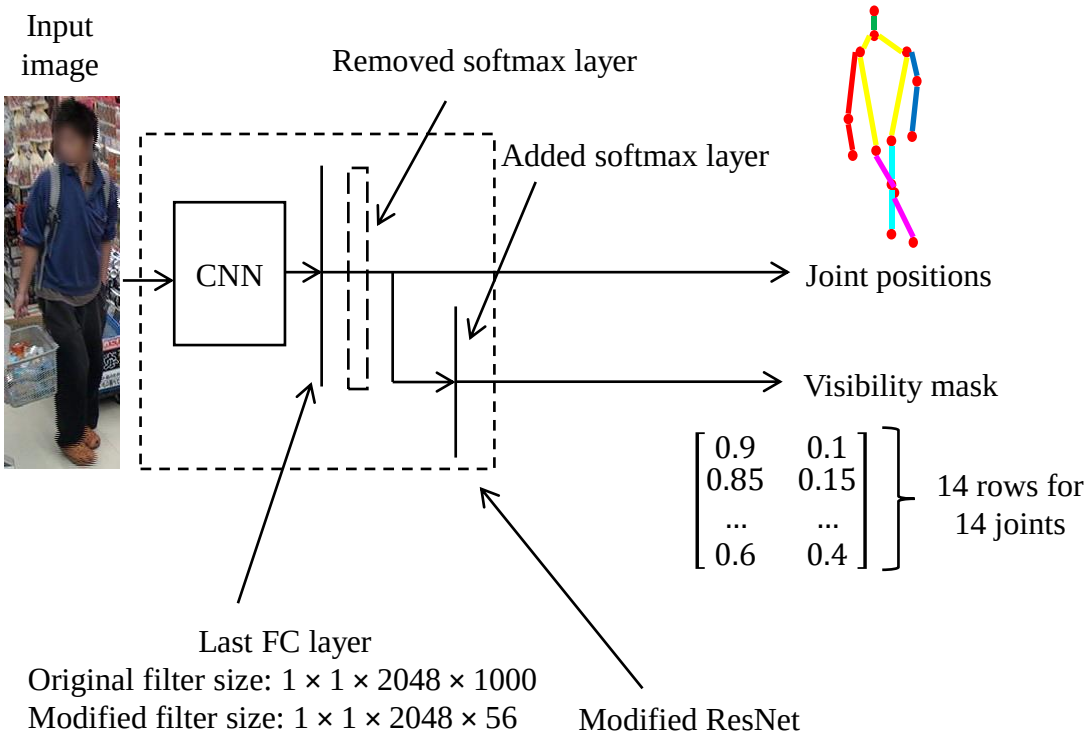


Figure 4.6 The framework of joint position estimation based on ResNet

size of last FC layer is modified t from 1 × 1 × 2048 × 1000 to 1 × 1 × 2048 × 56.

After constructing the network, the inputs and ground truth can be defined. Assume the input image is I and the pose vector is $h = (\dots, h_i, \dots)$ i $\in \{1, \dots, 14\}$, where h_i contains the x and y coordinates (x_i, y_i) of the ith joints. Because the ResNet requires the input image should be a square with size s = 224, the input image and pose are resized as follows, where Ir is the resized image, p' is the normalized the pose vector, b_w and b_h are the width and height of original input image, $b_c \in \mathbb{R}^2$ is the center position of original input image, p' is the label in learning process.

$$Ir = \text{resize}(I, s) \tag{44}$$

$$p' = \begin{pmatrix} s/b_w & 1 \\ 1 & s/b_h \end{pmatrix} (h - b_c) \tag{45}$$

Assuming p is the predicted pose of network, the final predicted pose in original image size is calculated as below.

$$p_{final} = \begin{pmatrix} b_w/s & 1 \\ 1 & b_h/s \end{pmatrix} (p + b_c) \tag{46}$$

Then loss of prediction can be calculated. The pose loss and gradient are calculated by Mean Square Error (MSE) between predicted pose p and the normalized true pose p'. The loss and its gradient are shown as below, where L_p is loss of pose and e_p is the gradient of loss.

$$L_p = \frac{1}{28} \sum_{i=1}^{14} \|p_i - p_i'\|_2^2 \tag{47}$$

$$e_p = p - p' \tag{48}$$

The loss of visibility mask is calculated by the Cross Entropy. The loss and its gradient are shown as below, where L_v and e_v is visibility loss and its gradient, v and v' is the predicted and true visibility vector respectively.

$$L_v = -\sum_{i=1}^{28} v_i \log v_i' \tag{49}$$

$$e_v = v - v' \tag{50}$$

The overall loss and its gradient are calculated as below, where α and β are the weight

of pose loss and visibility loss respectively, $[\alpha e_p, \beta e_v]$ means the concatenation of αe_p and βe_v. Empirically, when α = 1 and β = 10, the performance reaches the peak.

$$L = \alpha L_p + \beta L_v \tag{51}$$

$$e = [\alpha e_p, \beta e_v] \tag{52}$$

This network can be also improved by introduced the body orientation as Figure 4.7. In Figure 4.7, the last FC layer not only receives the output of previous layer, but also the body orientation vector. The filter size of the last FC layer also becomes 1 × 1 × 2048 × 56 × 8. The output of the last FC layer is calculated as below, where o_i is the element of orientation vector, operator $\otimes$ means the convolution.

$$output_{fc} = output_{previous} \otimes (\sum_{i=1}^{8} w(:,:,:,:,i)o_i) + \sum_{i=1}^{8} b(:,i)o_i \tag{53}$$

The experiments in Chapter 4.6 shows that the ResNet + Body Orientation method outperforms the ResNet only method and the DPM + Body Orientation method. However, there is a limitation in these trivial deep learning method: they only give one pose prediction per image. For example, Figure 4.8 is an example result of not successful hand detection using ResNet + Body Orientation. Even though the elbow positions are relatively accurate, it is difficult to correct the hand positions based on elbow positions,

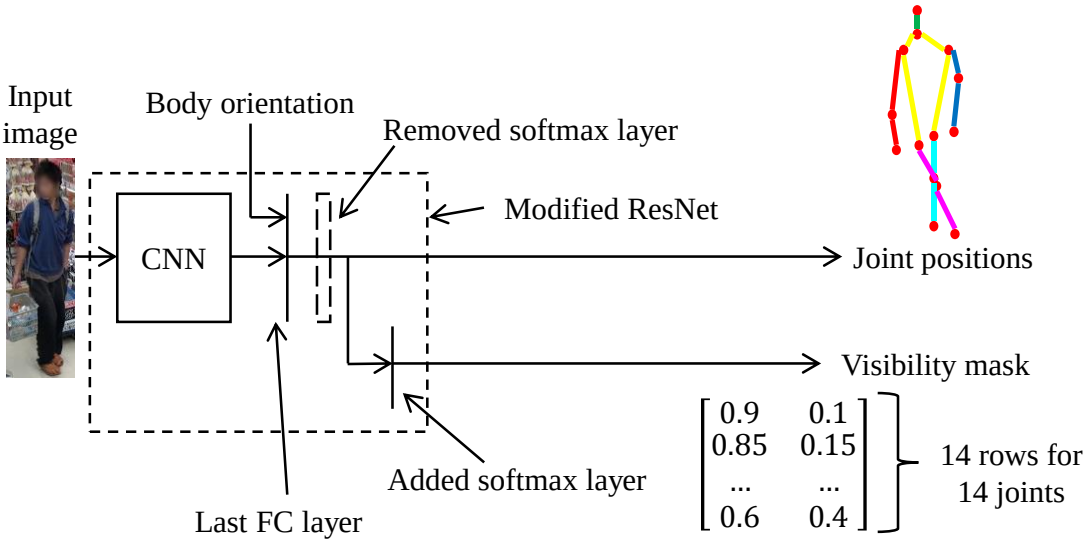


Figure 4.7 The framework of joint position estimation based on ResNet combining body orientation

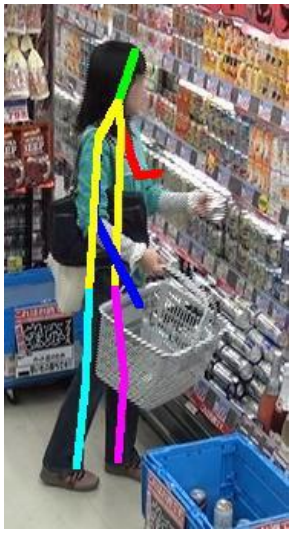


Figure 4.8 An example of not successful hand detection using ResNet + Body Orientation. It is difficult to correct the hand positions based on the single prediction

because the single prediction without any confidence information or candidates could not provide any extra information. One solution is to generate joint probabilistic heatmaps. Therefore, Chapter 4.3.2 will propose a joint heatmap estimation using deep learning.

4.3.2. Deep Joint Heatmap Learning

As the discussion of Chapter 4.3.1, the main idea behind the joint heatmap is to increase the prediction candidates. Based on the heatmap, many flexible correction methods could be applied. For example, a set of message-passing layer can correct the joint heatmap by modeling the relationship between neighboring joints.

There are many methods to generate the heatmaps. One kind of methods is to trained the CNN with local image patches and tested by sliding window to generate the joint heatmaps [53] [54]. That means these methods convert the joint estimation problem to a pixel-wise classification problem. In these pixel-wise classification methods, the training and testing are time consuming. Besides, the negative patches are far more than the positive patches in the image, the number of negative patches usually need to be manually controlled. To cover this shortage, another kind of methods is to extend the popular CNN, such as GoogLeNet or VGG, to a Full Convolutional Network (FCN).

The mainly different between FCN and conventional CNN is that FCN replaces the Fully Connected (FC) layer in CNN to a 1 × 1 size convolution layer. Therefore, FCN can receive the whole image as input, rather than the local image patch, to directly generate a feature map. According to different applications, the FCN could be trained to generate different semantic heatmaps. For example, Chu *et al*. [58] extend the VGG-16 [48] to an FCN-VGG to generate joint heatmaps. Shelhamer *et al*. [109] proposed a series of general methods to extend CNN to FCN, while keep the same ability of pixel-wise classification method. Then they found that the FCN-VGG performs much better than FCN-GoogLeNet in segmentation task. Furthermore, they proposed their own FCN-8s, which outperforms both FCN-VGG and FCN-GoogLeNet in segmentation task.

Considering the good performance of FCN-8s in [109], this chapter will propose the joint heatmap learning based on FCN-8s. The overall framework is shown as Figure 4.9.

The customer image is given into a Fully Convolutional Networks (FCN) to generate initial joint heatmaps. Based on these heatmaps and body orientation input, a set of Orientational Message Passing (OMP) layers are constructed. These layers model the global pose configuration from orientation and the local neighboring joint connections from the generated heatmaps. After refining the joint heatmaps through the OMP layers, a BRNN is applied to model both the forward and backward temporal constraints of joint heatmaps. Finally, a softmax layer is used to normalize the heatmaps to probabilistic distribution. The output joint position is the centroid of each joint heatmap. If the maximum score in the joint heatmap is less than a threshold, this joint will be treated as occluded. Empirically, the best threshold is 0.55. The occluded joints are not shown in Figure 4.9.

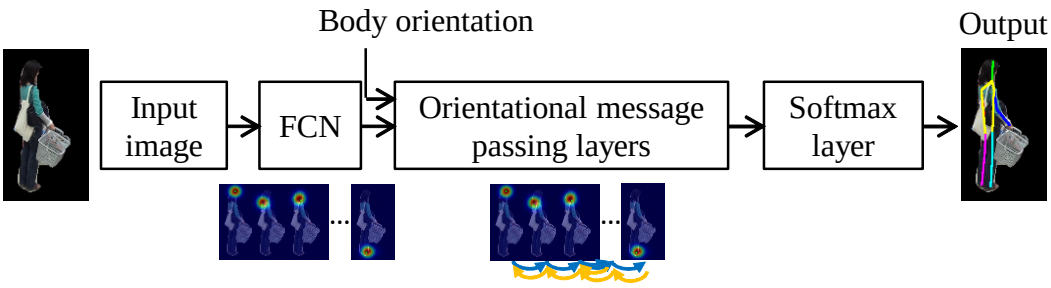


Figure 4.9 The framework of deep joint heatmap learning

Structure of FCN

Firstly, here will introduce the FCN-8s in [109]. Figure 4.10 shows the structure of FCN-8s. In Figure 4.10, the image is given input into the network and its feature size becomes smaller and smaller (one small block in Figure 4.10 means one unit size). The feature size becomes 1/8, 1/16 and 1/32 of original input size after the pool3 layer, pool4 layer and conv7 layer respectively. At last, the 4× upsampled conv7, 2× upsampled pool4 and the pool3 are stacked together. The upsampling is archived by the deconvolution. This structure is aimed to combine the features of different resolution to keep both the localization accuracy and the edge accuracy in segmentation. At last, the stacked features are 8x upsampled to generate the prediction. The prediction size is the same as the input image. Because the FCN-8s is originally used for segmentation of 20 classes object, the channel of prediction is 20. The value of each channel is the score of that class. An additional pixel-wise softmax layer can be added on top of the prediction to generate the probability of each class. The class with maximum score of each pixel can be determined and the segmentation is then finished. An example result of FCN-8s is shown as Figure 4.11.

The original FCN-8s is applied for segmentation. If a joint area can be seen as a class, the FCN-8s can be also used for pose estimation. To this end, FCN-8s is modified in this dissertation. The modified structure for pose estimation is illustrated as Figure 4.12. Compared to the original FCN-8s, a dropout layer is inserted after the pool3, pool4 and conv7 layer respectively. The stacked feature maps are also derived after these dropout layers. The dropout layers are applied to prevent the overfitting in training. The dropout rates of dropout3, dropout4 and dropout7 are set to 0.1, 0.3 and 0.5 respectively.

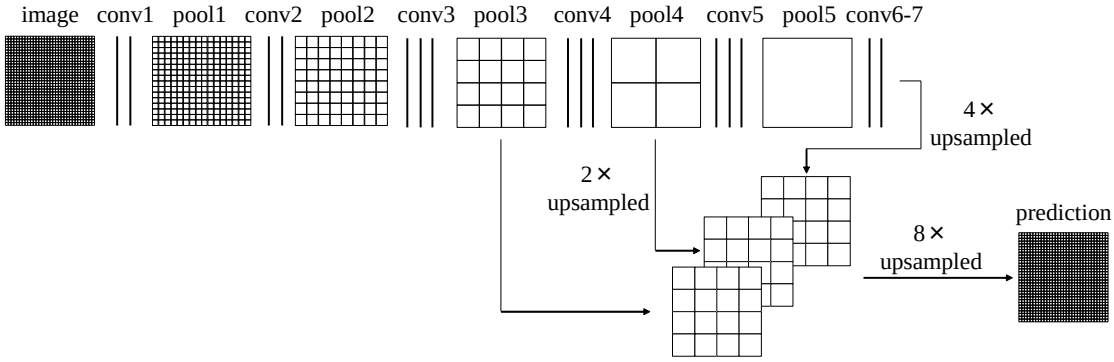


Figure 4.10 The structure of FCN-8s in [109]

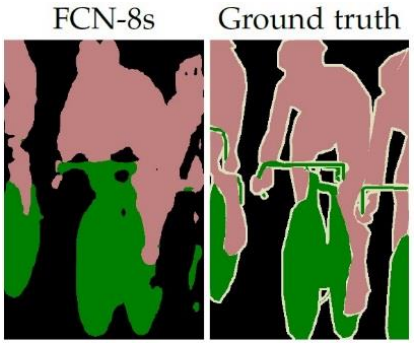


Figure 4.11 An example result of FCN-8s in [109]. The image is directly sourced from [109]

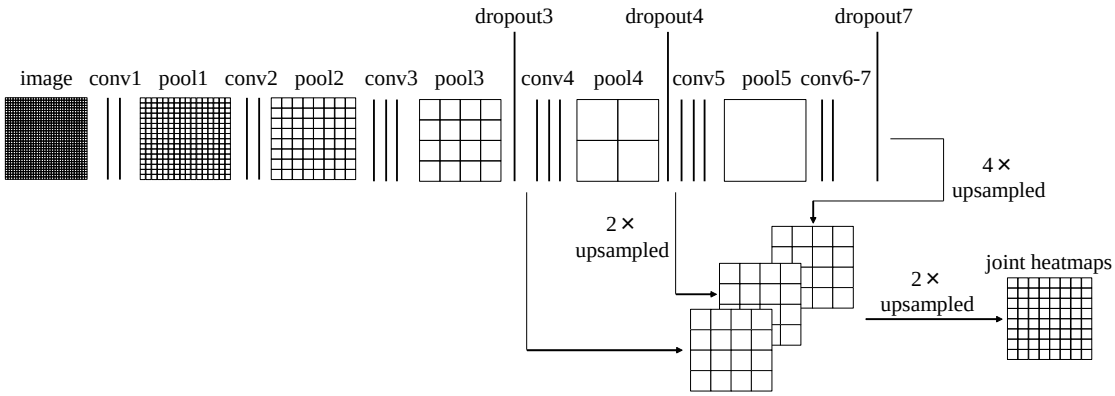


Figure 4.12 The structure of modified FCN-8s for pose estimation

Another modification is that the last upsampling is 2 times for reducing computation. This means the output prediction size is 1/4 of the original image size. Although the FCN-8s does not restrict the input image size, the input customer images are still normalized to average size 200 × 366 for convenience. Thus the size output heatmap is 50 × 92. For comparison, the size of output heatmap is 56 × 56 [58] and the height of output heatmap is 38 in [57]. The channel number of prediction is $14 \times T + 1$, where 14 is the number joints, T is the types number of Mixture-of-Parts, plus one means the background class. The Mixture-of-Parts will be explained later.

In order to visualize the joint heatmaps as Figure 4.9, a pixel-wise softmax layer could be added on top of the FCN. The softmax layer converts the output of FCN to probability distributions of $14 \times T + 1$ classes. Then by excluding the background class and summing up the probabilities of T types, the joint heatmaps with size 50 × 92 × 14 can be obtained. Finally, the heatmaps could be resized to the same size of original input image. The example of joint heatmap can be seen in Figure 4.9

Mixture-of-Parts

In order to model the relationship between neighboring joints, this dissertation also applies the concept of Mixture-of-Parts which is proposed by [41]. The idea of Mixture-of-Parts is to cluster the joints into *T* types according to its relative position to its neighboring joint. For example, the left shoulders in front view and in back view probably belong to different clusters, because the left shoulder is at the right bottom of neck in front view and is at the left bottom of neck in back view. That means the type of Mixture-of-Parts gives an inference of position of neighboring joint. Therefore, different types of joints can send different messages to its neighboring joint in the message passing layers. The message passing layers will be discussed later in this chapter.

In [41], [57] and [58], the joints are clustered by using K-means algorithm. However, there are two drawbacks of K-means algorithm. The first one is that it only give hard clusters. The second one is that it tends to generate equal sized clusters, thus it does not work well in the distribution with unbalanced density. To solve the first problem, some extensions of K-means could be used to generate soft clusters, such as Fuzzy C-means algorithm and Expectation Maximization (EM) algorithm. However, the Fuzzy C-means algorithm still cannot solve the second problem, because the algorithm is also based on the distance like K-means. In contrast, EM algorithm can solve the second problem by benefiting the Gaussian Mixture Models (GMM). Therefore, this dissertation chooses the EM algorithm for the joint clustering.

To implement the GMM clustering, firstly the normalized relative positions of joints need to be calculated. For image *m*, the relative position of joint *i* to the neighboring joint *j* is calculated as below, where p_i and p_j are the absolute position of joint *i* and joint *j*.

$$r_{ij,m} = p_{i,m} - p_{j,m} \tag{54}$$

Here, the structure of joints is seen as a tree and *head* is the root. The structure can be seen in Figure 2.5. The index *j* is defined as the parent joint of *i* in the tree structure. For joint *head*, the neighboring joint is defined as joint *neck*. This relative position $r_{ij,m}$ will be normalized by the average torso length. The torso length and average torso length are defined as below.

$$tl_m = \left(\left|p_{lshou,m} - p_{lhip,m}\right| + \left|p_{rshou,m} - p_{rhip,m}\right|\right)/2 \quad (55)$$

$$tl' = average(tl_m) \quad (56)$$

where $p_{lshou,m}$ and $p_{rshou,m}$ are the positions of left and right shoulders, $p_{lhip,m}$ and $p_{rhip,m}$ are the positions of left and right hips, tl_m is the torso length of image m, tl' is the average torso length on the training data set. Then the normalized relative position is calculated as follow.

$$r'_{ij,m} = r_{ij,m} \times tl'/tl_m \quad (57)$$

Based on the normalized relative positions, the GMM is fitted by Expectation-Maximization (EM) algorithm. The initial centroids of EM algorithm are generated by K-means++ [110] algorithm. The number of component in GMM is *T*. The *T* is also the number of Mixture-of-Parts. The fitted GMM will assign a set of scores to each joint. The score means the probability corresponding to each component in GMM. Following the setting in [57] and [58], this dissertation also sets *T* as 13.

Soft Label and the Loss Function

Based on the soft clusters, the labels of training data also became soft labels, which can produce more reasonable estimation. For each joint, the ground truth is not only a position, but also with *T* channels. At the joint position, the values of corresponding *T* channels are the *T* probabilities of the soft clusters. Therefore, the labels of training data have $14 \times T + 1$ channels. In the non-joint pixels, the label is a one-hot vector, where the value of background channel is 1. For the occluded joint, the label is given as the background.

Furthermore, this dissertation extends he joint position from one pixel to a circle area, because this dissertation considers the joint is more like an area in the image rather than one single pixel. In the output scale 50×92, the experiments on the circle radius as 1 pixel (equals to single pixel), 3 pixels and 5 pixels are conducted. As a result, the Percentage of Correct Keypoint (PCK) of 3 pixels radius achieved the highest value. The correct keypoint is defined as the predicted joint whose distance from true joint is less than a given threshold. When the threshold is the 0.09 times the body height, the PCK

of 3 pixels radius was 5% higher than the PCK of 1 pixel radius and 1% higher than the PCK of 5 pixels radius. Therefore, this dissertation chooses 3 pixels as the radius of the joint area. If there are n joints circle areas are overlapped, the value of each channels should be divided by n, in order to keep the sum of probabilities is 1.

The tensor of ground truth can be seen as an extension of pixel-wise labeling as Figure 4.13. In each pixel of the ground truth, there is an $14 \times T + 1$ length vector in the z coordinate. For joint n, the values of elements from index $(n - 1) \times T + 1$ to index $n \times T$ are the probabilities of T clusters. For the background pixel, the value of $(14 \times T + 1)$th element is 1 and the values of other elements is 0.

For the probabilistic labels, the cross-entropy is suitable to evaluate the loss. Assume the output heatmap in map_f. Then the probability map map_p is calculated by a softmax function as follow, where l is the location in the map.

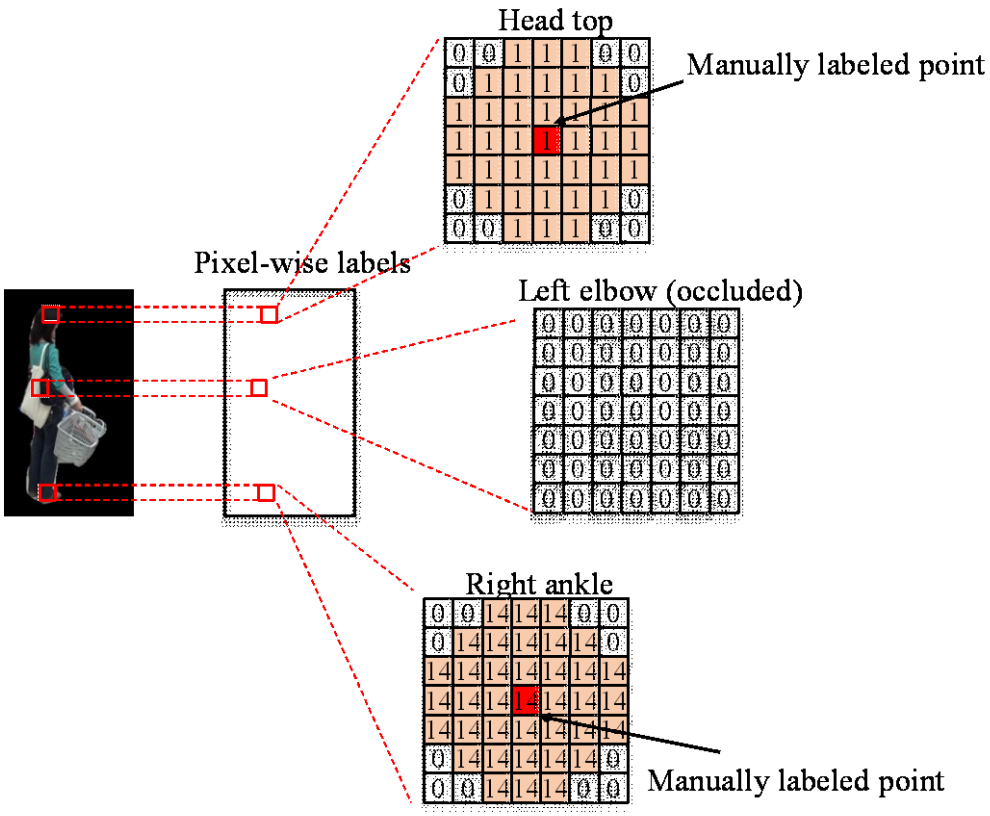


Figure 4.13 An example of pixel-wise labeling. The ground truth tensor can be seen as an extension of this labeling to $14 \times T + 1$ channels. For each pixel, there is an $14 \times T + 1$ length vector label

$$map_p(l) = \frac{exp\left(map_f(l)\right)}{\sum_l exp\left(map_f(l)\right)} \tag{58}$$

Thus the loss is calculated as the follow, where $c = \{1, 2, \dots, 14 \times T + 1\}$ is channel index of the map, where $\text{label}(l, c)$ is the ground truth value at location *l* and channel *c*.

$$\text{loss} = -\sum_{l,c} \text{label}(l, c) \log\left(map_p(l, c)\right) \tag{59}$$

The gradient of loss is calculated as equation (60)

$$e = map_p - \text{label} \tag{60}$$

Orientational Message Passing Layers

The Orientational Message Passing Layers are inspired by [58]. In [58], the message passing layers are implemented by a series of convolutions. Here briefly introduce the message passing layers in [58].

As shown in Figure 4.14, there are upstream pass and downstream pass in the message passing layer. Assume A_i and A_i' are the feature maps before and after upstream pass, B_i and B_i' are the feature maps before and after downstream pass, where $i = \{1, 2, \dots, 14\}$. Before the message passing starts, $A_i = B_i$.

In the upstream pass, the A_i' is calculated as below, where *j* is the child index of joint *i*, operator $\otimes$ means the convolution, $w^{ai,aj}$ is the filter of convolution, $f()$ is an activation function. The order of calculation follows the upstream in Figure 4.14.

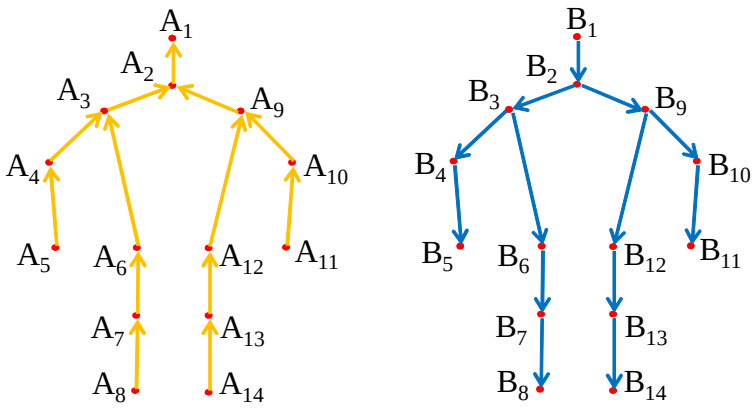


Figure 4.14 The upstream pass and downstream pass of message passing

$$A'_5 = A_5,\ A'_8 = A_8,\ A'_{11} = A_{11},\ A'_{14} = A_{14} \tag{61}$$

$$A'_{i\notin\{5,8,11,14\}} = f\left(A_i + \sum_j A'_j \otimes w^{ai,aj}\right) \tag{62}$$

In the downstream pass, the B'_i is calculated as below, where *j* is the parent index of joint *i*, $w^{bi,bj}$ is the filter of convolution. The order of calculation follows the downstream in Figure 4.14.

$$B'_1 = B_1 \tag{63}$$

$$B'_{i\notin\{1\}} = f\left(B_i + B'_j \otimes w^{bi,bj}\right) \tag{64}$$

The size of $w^{ai,aj}$ and $w^{bi,bj}$ should cover the max shift between neighboring joints. In [58], the $w^{ai,aj}$ and $w^{bi,bj}$ are implemented by a series of cascade filters to cover a large area and meanwhile keep relatively small size of parameters. The max shift between neighboring joints in the customer video are 38 pixels in output size of feature map. Thus in this dissertation, the $w^{ai,aj}$ and $w^{bi,bj}$ are constructed by 10 cascade filters with size 9, which can cover maximum $(9-1)/2 \times 10 = 40$ pixels shift. Thus these cascade filters equal to a filter with size $S_{mp} \times S_{mp} \times T \times T$, where $S_{mp} = 40$.

The result of message passing is calculated as equation (65), where $[A'_i, B'_i]$ is the concatenation of A'_i and B'_i, w_s is the filter of convolution. The $score_i$ has the same number of channels as A'_i and B'_i. Figure 4.15 illustrated an example process of the message passing algorithm in [58], where the white points in the heatmaps are the predicted positions of joints. In Figure 4.15, the initial heatmap of right hip is diverse and the predicted position is not accurate. After receiving the messages from its neighboring joints (right shoulder and right knee), the heatmap of right hip became more focused and the predicted position is also corrected.

$$score_i = [A'_i, B'_i] \otimes w_s \tag{65}$$

Based on the message passing layers of [58], this dissertation introduces the Orientational Message Passing (OMP) layers to utilized the body orientation information. The OMP layers could solve the problem of left-right similarity. For example, as shown in Figure 4.17(a), the left shoulder is wrongly located at the position of right shoulder before the message passing layers, because of the similarity between left shoulder in

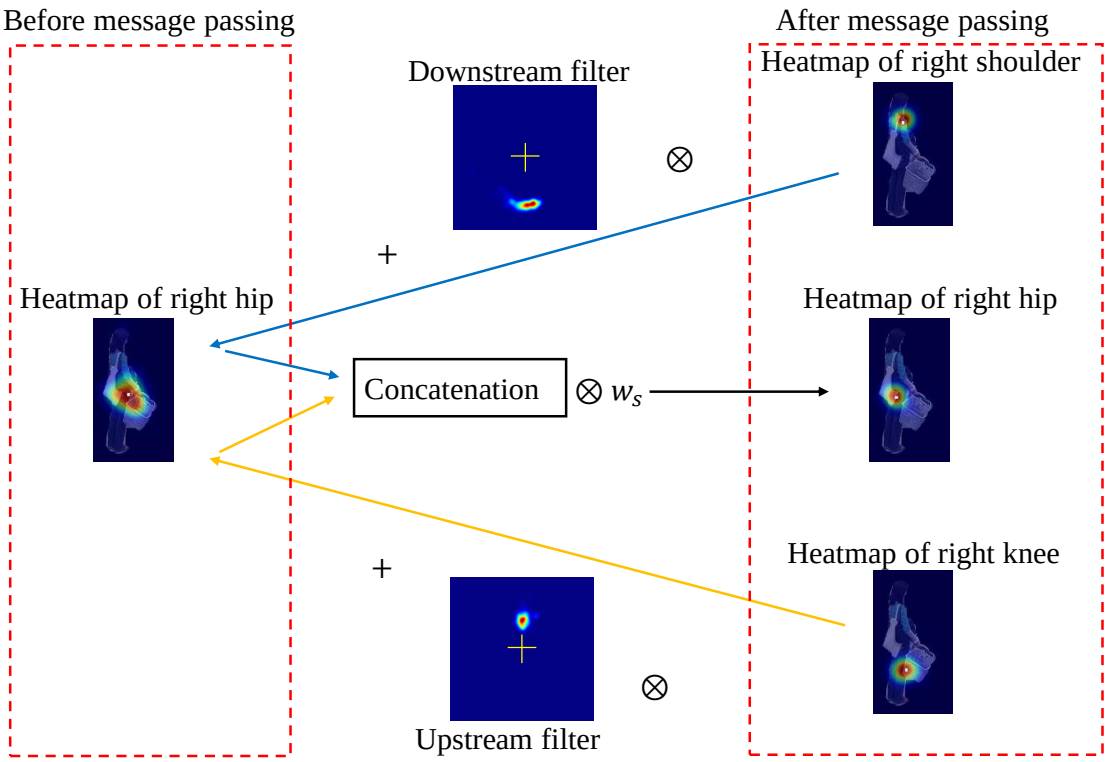


Figure 4.15 An example process of message passing algorithm in [58]. The heatmap of right rip receives the messages from its neighboring joints (right shoulder and right knee). The white points in the heatmaps are the predicted positions of joints

back view and the right should in front view. However, the traditional message passing layer cannot correct this error as Figure 4.17(b), because this position could be a totally reasonable configuration of left shoulder in back view. On the other hand, the OMP layers receive the information of body orientation. When the customer is facing the camera, the left shoulder should be at the lower right position of neck. Thus, there should be an orientational message is sent from neck to left shoulder to imply the position of left shoulder. The orientational message could be a strong cue to solve the problem of left-right similarity. The corrected heatmap of left shoulder is shown as Figure 4.17(c).

In order to construct the orientational message, the relative position histograms of joints should be counted at first. Assume h_{pu} and h_{pd} are the relative position histogram for upstream and downstream. They are defined as 5-D tensors. The first and second dimensions represent the *x* and *y* relative positions of neighboring joint. The third dimension means the cluster index of the neighboring joint. The fourth dimension is the

joint index. The fifth dimension means the body orientation. The relative position histograms are voted by each joint in the training data. For each orientation, h_p is smoothed by a 3D Gaussian filter and normalized its sum as 1. The calculation is shown as below, where $g()$ is a 3D Gaussian filter, $k = \{1, 2, \dots, 8\}$. The h'_{pu} and h'_{pd} are called statistical relative heatmaps.

$$h'_{pu}(:,:,:,:,k) = \frac{g(h_{pu}(:,:,:,:,k))}{\sum g(h_{pu}(:,:,:,:,k))} \tag{66}$$

$$h'_{pd}(:,:,:,:k) = \frac{g(h_{pd}(:,:,:,:,k))}{\sum g(h_{pd}(:,:,:,:,k))} \tag{67}$$

Then the orientational downstream filter and upstream filter can be calculated as follows, where o_k is the probability of the *k*-th orientation from the input orientation vector.

$$w_{ou} = \sum_{k=1}^{8} h'_{pu}(:,:,:,:,k) \times o_k \tag{68}$$

$$w_{od} = \sum_{k=1}^{8} h'_{pd}(:,:,:,:,k) \times o_k \tag{69}$$

Thus the orientational messages are calculated as below, where $t = \{1, 2, \dots, T\}$, $i = \{1, 2, \dots, 14\}$, m_{ui} and m_{di} are the upstream and downstream orientational message from joint *i*, the operator $\otimes$ means the convolution. The third dimension of w_{ou} and w_{od} can be seen as multiple filters in the convolution. The summation on mixture type dimension means that the orientational message is only related with position of source joint, and does not consider the mixture type of source joint.

$$m_{ui} = sum_t\big(A'_i(:,:,t)\big) \otimes w_{ou}(:,:,:,i) \tag{70}$$

$$m_{di} = sum_t\big(B'_i(:,:,t)\big) \otimes w_{od}(:,:,:,i) \tag{71}$$

Then the equation (62) and (64) can be rewritten as equations (72) and (73), where α and β are the relative weight of the local pair-wise message and orientational message, $w^{ai,aj}$ and $w^{bi,bj}$ are the filters of convolutions in upstream and downstream, which are the same as those in equations (62) and (64). Empirically, when $\alpha_a = \alpha_b = 1$ and $\beta_a = \beta_b = 5$, the system achieves best performance.

$$A'_{i \notin \{5,8,11,14\}} = f\big(A_i + \sum_j \big(\alpha_a A'_j \otimes \big(w^{ai,aj} \times o\big) + \beta_a m_{uj}\big)\big) \tag{72}$$

$$B'_{i\notin\{1\}} = f\left(B_i + \alpha_b B'_j \otimes \left(w^{bi,bj} \times o\right) + \beta_b m_{dj}\right) \quad (73)$$

The right sides of equations (72) and (73) include 3 parts. The first part is the heatmap before message passing (A_i or B_i). The second part is the conventional pair-wise message ($A'_j \otimes w^{ai,aj}$ or $B'_j \otimes w^{bi,bj}$). The third part is the orientational message (m_{uj} or m_{dj}). Figure 4.16 shows a detailed example of downstream orientational message passing including these 3 parts from neck to right shoulder. In Figure 4.16, the green point is the ground truth joint position. The probability of each body orientation is obtained from ResNet based SSL algorithm for body orientation classification in Chapter 3. In order to visualize the statistical relative heatmaps and messages, here sums up their third (cluster) dimension and shows them as normalized heatmaps. Figure 4.17 illustrates the comparison of the proposed OMP with conventional message passing. In Figure 4.17, the message passing algorithm in [58] cannot correct the left-right similarity problem, because the 'left shoulder is at the bottom left of neck' is a totally reasonable configuration, as long as the person is in back view. Different from that, the OMP layers incorporate the 'front view' information. Therefore, the system can predict the left shoulder should be at the right bottom of neck.

4.4. Integral Pose Estimation

The framework of the IntePoseNet is shown as Figure 4.18. In the input end, taking the advantage of surveillance camera, the background of video can be easily subtracted by using Multi-Layer Background Subtraction algorithm [90]. After the background subtraction, there are 3 streams. In the first stream, the image is fed to an FCN to generate the initial joint heatmaps. Based on these heatmaps, this chapter constructs a set of OMP layers to incorporate the body orientation with the pair-wise joint model. The body orientation is obtained from the ResNet based SSL in Chapter 3.3.2. It is a vector including the probabilities of 8 body orientations. The second stream is the occluding object detection. In retail store, the main occluding object is the shopping basket. The basket heatmap is generated by the ResNet [23]. Because the ResNet is trained from ImageNet dataset [111], it can classify the 'shopping basket' object. Thus the basket heatmap can be obtained by applying a sliding window method with ResNet. In the third stream, the optical flow is calculated by the FlowNet [112] using two continuous images.

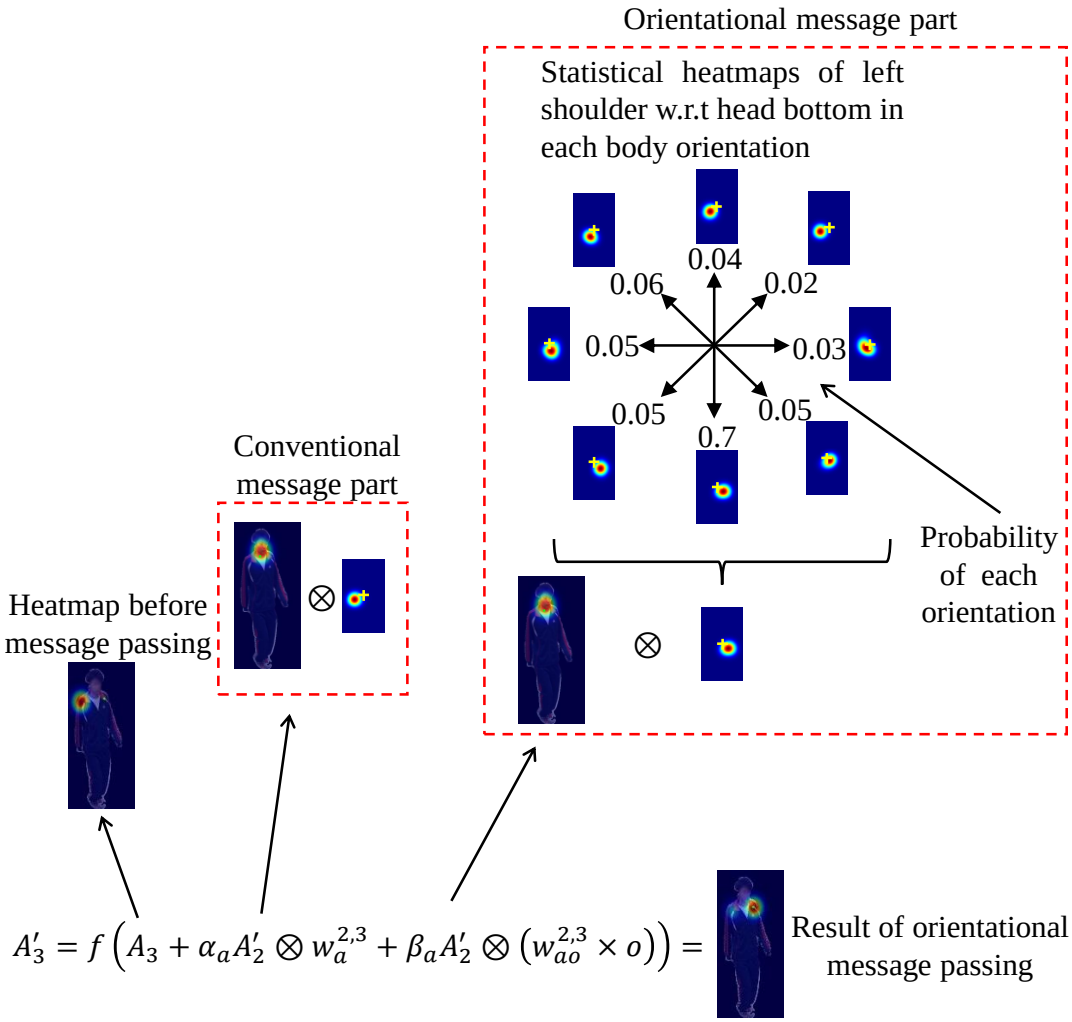


Figure 4.16 An example of downstram orientational message passing from neck to right shoulder. The green point is the ground truth joint position. The statistical relative heatmaps are counted from training data

Then the optical flow, basket heatmap and the output of OMP layers are combined with 3 convolution + Rectified Linear Unit (ReLU) layers. On the top of the network, a BRNN is employed to produce the refined heatmap by optimizing the temporal consistency in both forward and backward sequences. For each output joint heatmap, the joint position is calculated as the centroid of the heatmap. The visibility mask is determined by whether the maximum value of each heatmap is larger than a threshold.

4.4.1. Optical Flow for Pose Estimation

Optical flow is a pattern of apparent motion of objects in two images. Assume that

(a) (b) (c)

Figure 4.17 The heatmaps of left shoulder (a) before message passing layers (b) after traditional message passing layers (c) after orientational message passing layers. The green point is the ground truth joint position

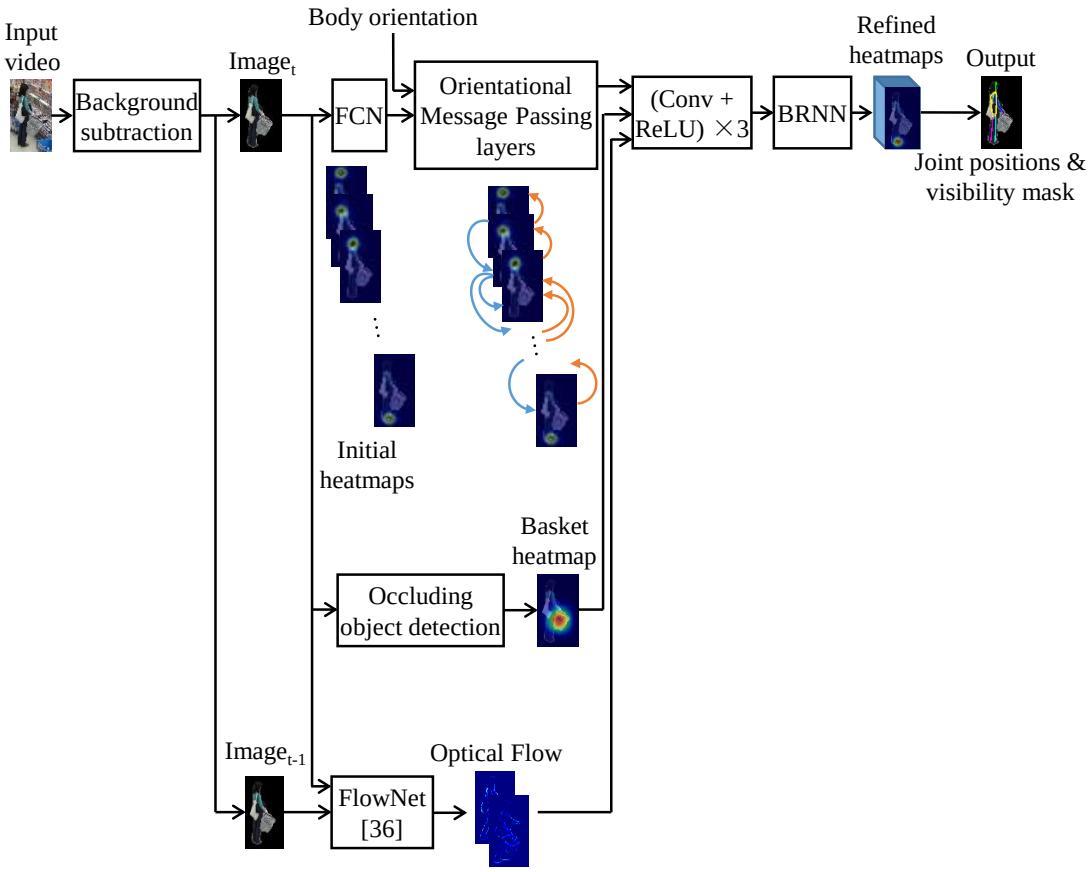


Figure 4.18 The structure of Integral Pose Network (IntePoseNet)

in the XYT space, a voxel at location (x, y, t) with intensity $I(x, y, t)$ moves by $(\Delta x, \Delta y, \Delta t)$ in two images. Thus the brightness constancy constraint can be given as

below.

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{74}$$

Assuming the movement to be small, the image constraint at $I(x, y, t)$ with Taylor series can be developed as follow, where ε is the high order term.

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \varepsilon \tag{75}$$

From equation (74) and (75), it follows that

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{76}$$

or

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0 \tag{77}$$

which results in

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \tag{78}$$

where V_x and V_y are the *x* and *y* components of the optical flow of $I(x, y, t)$, $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x, y, t) in the corresponding directions. I_x, I_y and I_t can be written for the derivatives in the following.

$$I_x V_x + I_y V_y = -I_t \tag{79}$$

The state-of-art method to calculate the optical flow is the FlowNet [112]. The structure of FlowNet is shown as Figure 4.19. FlowNet constructs two streams to extract the features from two images respectively. With this architecture, the network is constrained to firstly produce meaningful representations of the two images separately and then combine them on a higher level.

This dissertation performs optical flow calculation as a pre-processing step. The input of FlowNet is a pair of continuous images which are resized to 200 × 366. The output is a 200 × 366 × 2 tensor. The third dimension of the output tensor means the *x* and *y* component of optical flow. In order to fit the size of joint heatmaps, the optical flow tensor is resized to 50 × 92 × 2. Before given input into the IntePoseNet, the optical

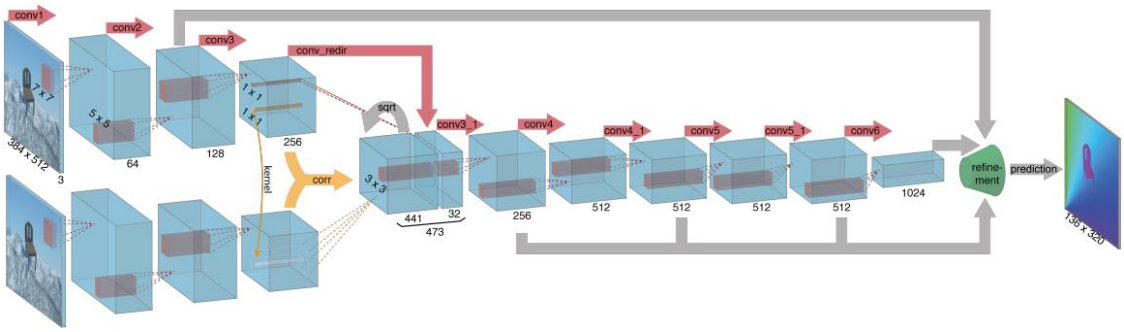


Figure 4.19 The structure of FlowNet. This image is directly sourced from [112]

flow applies the Local Motion Normalization (LMN), which is proposed by [59]. TheLMN is formulated as the local subtraction with the response from a Gaussian kernel with large standard deviation. The aim of LMN is to remove the unwanted background motion as well as normalizing the local intensity of motion. At last, the optical flow is converted into the abstract value, because only the position of joint is important for pose estimation.

4.4.2. Basket Heatmap for Pose Estimation

In order to explicitly give the information of occluding object, this dissertation applies the shopping basket detection, because the shopping basket is the main occluding object in retail store through observation. The shopping cart is not considered in this dissertation, because the shopping cart is constructed by multiple thin bars in the customer dataset. As shown in Figure 4.20, the main occluding object is still the basket on the shopping cart.

As the introduction before, the ResNet is originally used to classify 1000 kinds of objects. The 791st object is the shopping basket. Thus the original ResNet can be used for basket detection. The process is shown as Figure 4.21 Firstly, the image is resized to different scale to construct an image pyramid. For each scale, a set of windows with different width-height ratios slides on the image. The image patch in the window is fed to ResNet and generates the score of shopping basket class. Therefore, the after applying the sliding window in images with different scales, a feature pyramid can be obtained. Then the feature layers in the feature pyramid are resized to the largest size and then are stacked together to construct a feature map. Each layer of this feature map indicates the

Figure 4.20 The example image of shopping basket and shopping cart. The main occluding object is the shopping basket

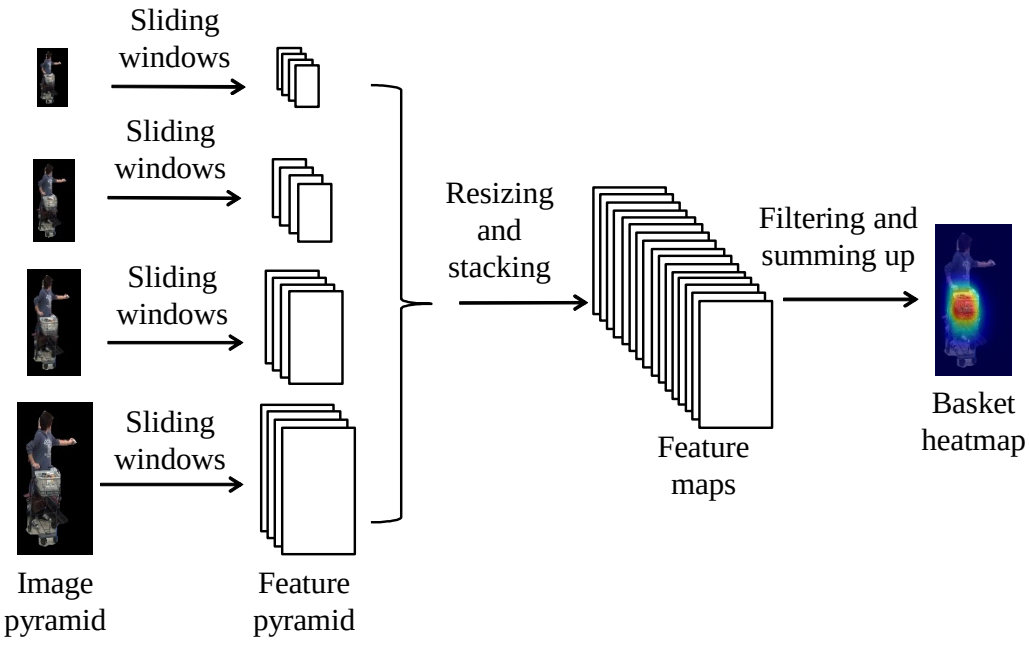


Figure 4.21 The process of basket detection

position of the center of basket. In order to generate the basket area but not only its center, each layer of the feature map is applied an average filter with the window size. The filtered feature layers are summed up to construct the final basket heatmap.

In the implementation, the image pyramid contains 8 scales and the sliding windows include 8 different width-height ratios. Thus the feature maps contain 64 layers. The stride of sliding window is set as 4 in both *x* and *y* directions. Therefore the size of

basket heatmap is 1/16 of original image size, which is the same as the size of joint heatmaps.

4.4.3. Integration of Joint Heatmaps, Optical Flow and Basket Heatmap

The joint heatmaps, optical flow and basket heatmap are combined in the IntePoseNet. As shown in Figure 4.22, these information are stacked together and then given into 3 convolutional + ReLU layers.

The number of channels of joint heatmaps is $14 \times T + 1 = 183$, where $T = 13$ is the number of Mixutre-of-Parts. The optical flow has 2 channels and the basket heatmap has 1 channel. Thus the input the 3 convolutional layers has $183 + 2 + 1 = 186$ channels. The filter sizes of the 3 convolutional layers are listed in Table 4.1. The output of the 3 convolutional layers becomes to 183 channels again. Figure 4.23 shows the effects of adding optical flow and basket heatmap. In the first row of Figure 4.23, benefiting the information optical flow, the predicted position of left wrist is corrected to the moving part. In the second row of Figure 4.23, the basket heatmap indicates the occlusion of left hip. Thus the result heatmap of left hip becomes 'colder'.

4.4.4. Pose Estimation using Bidirectional RNN

In order to model the customer pose in video sequence, this dissertation uses Bidirectional RNN (BRNN) to integrally consider the forward and backward temporal

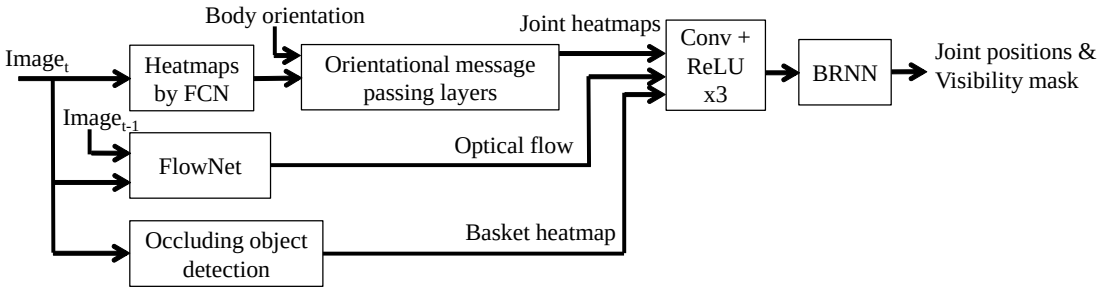


Figure 4.22 A concise version of the structure of IntePoseNet

Table 4.1 The filter sizes of the 3 convolutional layers for information combination

	Conv1	Conv2	Conv3
Size of filter	5 × 5 × 186 × 512	1 × 1 × 512 × 256	1 × 1 × 256 × 183

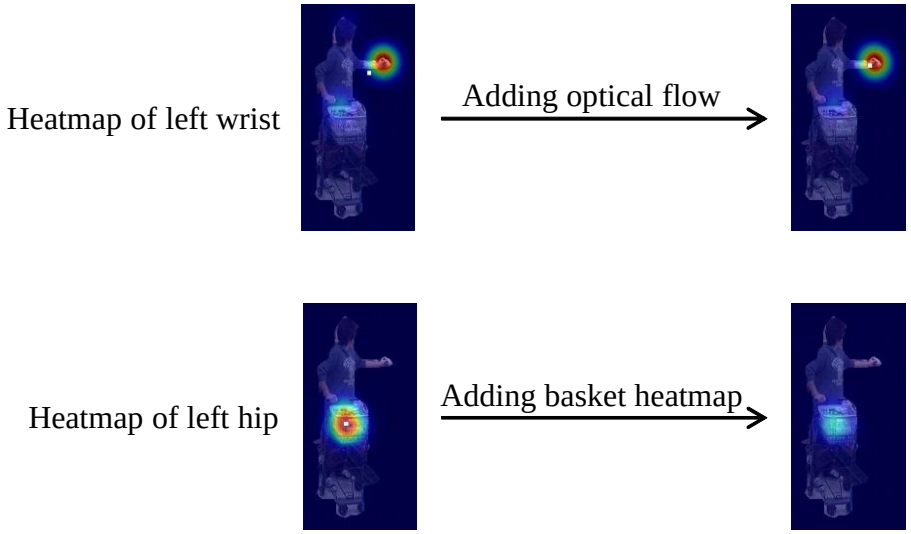


Figure 4.23 The effects of adding optical flow and basket heatmap. The white points in the heatmaps are the predicted positions of joints

constraints. Before explaining the pose estimation using BRNN, the RNN will be briefly introduced here.

The Recurrent Neural Network is a kind of neural network with a directed cycle. The main idea of RNN is to make use of sequential information. For example, in order to predict next word in a sentence, one should not consider only current word but also the previous words. The cycle structure in RNN creates an internal state of the network which allows it to memorize the previous inputs.

The structure of RNN and its unfolded format are shown as Figure 4.24. In Figure 4.24, x_t and o_t are the input and output at time step *t*, s_t is the hidden state at time step *t*, *W*, *V* and *U* are the learnable filters. They are calculated as follows, where *f*() is an activation function.

$$s_t = f(Ux_t + Ws_{t-1}) \tag{80}$$

$$o_t = \text{softmax}(Vs_t) \tag{81}$$

The hidden state s_t is the core of RNN. It can be seen as the memory of the network. It captures the information about all of previous inputs. The output is calculated solely based on this memory.

There are many extensions of RNN. One of them is the Bidirectional RNN (BRNN). BRNN is proposed to combine both past and future information. It is very suitable for

the off-line customer pose estimation from surveillance video.

BRNN can be seen as two RNNs stacked on top of each other, which is shown as Figure 4.25. The output is then computed based on the hidden state of both RNNs. In Figure 4.25, the *sf* and *sb* mean the hidden states of forward RNN and backward RNN respectively. Each variable is calculated as equation (82), (83) and (84), where U_f, W_f, U_b, W_b and V are the learnable filters, $[sf_t, sb_t]$ denotes the concatenation of sf_t and sb_t.

$$sf_t = f\left(U_f x_t + W_f sf_{t-1}\right) \tag{82}$$

$$sb_t = f\left(U_b x_t + W_b sb_{t+1}\right) \tag{83}$$

$$y_t = \text{softmax}(V[sf_t, sb_t]) \tag{84}$$

For the customer pose estimation, the input *score* is the output of 3 convolutional layers which introduce in Chapter 4.4.3, the output *y* is the final joint heatmaps.

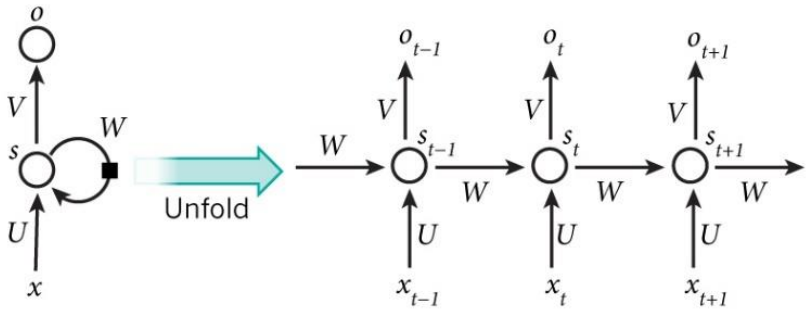


Figure 4.24 The structure of RNN and its unfolded format. This image is directly sourced from [113]

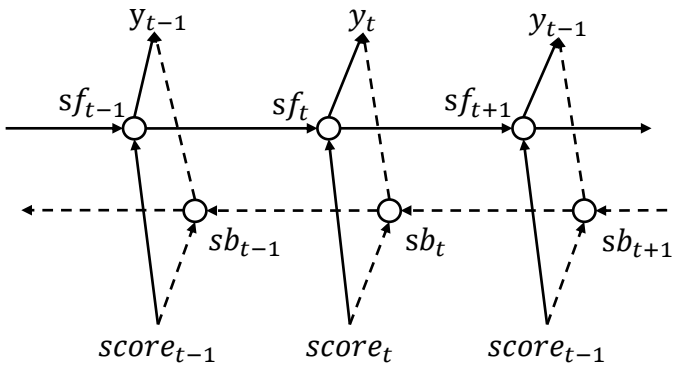


Figure 4.25 The structure of BRNN. Note that BRNN cannot be drawn as the folded format

The BRNN can be seen as a temporal message passing. The heatmap at frame t $score_t$ receives the forward message $W_f sf_{t-1}$ from frame $t-1$ and the backward message $W_b sb_{t+1}$ from frame $t-1$. Combining the both forward and backward messages, the BRNN produces the refined heatmap. Figure 4.26 shows an example process of this temporal message passing. The white points in the heatmaps are the predicted joint positions.

This BRNN is trained by using the method in [114]. That means the parameters of the parts before BRNN are fixed and only the parameters in BRNN would be updated. Theoretically, the end-to-end training should be better. However, the end-to-end training of BRNN needs to occupy extremely large memories and much time. For example, if the depth of BRNN is set as 5 (5 forward states and 5 backward states), the training needs 10 times of memories as that in the training of single image. Considering the FCN-8s is already quite heavy (about 10 times parameters than that of GoogLeNet and 5 times than that of ResNet), the end-to-end training of BRNN is impractical in personal computer.

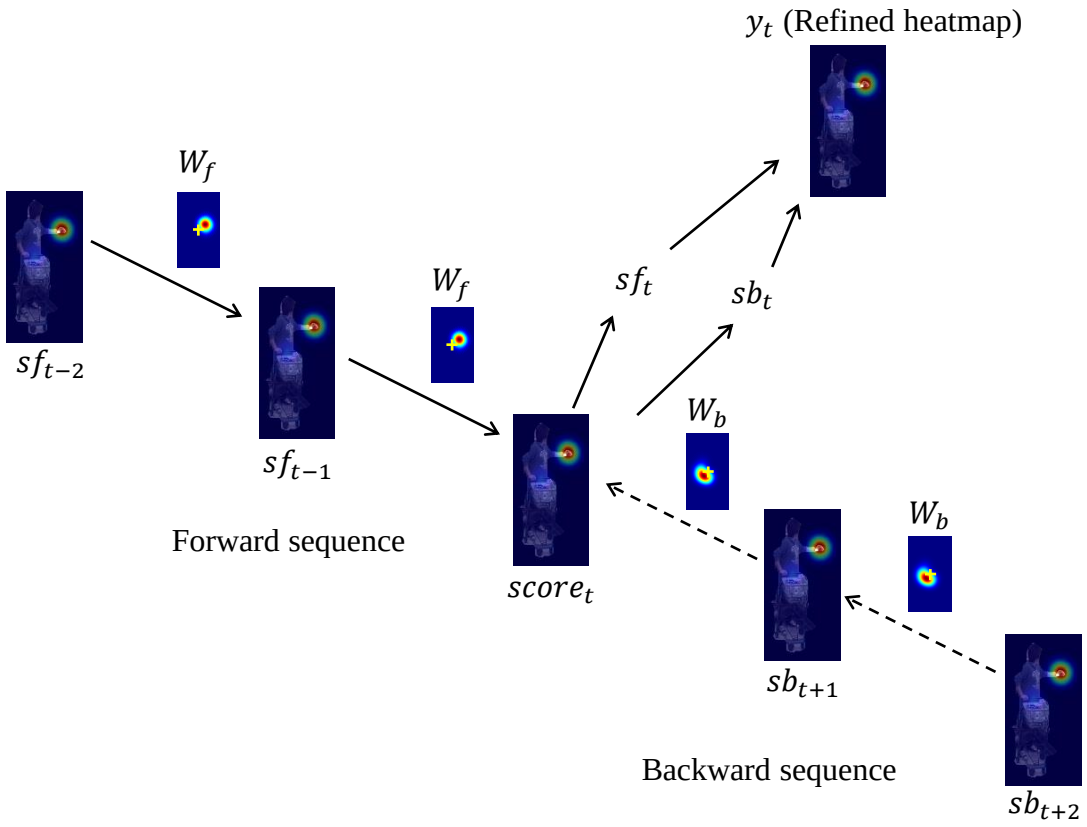


Figure 4.26 An example process of BRNN. It can be seen as a temporal message passing. The white points in the heatmaps are the predicted joint positions

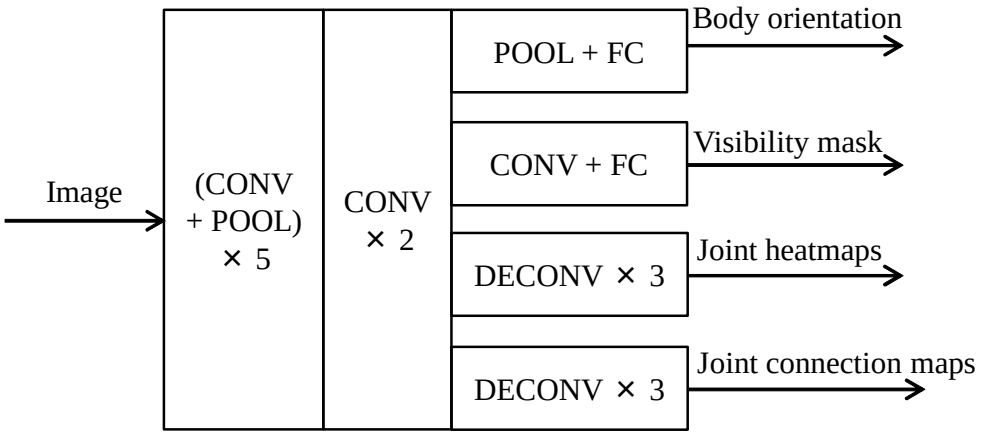
4.4.5. Pose Estimation Incorporating Visibility Mask

The visibility mask is not explicitly applied in the above modules. However, it implicitly runs through the whole model. In the ground truth labeling, the occluded joints are labeled as background. This labeling defines the learning target of visible and occluded joints. In the OMP layers, the occluded joints are not counted in the relative position histograms. This means that, for some specific body orientations (e.g. 90° or 270°), the corresponding occluded joints (e.g. the right body part or the left body part) will receive a weak orientational message and then produce a relative weak result heatmap. In the information combination part, the basket heatmap indicates the probability of occlusion and results in the 'cooling down' in the basket area. In the BRNN, the temporal changing between visibility and occlusion is also learned in the network. Therefore, the visibility mask is a latent thread from the input to the output. Note that these optimizations are established on the joint heatmaps. These methods are difficult applied in the structures which directly generate the joint positions [50] [51] [52].

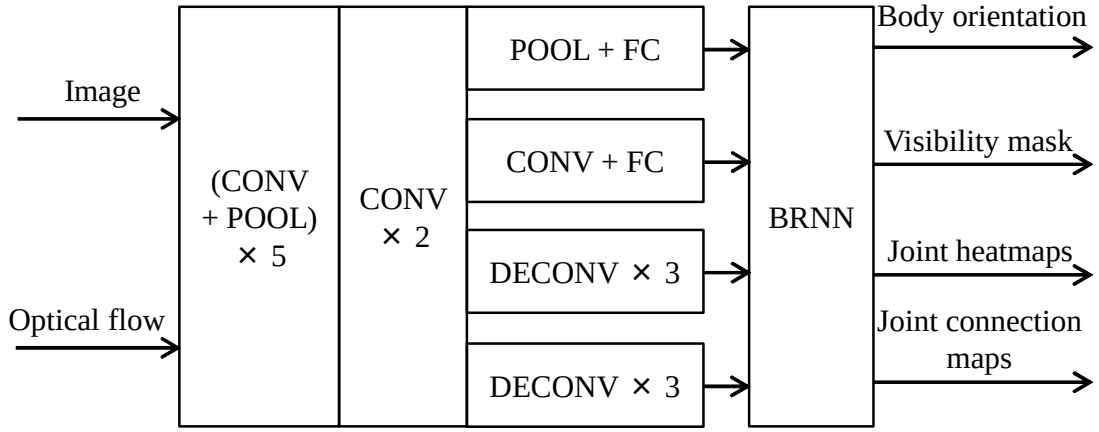
4.5. Multi-Task Neural Network for Pose Estimation

In order to improve and simplify the system further, this chapter propose a multi-task neural network for pose estimation. There are two variations of this network, which are shown as Figure 4.27. In Figure 4.27, "CONV" means convolutional layer, "POOL" means pooling layer, "FC" means fully connected layer, "DECONV" means deconvolutional layer. The Figure 4.27(a) is designed for pose estimation in single image. The network simultaneously generates the outputs of 4 task: body orientation, visibility mask, joint heatmaps and joint connection maps. These 4 tasks share a backbone of FCN, and then produce the output from respective specific output layers. The Figure 4.27(b) is a version for image sequence. Comparing to Figure 4.27(a), the differences are stacking the optical flow in the input end and adding a BRNN on top of the network.

Comparing with the IntePoseNet, in the multi-task neural network, the body orientation shifts to the output end. Besides, the visibility mask is explicitly given rather than calculated from joint heatmaps. Furthermore, the network incorporate novel joint



(a)



(b)

Figure 4.27 The structure of multi-task neural network for pose estimation in (a) single image, (b) image sequence

connection maps to model the joint connection rather than applying DPM.

The body orientation, visibility mask and joint heatmaps have already been introduced in this and previous chapter. Here will introduce the joint connection maps. Similar with the joint heatmap, one joint connection map is a 2-D matrix, which represents the image area. In joint connection map, only the area between two visible joints has positive values. The values of other area are all-zero. The area between two joints are defined as Figure 4.28. In Figure 4.28, there are 3 kinds of condition about the joint pair. The first one is that the both joints are visible. In this condition, the joint connection area is a rectangle determined by the parameters *Rh* and *Rw*. The second

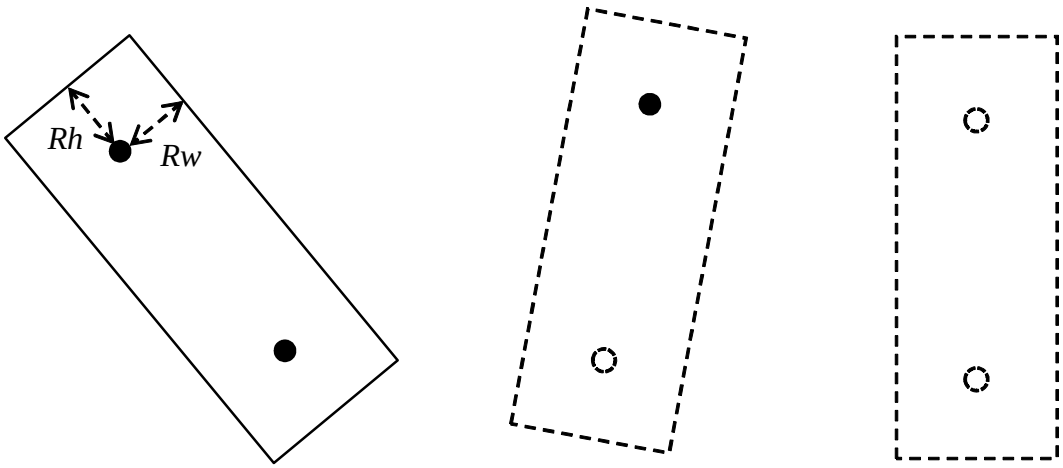


Figure 4.28 The strategy to generate the joint connection mask. (a) Both joints are visible; (b) Only one joint is visible; (c) Both joints are occluded

condition is that only one joint is visible. Because the other joint is occlude, there is no connection between these two joints. The connection map will be all-zero. Similarly, when both joints are occluded, the connection map is also all-zero. This strategy ensures that the connection map is based on the appearance of image.

In the joint connections areas, the values are set as the values of gradient magnitude image within those area. As shown in Figure 4.29, for an input image (a), firstly the gradient magnitude image is calculated as (b). Then the joint connections areas are applied as masks to crop different areas from the gradient magnitude image and then construct the joint connection maps. As in Figure 4.29(c), because the left elbow is occluded, the connection map between left shoulder and left elbow is all-zero. The gradient magnitude image is calculated by the Sobel operator. Assuming I is the gray image converted from input image, the gradient magnitude G is calculated as below.

$$G_x = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \otimes I \tag{85}$$

$$G_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \otimes I \tag{86}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{87}$$

where G_x and G_y are the gradients in *x* and *y* directions, the operator $\otimes$ means 2-D convolution, the 3×3 matrices are the Sobel operators. Zero-padding is applied at the edges of image in order to keep the output size the same as input size in the convolutions. The gradient image ensures that the network learns joint connection based on the context of image. It can help the network to understand the appearance of joint connection.

The Euclidean loss is applied to evaluate the prediction of joint connection map. Assume c_i and c_i' are the values in the prediction *c* and ground truth *c'*. The loss L_c is calculated as below.

$$L_c = \frac{1}{2N}\sum_{i=1}^{N}(c_i - c_i')^2 \tag{88}$$

where *N* is the number of pixels in the joint connection map. The gradient of loss is shown as below.

$$e_c = c - c' \tag{89}$$

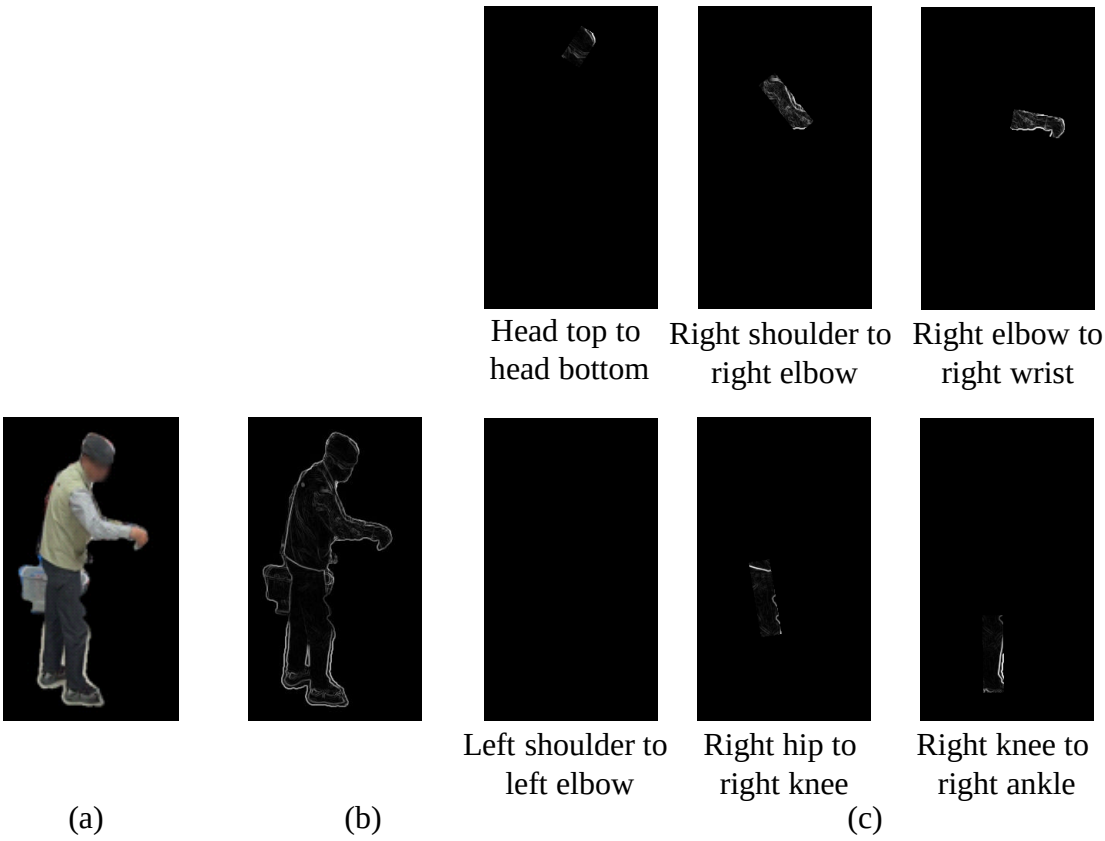


Figure 4.29 Example ground truth of joint connection maps. (a) The input image; (b) The gradient image; (c) A part of the joint connection maps

On the other sides, because the calculation of joint heatmaps shares the same structure of joint connection heatmaps, the joint heatmaps also changes the representation from the element-wise label in the IntePoseNet. Figure 4.30 shows the new representation of joint heatmaps. Centered on each visible joint position, a normalized Gaussian distribution with small variance constructs the joint heatmap. For the occluded joint, the joint heatmap is all-zero.

Therefore, the joint heatmaps can also apply the Euclidean loss. Similarly, the loss of joint heatmap L_h is shown as below.

$$L_h = \frac{1}{2N}\sum_{i=1}^{N}(h_i - h_i')^2 \tag{90}$$

where h_i and h_i' are the predicted and ground truth value of the joint heatmap, *N* is the number of pixels in the heatmap.

Again, the loss of orientation L_o and the loss of visibility mask L_v are calculated as following.

$$L_o = -\sum_{i=1}^{8} o_i \log o_i' \tag{91}$$

$$L_v = -\sum_{i=1}^{28} v_i \log v_i' \tag{92}$$

where o_i and o_i' are the values of predicted and ground truth of body orientation, v_i and v_i' are the values of predicted and ground truth of visibility mask.

Therefore, the overall loss of the multi-task neural network is shown as below.

$$L = \alpha_1 L_h + \alpha_2 L_c + \alpha_3 L_o + \alpha_4 L_v \tag{93}$$

where α_1, α_2, α_3 and α_4 are the weights of each task. Empirically, $\alpha_1 = 1000$, $\alpha_2 = 0.1$, $\alpha_2 = 100$ and $\alpha_4 = 1$. The principal is to let the values of these losses have similar magnitude level.

The multi-task learning works because of several reasons. Firstly, the multi-task essentially augment the data. For any task, there are data-dependent noises more or less. One wants to exclude these noises as more as possible. Because different tasks have different noise patterns, a network that learns multiple tasks simultaneously can learn a more general representation. In other word, the multi-task learning reduces the risk of overfitting comparing with the single task learning. Secondly, the highly related tasks

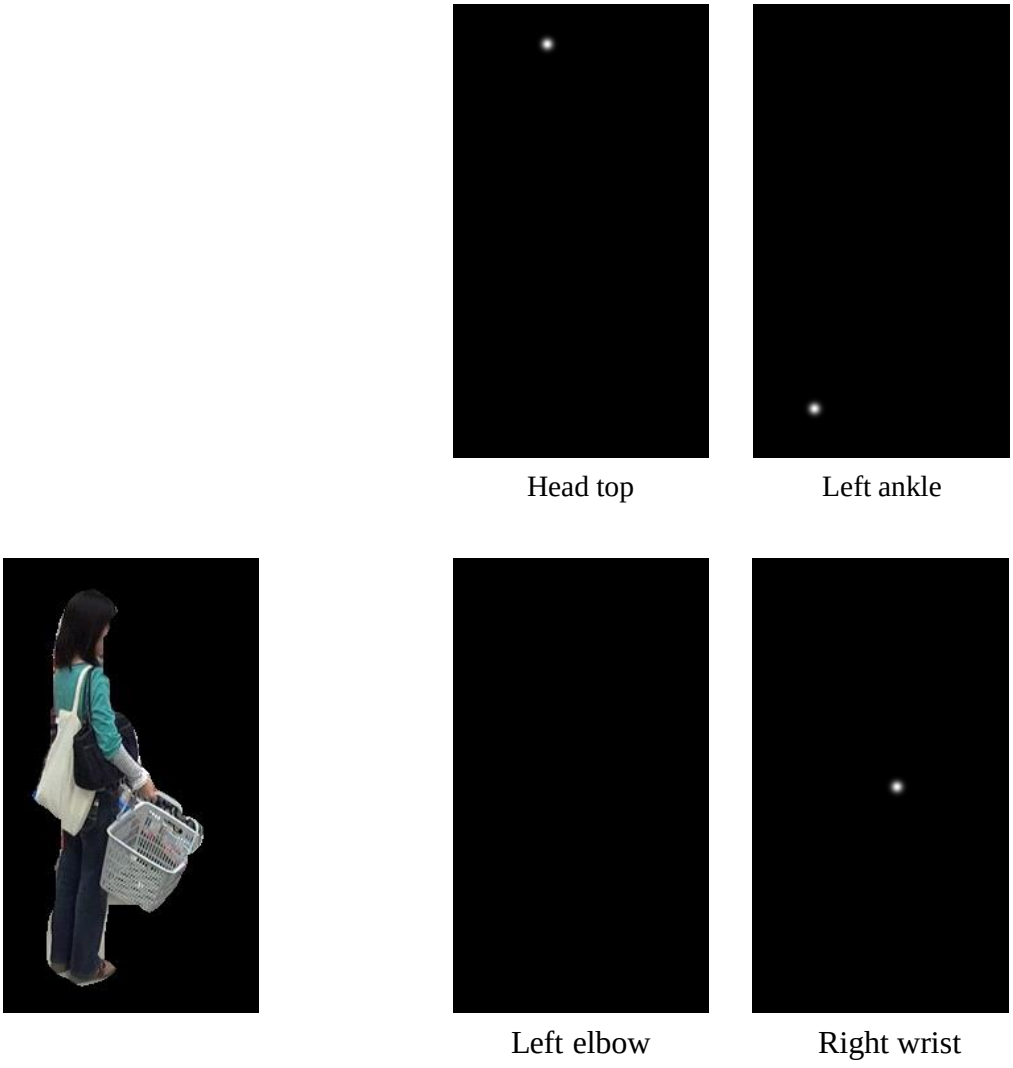


Figure 4.30 Example ground truth of joint heatmap in Gaussian distribution form. (a) The input image; (c) A part of the joint heatmaps

can indicate the most relevant features, while the single task system may learn both relevant and irrelevant features. The multi-task gives more evidences about what kinds of features are essentially important. It means the multi-task can help the system focusing on the relevant features. Thirdly, the multiple tasks improve the feature learning for each other. In the single task learning, some features may be difficult to learn but easy in another task, because of the different learning processes. In the multi-task learning, these difficult features can be easily leaked from other tasks. Therefore, the multi-task learning is supposed to achieve a higher performance than single task learning. However, the

premise is that the tasks are highly related, otherwise the multi-task learning may not produce better results or may produce even worst results. In the IntePoseNet, the body orientation, visibility mask and joint connections are already proven to be helpful to pose estimation. Therefore, the 4-task neural work is expected to outperform the IntePoseNet. In the Chapter 4.6, a series of experiments prove this expectation.

4.6. Experiments

This chapter tests the above methods on several datasets comparing to the state-of-arts methods. In this chapter, the experiments are conducted on three datasets: the customer pose dataset, Leeds Sports Pose (LSP) dataset [115] and Joint-annotated Human Motion Data Base (JHMDB) [116]

4.6.1. Customer Pose Dataset

The images of customer pose dataset are shot from real surveillance camera of retail stores, where are from the same source as the customer orientation dataset in Chapter 3.5. The surveillance videos are selected from the security camera near the merchandise shelf and then are cut into clips when a single customer stops on the front of the shelf. Each video clips are convert into one image sequence. The selected videos are from 4 surveillance cameras, which are installed in 4 different retail stores respectively. The occlusion between customers is not included in the dataset.

For each image, the customer is labeled the positions of 14 joints: head top, head bottom, left/right shoulders, left/right elbows, left/right wrists, left/right hips, left/right knees and left/right ankles. As mentioned in Chapter 2, the labeling of joint follows the Person-Centric annotation. The customer pose images are also labeled body orientation. In the training data, the body orientation with clear class is given a strong label, while the body orientation with ambiguous class is given a pair of weak labels.

As shown in Table 1, in the customer pose dataset, there are 58 sequences/9,703 images for training, and 21 sequences/3,250 images for testing. The sequence length is from 60 to 360 frames. The resolution of original images is 720 by 480. The body height of customer is around 300-400 pixel, the head size is around 40-50 pixel. The frame rate

of videos is 29 frames per second. As the explanations in Chapter 2, all of the images used in experiments are after background substation. The figures shown with background are just for clarity. After the background substation, the customer images are normalized to average size 200×366. All of the images are augmented by left-right flipping.

For pose estimation, Percentage of Correct Keypoints (PCK) is a widely used evaluation metric [41] [44] [51] [53]. The correct keypoint is defined as the predicted joint whose distance from true joint is less than a given threshold. However, in this dissertation, the system not only output the joint positions but also the visibility masks. In order to evaluate the result of both joint position visibility mask, this dissertation proposes the Visibility sensitive Percentage of Correct Keypoints (VPCK), which is inspired by the Occlusion sensitive Percentage of Correctly detected Parts (OPCP) in [62]. The VPCK is defined as below, where N_v and N_{inv} are the number of visible and invisible joints respectively, PCK_v is the PCK of visible joints, $Accuracy_{inv}$ is the detection rate of invisible joint.

$$VPCK = \frac{PCK_v \times N_v + Accuracy_{inv} \times N_{inv}}{N_v + N_{inv}} \tag{94}$$

A series of comparison experiments are conducts as below. If the body orientation is used as input, it is obtained from the ResNet based SSL method in Chapter 3.3.2. The optical flow is calculated by using FlowNet [112].

(a) Heatmap by HOG + Message Passing (Conventional DPM [41]).

(b) Heatmap by HOG + Message Passing + Orientation. Eight models are trained for each body orientation. It means each model is trained on the data with same body orientation by using the method of [41]. In the testing, the testing data are firstly detected the body orientation, and then chooses one corresponding model to estimate the pose. The structure is introduced in Chapter 4.2.2.

Table 4.2 The details of customer orientation dataset

	Training data	Testing data	Total
Sequence number	58	21	79
Image number	9,703	3,250	12,953

(c) ResNet. The network is based on modified ResNet-50 [23] to directly output the joint positions and visibility mask. The structure is introduced in Chapter 4.3.1.

(d) ResNet + Orientation. The network is based on modified ResNet-50 [23] to directly output the joint positions and visibility mask. The last FC layer of network receives two inputs: the output of previous layer and the body orientation vector. The structure is introduced in Chapter 4.3.1.

(e) Heatmap by FCN + Message Passing. The structure is introduced in Chapter 4.3.2.

(f) IntePoseNet.

(g) Multi-task neural network for single image.

(h) Multi-task neural network for image sequence.

(i) Multi-person pose estimation using part affinity fields [71]. This is the state-of-art multi-person pose estimation method. This chapter applies this method on the single customer pose dataset to compare with the proposed methods.

For experiments (c) ~ (f), the networks are trained for 100 epochs. The learning rate starts from 10^{-6} and decreases by half every 10 epochs. For experiments (g) and (h), the training epoch is 200. The learning rate starts from 10^{-5} and decreases by half every 10 epochs. The depth of BRNN in experiments (f) and (h) is 6 in both directions.

The overall result and the results of each joint are shown as Figure 4.31. The threshold means the percentage of body height. MP means the Message Passing. The values of VPCK when the threshold is 9% of body height are list in Table 4.3. Note that in the experiment (a) and (b), the DPM does not generate the visibility mask. Thus for the experiment (a) and (b), the values in Figure 4.31 and Table 4.3 are the traditional PCK values. The comparison examples are shown in Figure 4.32.

The results show several interesting facts. (1) Multitask neural network for image sequence achieves the highest performance. (2) The methods using hand-craft features much underperform the deep learning methods. (3) The orientation information generally can improve the VPCK (or PCK) dramatically. However, for ResNet, the improvement of orientation is relative small. In Table 4.3, for wrist estimation, the result of ResNet + Orientation even worse than the result of ResNet only. The reason is that the ResNet only produce one prediction per image. The orientation information can

hardly correct the prediction without any other candidates. (4) The body orientation information can solve the problem of left-right similarity, such as the row 1 and row 4 in Figure 4.32. The left (red/purple) and right (blue/cyan) limbs are corrected from reverse side. (5) The multi-person method [71] performs well for the visible joints. However, it is difficult to detect the occluded joint and thus results in the false positives. The reason is that it is a kind of bottom-up method. This method firstly searches the joints aggressively and then groups the joints to construct the human pose. (6) The visibility mask can help the system to detect the occlusion well. Thus for the occluded joints, the system does not need to output their positions by guessing. (7) Considering the VPCKs (or PCKs) cross different body joints, the performances of less variate joints are higher and vice versa.

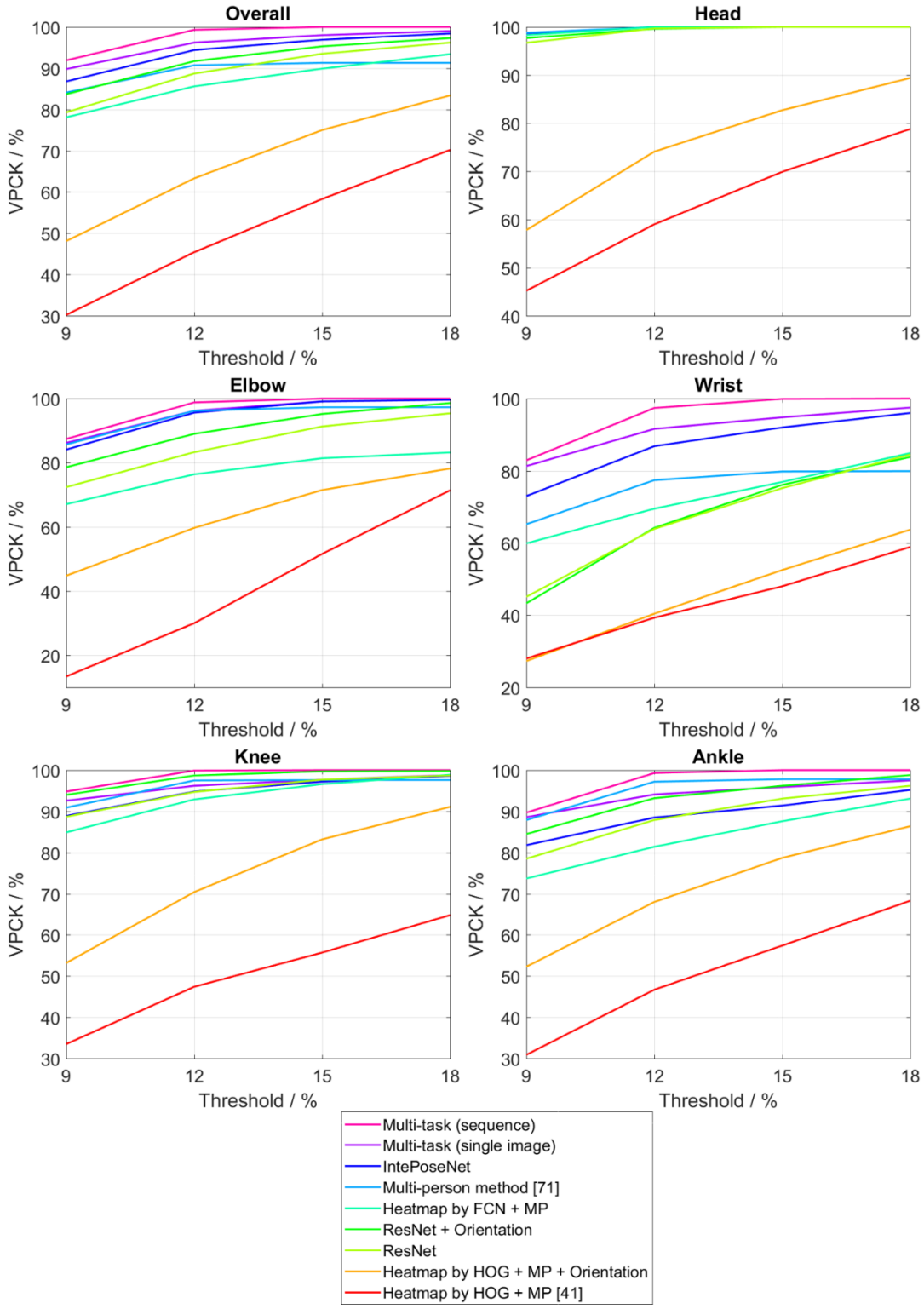


Figure 4.31 Examples results of customer pose estimation. For the HOG based methods, the curves represent the traditional PCK

Table 4.3 The VPCKs of comparison experiments on customer dataset

	Head	Elbow	Wrist	Knee	Ankle	Overall
Heatmap by HOG + MP [41]*	45.2	13.4	28	33.5	30.9	30.2
Heatmap by HOG + MP + Orientation *	57.8	44.8	27.3	53.2	52.3	48.1
ResNet	96.7	72.4	45.1	88.7	78.5	79.3
ResNet + Orientation	97.7	78.6	43.3	94.0	84.5	83.7
Heatmap by FCN + MP	98.3	67.1	59.9	84.9	73.7	78.1
IntePoseNet	98.7	84.1	73.0	88.9	81.8	86.8
Multi-task (single image)	98.7	86.2	81.3	92.6	88.6	89.8
Multi-task (sequence)	98.8	87.4	82.9	94.8	89.7	91.9
Multi-person method [71]	98.8	85.7	65.2	90.9	87.9	84.1

MP = Messing Passingw

* The values in this row are the normal PCKs

Figure 4.32 Examples results of customer pose estimation. (a) Heatmap by HOG + MP [41]; (b) Heatmap by HOG + MP + Orientation; (c) ResNet; (d) ResNet + Orientation; (e) Heatmap by FCN + MP; (f) IntePoseNet; (g) Multi-task (single image); (h) Multi-task (sequence); (i) Multi-person method [71]

Then the ablation studies are conducted to test the effect of each part of the multi-task neural network. There are 3 different multi-task structures are tested as shown in Figure 4.33. In Figure 4.33(I), the joint heatmap task and joint connection map task share totally same structure in network. They are output from a same tensor, where the first 14 channels are the joint heatmaps and the other are the joint connection maps. In Figure 4.33(II), the joint connection map task is removed. In Figure 4.33(III), the visibility mask task is furtherly removed. Thus the visibility mask in Figure 4.33(III) is calculated as IntePoseNet. The testing results of these structure are shown as Table 4.4.

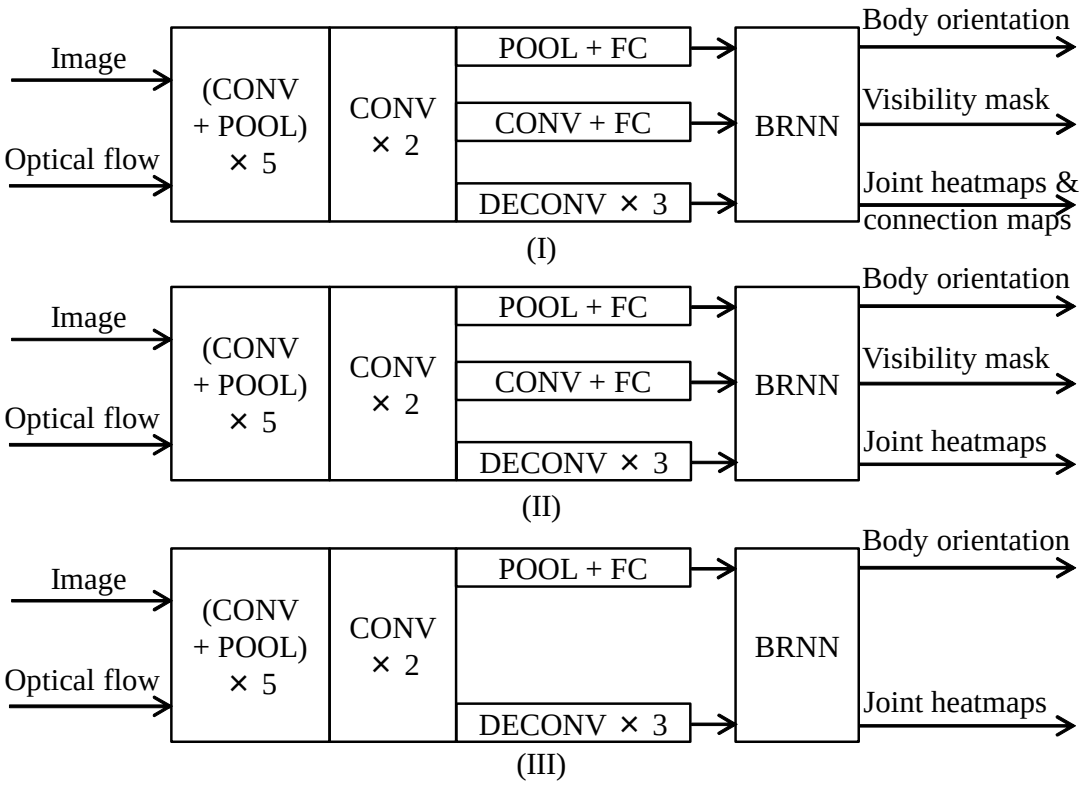


Figure 4.33 Ablation studies of the multi-task neural network. (I) Joint heatmaps and connection maps are output in a same tensor. (II) Joint connection map task is removed. (III) The visibility mask task is furtherly removed

Table 4.4 The VPCKs of different multi-task structures

	Head	Elbow	Wrist	Knee	Ankle	Overall
Strucutre I	71.6	36.8	33.9	61.8	42.5	52.0
Strucutre I	97.7	83.2	71.8	88.0	80.1	85.5
Strucutre III	92.8	79.1	66.5	81.7	77.1	80.5

In Table 4.4, the training of structure I actually fails. It is difficult to balance the weights of joint heatmap task and joint connection map task. The unbalanced errors in the same structure result in the low accuracy. In structure II, the VPCK is slightly lower than IntePoseNet. In fact, the accuracy of visibility mask is higher than IntePoseNet by checking the detail results. However, lacking of the joint connection maps, the pose accuracy is difficult to improve. In structure III, the VPCK becomes further lower. The main reason is that the inaccurate visibility mask increases. This result shows the

importance of visibility mask task.

.

4.6.2. LSP Dataset

The LSP dataset [115] is widely used benchmark for human pose estimation. In fact, customer pose dataset follows the same labeling of the LSP datasets: the same 14 joints with Person-Centric annotation. The original LSP dataset contains 2000 images of sportspersons, where 1,000 for training and 1,000 for testing. Besides, the Leeds Sports Pose Extended Training (LSPET) [117] dataset is also widely used as the extended training data with LSP. The LSPET contains 10000 images for training and follows the same labeling protocol as LSP.

However, there is no orientation label in LSP and LSPET dataset. Actually, these two datasets include many complex sports images, which cannot be simply given an orientation. Figure 4.34 shows some examples that are difficult to label a simple body orientation. In order to solve this problem, the body orientation is firstly manually given to for the images which show as Figure 2.3(b). Then the other images with complex body orientation are grouped into label “other orientation”. The image numbers of orientation labels on LSP and LSPET dataset are listed in Table 4.5. Besides, all of the images are augmented by left-right flipping, including the “other orientation” images.

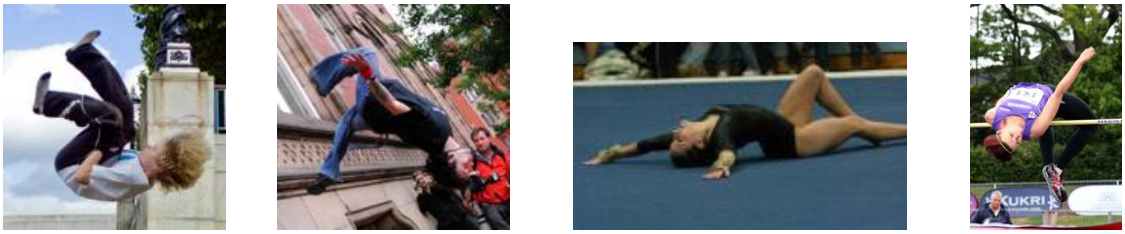
Based on the 9 orientations labeling, experiments are conducted on the below structures.

(a) Heatmap by FCN + MP

(b) Heatmap by FCN + MP + Orientation

(c) Multi-task neural network on single image.

Because LSP and LSPET do not contain image sequence, the IntePoseNet and multi-task neural network on image sequence are not applied in the experiments. In the experiments (b), the body orientation is detected by the ResNet based SSL in Chapter3.3.2. If the body orientation is detected as one of the 8 orientations as Figure 2.3(b), the proposed Orientational Message Passing layer will be applied for pose estimation. If the body orientation is detected as “other orientation”, the conventional Message Passing layer will be applied.



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

Figure 4.34 Examples in LSP and LSPET with complex body orientations. (a) and (b) are the images from LSP, (b) and (c) are the images from LSPET

Table 4.5 The image number of orientation labels on LSP and LSPET dataset

	Number of training images			Number of testing images		
	8 orientations	Other orientation	Overall	8 orientations	Other orientation	Overall
LSP	889	111	1,000	873	137	1,000
LSPET	4,552	5,448	10,000	-		

The results are list in Table 4.6 comparing with the state-of-arts. The threshold of PCK is defined as the 0.2 of the body height. In order to compare with other methods, the evaluation metric is PCK rather than VPCK. In Table 4.6, although overall PCKs of the proposed methods are lower than the state-of-arts, the proposed methods achieve higher performances in the 8 orientations. The orientation information bring significant improvements on the performance. Note that the PCKs of "other orientation" images are far lower than the PCKs of 8 orientations. These images cannot be benefited from orientational message layers because orientational message layers are not applied for them. Moreover, the "other orientation" group includes mixed complicated poses, it is difficult to be benefited from the multi-task neural network. The example results are shown in Figure 4.35.

Table 4.6 The PCK (%) of pose estimation on LSP + LSPET dataset

(a) PCKs comparing state-of-arts

Method	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Overall
Yang et al. [57]	90.6	78.1	73.8	68.8	74.8	69.9	58.9	73.6
Rafi et al. [118]	95.8	86.2	79.3	75	86.6	83.8	79.8	83.8
Yu et al. [119]	87.2	88.2	82.4	76.3	91.4	85.8	78.7	**84.3**
Heatmap by FCN + MP	84.1	82.3	61.3	53.5	91.7	61.9	42.4	68.2
Heatmap by FCN + MP + Orientation	90.9	86.3	75.8	67.8	92.5	82.6	72.1	81.2
Multi-task (single image)	91.1	86.9	76.9	68.1	92.9	83.1	73.0	82.0

(b) Overall PCKs of proposed methods in different orientation

Method	Orientation									
	1	2	3	4	5	6	7	8	other	Avg of 8 orientations
Heatmap by FCN + MP	76.6	77.5	69.3	69.3	55.1	63.8	67.0	78.9	32.3	69.7
Heatmap by FCN + MP + Orientation	90.1	86.5	82.6	82.0	78.8	81.5	82.5	90.2	32.1	85.5
Multi-task (single image)	90.9	87.7	83.5	82.8	79.8	83.2	83.7	90.9	39.9	**86.7**

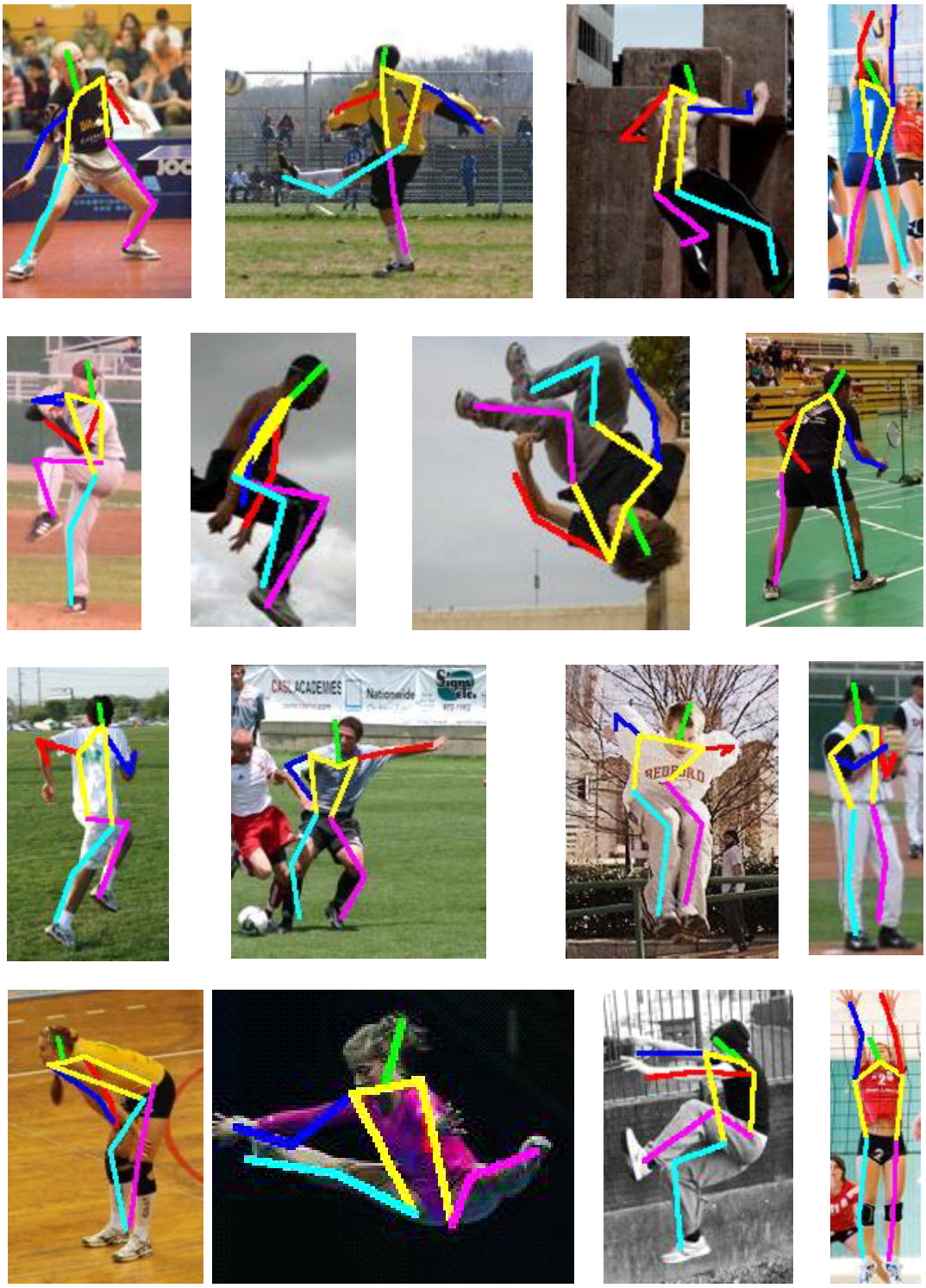


Figure 4.35 The example results of pose estimation on LSP dataset using Heatmap by FCN + MP + Orientation structure

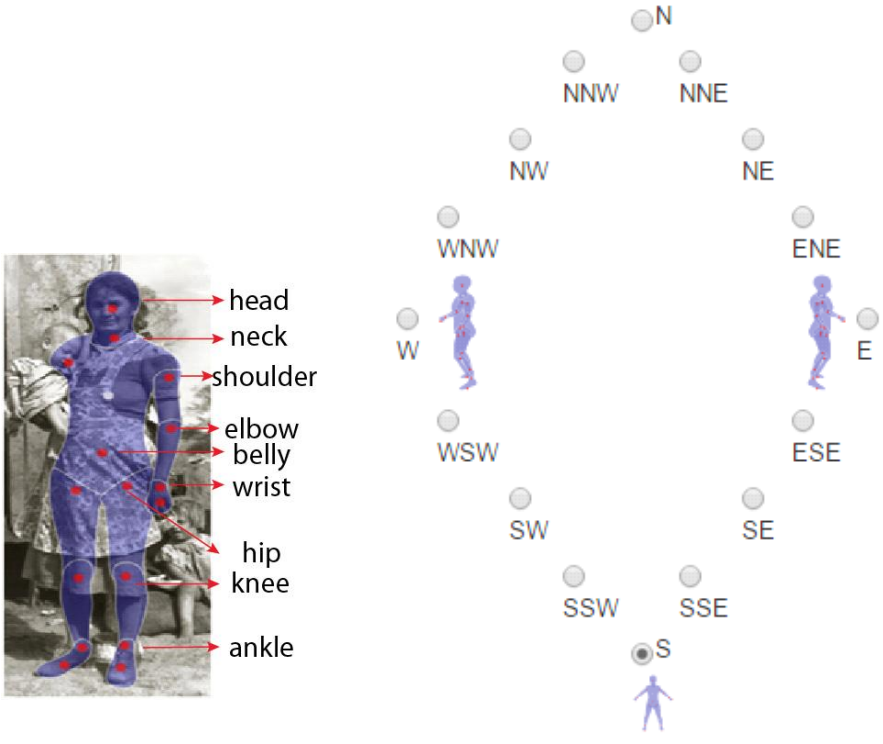
4.6.3. JHMDB Dataset

The JHMDB [90] dataset contains 928 videos and 21 action classes. The dataset not only annotates joint positions but also body orientations. The joint and body orientation labels are shown in Figure 4.36. The dataset also provides a subset sub-JHMDB, where the whole human body is inside the image, thus all joints are annotated. The subset contains 316 clips with 12 action categories. The experiments are conducted on the sub-JHMDB dataset. Besides, the sub-JHMDB dataset provides 3 splits of training data and testing data. Thus the experiments report the average performances. All of the images are augmented by left-right flipping.

The experiments are conducted on structures as below. In experiment (a), the body orientation is obtained from the ResNet based SSL method in Chapter 3.3.2. In experiments (a) and (c), the optical flow is calculated by using FlowNet [112].

(a) IntePoseNet (without basket heatmap).

(b) Multi-task neural network for single image.



(a) (b)

Figure 4.36 The example labels of (a) joint and (b) body orientation in JHMDB dataset

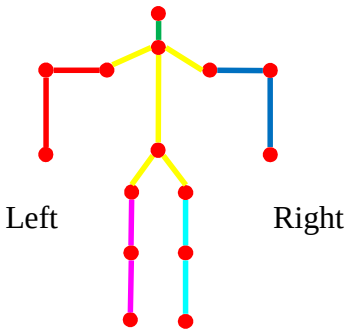


Figure 4.37 The joint tree structure used for message passing calculation in sub-JHMDB dataset

(c) Multi-task neural network for image sequence.

Because the dataset annotates 15 joints rather than 14 joints as in the customer dataset, an alternative joint tree structure is adopted to calculate the message passing. The joint tree structure is shown in Figure 4.37. The 'left' and 'right' in Figure 4.37 mean the left side and right side of person.

The experimental results are shown in Table 4.7 comparing with the state-of-arts. In order to compare with other methods, the evaluation metric is PCK rather than VPCK. The threshold of PCK is 0.2 of the body height. In Table 4.7, the multi-task neural

Table 4.7 The PCK (%) of pose estimation on sub-JHMDB

Method	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Overall
Nie et al. [120]	80.3	63.5	32.5	21.6	76.3	62.7	53.1	55.7
Iqbal et al. [121]	90.3	76.9	59.3	55.0	85.9	76.4	73.0	73.8
Song et al. [122]	97.1	95.7	87.5	81.6	98.0	92.7	89.8	92.1
IntePoseNet	99.1	90.9	85.8	81.4	98.2	93.6	89.7	91.2
Multi-task (single image)	99.2	91.0	86.9	82.3	98.2	93.8	89.6	91.8
Multi-task (sequence)	99.2	92.2	88.2	83.4	98.3	93.8	89.8	**92.3**

network for image sequence achieves the best performance. The both versions of multi-task neural networks outperform the InterPoseNet. This proves the advantage of the multi-task learning. The example results are shown in Figure 4.38.

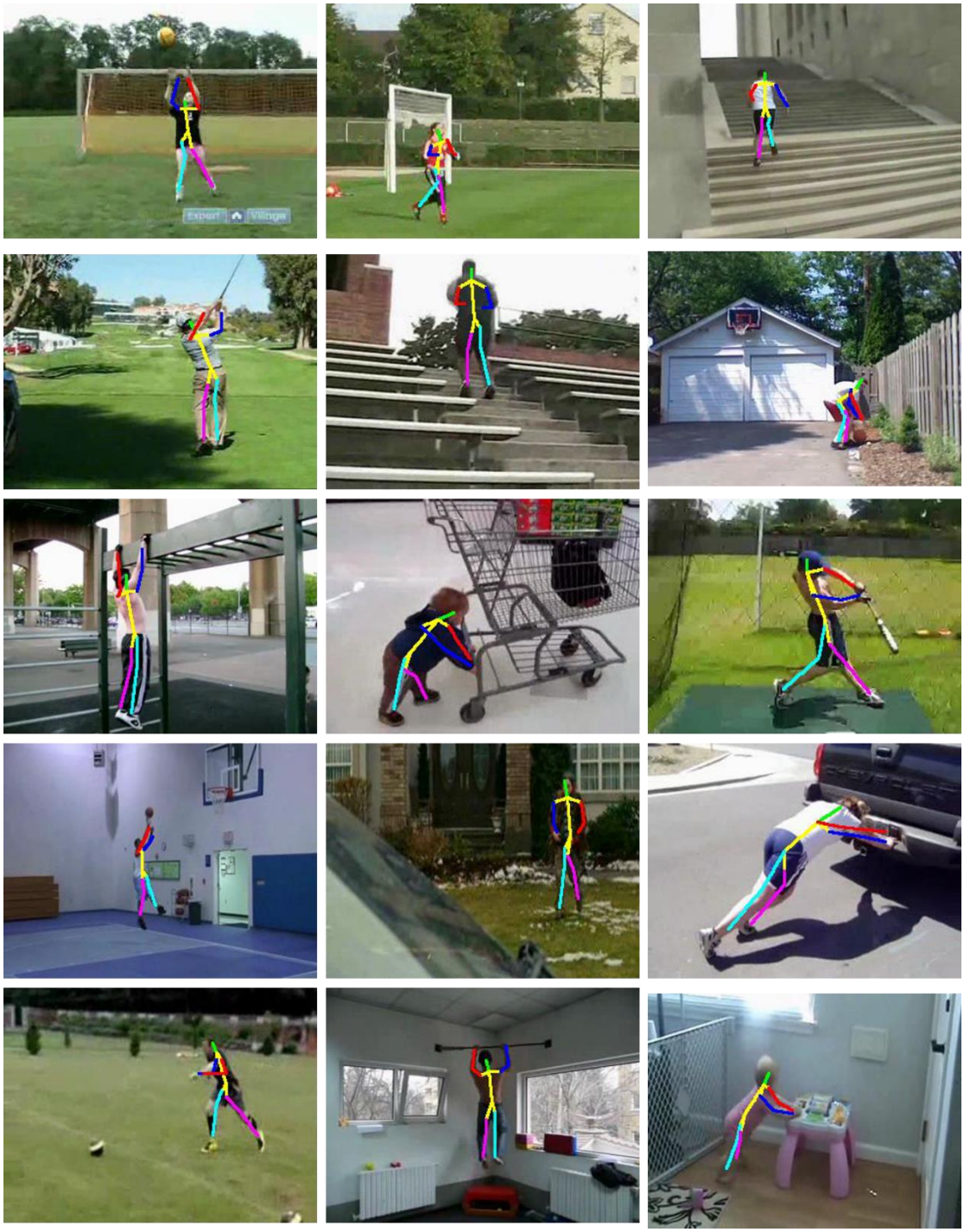


Figure 4.38 The example results of pose estimation on sub-JHMDB dataset using multi-task neural network for image sequence

Chapter 5. Customer Arm Action Classification

5.1. Overview

The customer arm action classification is especially important for retailers, because the arm actions are directly related with highest interest levels. This chapter proposes to classify 3 types of customer arm actions in real retail store environment: *touching*, *picking and returning* to shelf and *picking and putting* into basket. The definition of these 3 actions are already given in Chapter 2.

Physically, the most direct approach to classify these 3 actions is to detect and track the picked items. However, in the surveillance video, the picked items are usually too small to detect and are easily occluded by customer body. Therefore, this chapter proposes a novel *Combined Hand Feature* (CHF), which includes hand trajectory, tracking status and the relative position between hand and shopping basket, to describe the arm action in each frame. After extracting the CHF every frame, a Dynamic Bayesian Network (DBN) can be adopted to classify these CHFs into different arm actions.

5.2. Customer Stretching Arm Detection

Firstly, in order to start the customer arm action classification, the stretching arm needs to be detected. If the customer does not stretch his or her arm, the arm action classification would not be triggered. If the stretching arm is detected, the system starts to extract the CHF and classify them into different arm actions.

Based on the head orientation detection result, one can predict which side the arm will stretch (left or right side in the image). Without losing generality, all of the algorithms in the rest of this chapter assume that the customers always stretch their arms to the right side in the image.

In the previous chapter, the customer pose is obtained from the multi-task neural network. Therefore, the pose information can be used for the stretching arm detection. When the below condition is satisfied, the arm would be treated as stretching.

$$l_{arm} = \alpha_{arm} h_{body} \tag{95}$$

where l_{arm} is the stretching length of arm, h_{body} is the body height and α_{arm} is a

threshold. The body height is defined as the distance between the highest joint and the lowest joint. The stretching length of arm is defined as below, where x_{lw}, x_{rw}, x_{ls}, x_{rs}, x_{lh} and x_{rh} are the *x* coordinates of left wrist, right wrist, left shoulder, right shoulder, left hip and right hip respectively, x_{center} is the *x* coordinate of the body center. Empirically, the α_{arm} is set as 0.3.

$$l_{arm} = \max(|x_{lw} - x_{center}|, |x_{rw} - x_{center}|) \tag{96}$$

$$x_{center} = (x_{ls} + x_{rs} + x_{lh} + x_{rh})/4 \tag{97}$$

5.3. Combined Hand Feature Extraction

After detecting the stretching arm, the next step is to extract the CHF. Firstly, in order to understand the arm action of customer, the hand area is tracked by using particle filter. The hand area is defined as the largest square inside the silhouette centered at the wrist position from the pose information, which is shown as Figure 5.2. Therefore, the hand area can be defined as below.

$$dh_t = \{x_t, y_t, size\} \tag{98}$$

where x_t^d and y_t^d is the position of wrist point at time *t*, $size$ is the size first detected hand area. The color histogram of the hand area can be calculated as below.

$$H_t = h(dh_t) \tag{99}$$

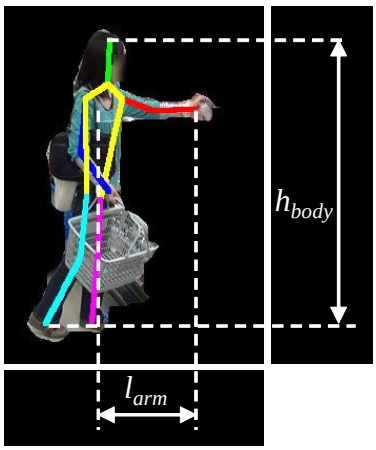


Figure 5.1 The measurements of the stretching length of arm and the body height based on the pose estimation result

where the function $h(dh_t)$ means the operation of calculating the color histogram of the area dh_t.

Once the hand area is found, the particle filter is initialized to track the hand area. The wrist position from pose estimation can be seen as the detection value in the tracking. *N* particles are randomly distributed around the wrist point as equation (100).

$$p_{t=0}^{i} = \{x_{t=0}^{i}, y_{t=0}^{i}, size_{t=0}^{i}\}, i = 1, \dots, N \tag{100}$$

The positions of particles are subject to a normal distribution centered at $dh_{t=0}$. Meanwhile, the particle filter maintains a template of color histogram HT_t every frame. Initially, the $HT_{t=0}$ is assigned as below.

$$HT_{t=0} = H_{t=0} \tag{101}$$

Then the particle filter starts to work by iterating the below 4 steps:

Weight: The weight of each particle is defined as below.

$$w_t^i = e^{-d^2(HP_t^i, HT_t)} \tag{102}$$

$$HP_t^i = h(p_t^i) \tag{103}$$

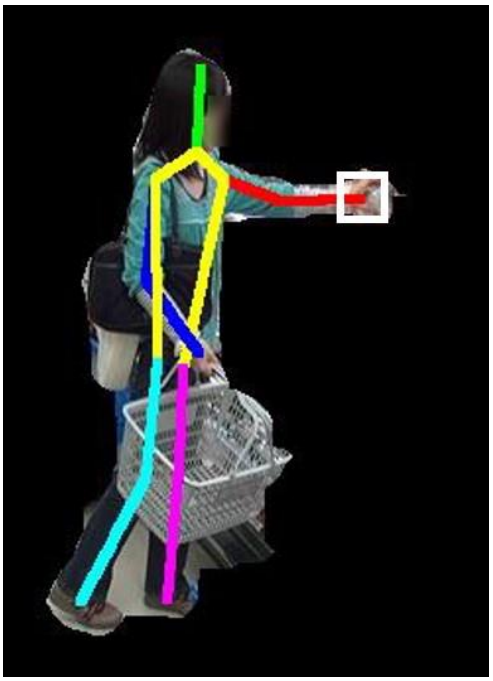


Figure 5.2 The hand area (white bounding box) centered at the wrist position

where *d* is the Bhattacharyya Distance and HP_t^i is the histogram of the area p_t^i. The Bhattacharyya Distance between two histograms H_1 and H_2 is defined as below.

$$d(H_1, H_2)=\sqrt{1-\frac{1}{\sqrt{\sum_I H_1(I)\sum_I H_2(I)}}\sum_I\sqrt{H_1(I)H_2(I)}}\,, I=1, \dots, M \tag{104}$$

where *M* is the total number of histogram bins. Then the weight is normalized to a probability forms whose sum is 1.

$$wn_t^i = w_t^i/\sum_i w_t^i \tag{105}$$

Estimation: The position of wrist in the next frame is estimated as below.

$$eh_{t+1} = \sum_i wn_t^i\{x_t^i, y_t^i, size_t^i\} \tag{106}$$

The template of color histogram is also updated as below.

$$HT_{t+1} = h(eh_{t+1}) \tag{107}$$

However, if the stretching arm cannot be detected at frame *t* + 1, H_{t+1} will be replaced by the *average hand histogram*. The *average hand histogram* will be discussed in the *stop condition* paragraph.

Resample: The particles are sorted in descending order by their weights. The last half particles are replaced by the new particles around the detected wrist position dh_t. If the stretching arm cannot be detected, the last half particles are replaced by the replicas of the first half particles.

Propagate: Each particle is propagated by a linear stochastic differential equation as equation (108), where w_t^i is a multivariate Gaussian random variable.

$$p_{t+1}^i = p_t^i + w_t^i, i = 1, \dots, N \tag{108}$$

In the resample step, unlike the traditional particle filter, the last half particles are replaced by the new particles around the detected wrist position, instead of replicas of first half particles. The position information of detected wrist can help correcting the tracking error. Figure 5.3 shows the distribution of particles. The red rectangle is the searched hand area and the center of the rectangle is the searched wrist point. In Figure 5.3(b), there are two groups of particles because the searched wrist point is far away from the detected wrist point. One group of particles is distributed around the searched

wrist position. Another group of particles is distributed around the detected wrist position. The particles around the detected wrist point can help 'pushing' the tracking result to the correct position.

Finally, the particle filter should be terminated when the arm action finishes. At this time, the arm is supposed to have withdrawn and no stretching arm can be detected. The hand area should be not seen or change a lot in appearance comparing that when the customer is picking an item. Based on these assumptions, the stop condition of particle filter is introduced as below.

Stop condition: As discussed above, the hand area is detected every frame. The system will accumulate the color histogram of every detected hand area and maintain an *average hand histogram*. Assume that the *average hand histogram* at time *t* is HA_t. When the stretching arm cannot be detected, the particle filter will switch the template histogram from the histogram of last detected arm to the *average hand histogram* as (109).

$$HT_t = HA_t \text{ when } dh_t \text{ does not exist} \tag{109}$$

It means the weight of each particle will be calculated by comparing with the *average hand histogram*. When the hand cannot be seen (of course also cannot be detected), the weights of particles are probably very similar. If the standard deviation of weights is smaller than a threshold Th_σ for a while, the particle filter will be terminated. Generally, the standard deviation of weights decreases with fluctuations and the searched hand area also deviates from the true hand area gradually. The threshold is supposed to equal the standard deviation of weights when the searched hand area just separates from the true hand area. By observing the dataset, the value of threshold is empirically decided as 10^{-5} and the particle filter will be terminated when the standard deviation of weights is continuously less than Th_σ for 10 frames.

In conclusion, there are 3 different statuses in the tracking process: *detected*, *tracked* and *lost*, which are shown as Table 5.1. The tracking status *TS* can be defined as (110).

$$TS \in \{detected, tracked, lost\} \tag{110}$$

The typical trajectory of each type of arm action is shown as Figure 5.4. The coordinates

x and y start from the left bottom corner of the frame. In Figure 5.4, the x and y are both normalized to body height as (111) and (112), where x_{wrist} and x_{center} are the x coordinates of stretching wrist and the body center, y_{wrist} and y_{foot} are the y coordinates of stretching wrist and the lower foot, h_{body} is the body height.

$$x = (x_{wrist} - x_{center})/h_{body} \tag{111}$$

$$y = (y_{wrist} - y_{foot})/h_{body} \tag{112}$$

(a) (b) (c)

Figure 5.3 (a) Particles around the searched wrist point when arm is stretching. (b) Two groups of particles around searched wrist point and detected wrist point. (c) Particle around the searched wrist point when arm has withdrawn

Table 5.1 The status of tracking

Tracking status	Definition
Detected	Stretching arm can be detected.
Tracked	Stretching arm cannot be detected but the standard deviation of weight is larger than Th_{σ}.
Lost	Stretching arm cannot be detected and the standard deviation of weight is smaller than Th_{σ}.

The trajectory of *touching* is like a turn U shape. See Figure 5.4 (a). It means once 'stretching' and once 'withdrawing'. The shape of trajectory of *picking and retuning* is like an M shape. See Figure 5.4 (b). It shows twice 'stretching' and twice 'withdrawing'. The trajectory of *picking and putting* is also like a turn U shape. See Figure 5.4 (c). In Figure 5.4 (c), one can see when the customer picks a product, he or she holds it for a while and then puts it into basket. This is a unique characteristic of *picking and putting* action.

Because the customer can pick a product from the shelf at any height, only the x_{hand} is important to estimate the customer action. Assuming the length of arm is not larger than the half of body height, the normalized hand *x* position (equation (111)) can be quantized by 0.05 from 0 to 0.5. Thus the position of hand *PH* can be defined as below.

$$PH \in \{0, 0.05, 0.10, 0.15, \ldots, 0.50\} \tag{113}$$

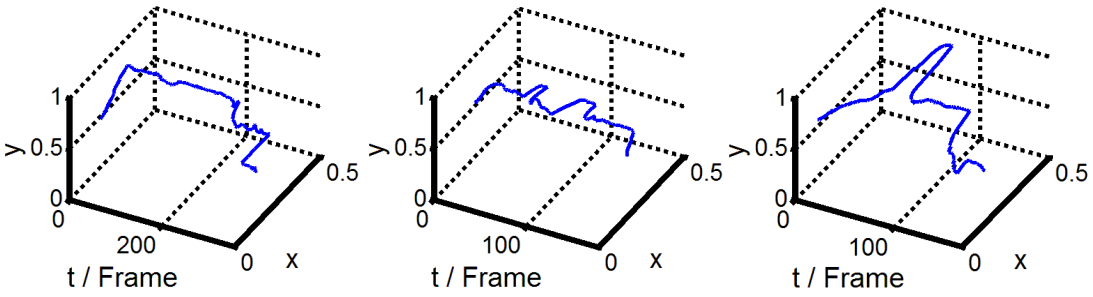
Only the trajectory of hand is not enough to classify different arm actions. The interaction between hand and bask*et al*so needs to be considered. For example, at the end of *touching* action, the hand should be outside the basket. At the end of *picking and putting* action, the hand should be inside the basket. For the *picking and returning* action, when the hand withdraws at the first time, it should be above the basket and when it withdraws at the second time, it should be outside the basket.

To study the relation between hand and basket, the basket should be detected at first. In the Chapter 4.4.2, the feature maps of basket and the basket heatmap are already obtained. If the maximum value in the feature maps is larger than a threshold, the basket will be treat as detected. Empirically, the best threshold value is 0.6. The centroid of basket heatmap can be seen as the center of basket. The corresponding size of window of the layer containing the maximum value is the size of basket.

Then the status of hand is defined as equation (114) and Table 5.2.

$$HS \in \{above, outside, inside\} \tag{114}$$

Note that the status *inside* is defined when the tracking status is *lost*. It means the hand will be regarded as inside the basket only when the hand disappears in the basket region. Otherwise the hand will be considered as just moving through the basket region outside the basket.



(a) (b) (c)

Figure 5.4 Typical trajectories of different arm action. (a) Hand trajectory of *touching* action (b) Hand trajectory of *picking and retuning* action (c) Hand trajectory of *picking and putting*action

Table 5.2 The status of hand

Status of hand	Definition
Above	Wrist point is above the upper edge of basket region or basket line.
Outside	Wrist point is below the upper edge of basket region or the basket line. The tracking status is ‘tracked’ or ‘detected’.
Inside	Wrist point is in the basket region. The tracking status is ‘lost’.

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

Figure 5.5 A tracking result of *picking and putting* action. (a) The hand is detected first time. The particle filter starts. (b) The customer is stretching arm. (c) The customer is picking a product. (d) The customer is taking the product to the shopping basket. (e) The customer is putting the product into the shopping basket. The hand area is inside the basket area. (d) The tracking is terminated

A tracking result of *picking and putting* action is shown as Figure 5.5. The hand is

tracked from the start of 'stretching arm' until 'putting into basket'. Note that in Figure 5.5 (e) the hand can be still tracked when it is inside the basket area. At this time the stretching arm cannot be detected from silhouette. The hand is tracked by comparing the *average hand histogram*. In Figure 5.5 (f) the hand at last disappears in the basket area because the standard deviation of particle weights is smaller than the threshold.

The position of hand *PH*, tracking status *TS* and hand status *HS* are combined as *Combined Hand Feature* (CHF) {*PH, TS, HS*}. The position of hand *PH* records the trajectory of hand movement. The tracking status *TS* describes the stretching status of arm. The hand status *HS* tells the interaction between hand and basket. Combining this information, CHF can be seen as a series of specific codes of each action. In Chapter 5.4, the classification of arm actions by using CHF will be discussed.

5.4. Customer Arm Action Classification using Dynamic Bayesian Network

The CHF {*PH, TS, HS*} of each frame are used to predict the intention *I* of customer. The intention *I* includes 3 types of arm actions. *I* is defined as (115).

$$I \in \{\text{touching, picking and returning, picking and putting}\} \tag{115}$$

Dynamic Bayesian Network (DBN) is applied to classify the arm action. Firstly, a Bayesian Network (BN) is a stochastic model which can be seen as a Directed Acyclic Graph (DAG). The nodes in DAG denote random variables and the edges denote the dependencies between nodes. A DBN is a sequential BN where nodes relate each other in over adjacent time steps.

When the hand is being tracked, its status can be modeled as a DBN which is shown as Figure 5.6. In Figure 5.6, *I* is the intention of customer.*PH*, *TS* and *HS* are the measurements.

From the all obtained measurements $PH_{1:t}$, $TS_{1:t}$, $HS_{1:t}$, the probability of I_t can be determined recursively as follows:

$$p(I_t|PH_{1:t}, TS_{1:t}, HS_{1:t}) \propto$$
$$p(PH_t, TS_t, HS_t|I_t) \sum_{I_{t-1}} p(I_t|I_{t-1}) p(I_{t-1}|PH_{1:t-1}, TS_{1:t-1}, HS_{1:t-1}) \tag{116}$$

$$I_t = arg\,max\, p(I_t|PH_{1:t}, TS_{1:t}, HS_{1:t}) \quad (117)$$

To obtain each value of conditional probability in the right side of (116), a set of training data is needed. Assume there is a set of training video clips which contain *N* frames in total. Thus

$$P(I_t = i|I_{t-1} = j) = N(I_t = I, I_{t-1} = j)/N(I = j) \quad (118)$$

$N(I_t = i, I_{t-1} = j)$ is the number of pairs frames that$I_t = i, I_{t-1} = j$. $N(I = j)$ is the number of frames that $I = j$.

$$p(PH_t = k, TS_t = m, HS_t = n|I_t = i) = N(PH = k, TS = m, HS = n, I = i)/N(I = i) \quad (119)$$

$N(PH = k, TS = m, HS = n, I = i)$ is the number of frames that $I_t = i$ and measurements are $PH = k, TS = m, HS = n$.

Initially,

$$p(I_1|PS_{1:1}, TC_{1:1}, HC_{1:1}) = p(I) \quad (120)$$

$$p(I = i) = N(I = i)/N \quad (121)$$

Note that although the intention *I* cannot be observed directly, it can be judged by human in training process. The estimated I_t will be output when the tracking is *lost*.

5.5. Experiments

5.5.1. Dataset

The experiments of customer arm action classification are based on the customer behavior dataset. The images of customer behavior dataset are shot from real surveillance camera of retail stores, where are from the same source as the customer orientation dataset in Chapter 3.5. The surveillance videos are selected from the security camera near the merchandise shelf and then are cut into clips when a single customer stops on the front of the shelf. Each video clips are convert into one image sequence. The selected videos are from 4 surveillance cameras, which are installed in 4 different retail stores respectively. The occlusion between customers is not included in the dataset.

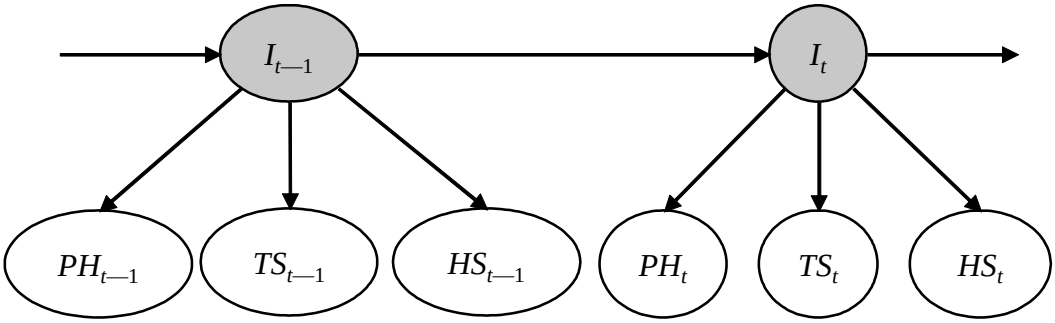


Figure 5.6 Illustration of the DBN model

The customer behavior dataset includes multiple image sequences. Each sequence is labeled one customer behavior. The details of customer behavior dataset are shown as Table 5.3. The *no interest* sequence means neither the customer head and body orientation is facing to the shelf. In the *looking at shelf* sequence, only customer head orientation is facing to the shelf for more than 1 second. In the *turning to shelf* sequence, customer body orientation is facing to the shelf for more than 1 second. In the *looking at shelf* sequence, the customer touches a product but does not pick it from the shelf. In the *picking and returning* sequence, the customer picks a product then puts it back to the shelf. In the *picking and putting* sequence, the customer picks a product then puts it into the shopping basket. In the first 3 kinds of sequences, the customer does not stretch arm. The recognition of these 3 behaviors are only rely on the head and body orientation. The images in the customer behavior dataset are also after background subtraction. The images of customer behavior dataset do not overlap with the images in the customer pose dataset. Furthermore, the actions *no interest*, *looking at shelf*, *turning to shelf* can

Table 5.3 The details of customer behavior dataset

	Sequence number
No interest	20
Looking at shelf	20
Turning to shelf	29
Touching	20
Picking and returning	20
Picking and putting	34
Total	143

be grouped into the *no arm action* class. Therefore the dataset for customer arm action classification can be listed as Table 5.4.

5.5.2. Experiments of Customer Arm Action Classification

In the experiments, the *no arm action* is classified by the stretching arm detection. If there is no stretching arm detected, the behavior will be labeled as *no arm action*. Once the stretching arm is detected, the arm action classification based CHF starts. Leave-one-out cross-validation is applied to evaluate each arm action classification. It means one of arm action will be chosen for testing, and the other cases are for training. Iterate this process until every case is chosen for testing once.

The results are shown as Table 5.5. The overall recognition rate is 90.9%. In Table 5.5, note that all of the wrong samples of *touching* is estimated as *picking and putting* and all the wrong samples of *picking and putting* is estimated as *touching*. The author checked the data of these two actions and found that they are sometimes very similar. If a customer holds the product for a while before put into basket, this action can be clearly classified from the *touching*. Because *touching* will not hold a product for a while. However, if the customer puts the product into basket without hesitation, this action looks very like *touching* from the system's view. Moreover, if the basket cannot be detected due to occlusion, the *picking and putting* action is also difficult to be distinguished from *touching*. In this condition, although some cases can be recognized correctly, all of the wrong cases occur in this condition.

Figure 5.7 shows 4 examples of *picking and putting* action classification results. In Figure 5.7, the first picture of each subfigure is the curves of CHF. To be easily shown

Table 5.4 The dataset for customer arm action classification

	Sequence number
No arm action	69
Touching	20
Picking and returning	20
Picking and putting	34
Total	143

in picture, the values of *PH* {0, 0.05, 0.10, …, 0.5} are assigned {5, 6, 7, …, 15}, the values of *TS* {detected, tracked, lost} are assigned {1, 2, 3} and the values of *HS* {*above*, *outside*, *inside*} are assigned {1, 2, 3}. The *t* axis starts when the arm becomes detected and ends when the particle filter is terminated. Figure 5.7(a) is a correct case. The customer picks a product and hold for a while then puts it into basket. Figure 5.7(b) is also a correct sample. The customer picks a product and puts it into basket without hesitation. Figure 5.7(c) is a wrong case. The bask*et al*most cannot be seen due to the occlusion. Moreover, the customer puts the product into basket without hesitation. The system wrongly classifies the action as *touching*.

Table 5.5 The confusion matrix of customer arm action classification

	No arm action (est.)	**Touching (est.)**	**Picking and returning (est.)**	**Picking and putting (est.)**
No arm action (act.)	**97.1%**	**2.9%**	**0.0%**	**0.0%**
Touching (act.)	0.0%	**85.0%**	**0.0%**	**15.0%**
Picking and returning (act.)	0.0%	**10.0%**	**80.0%**	**10.0%**
Picking and putting (act.)	0.0%	**11.8%**	**0.0%**	**88.2%**

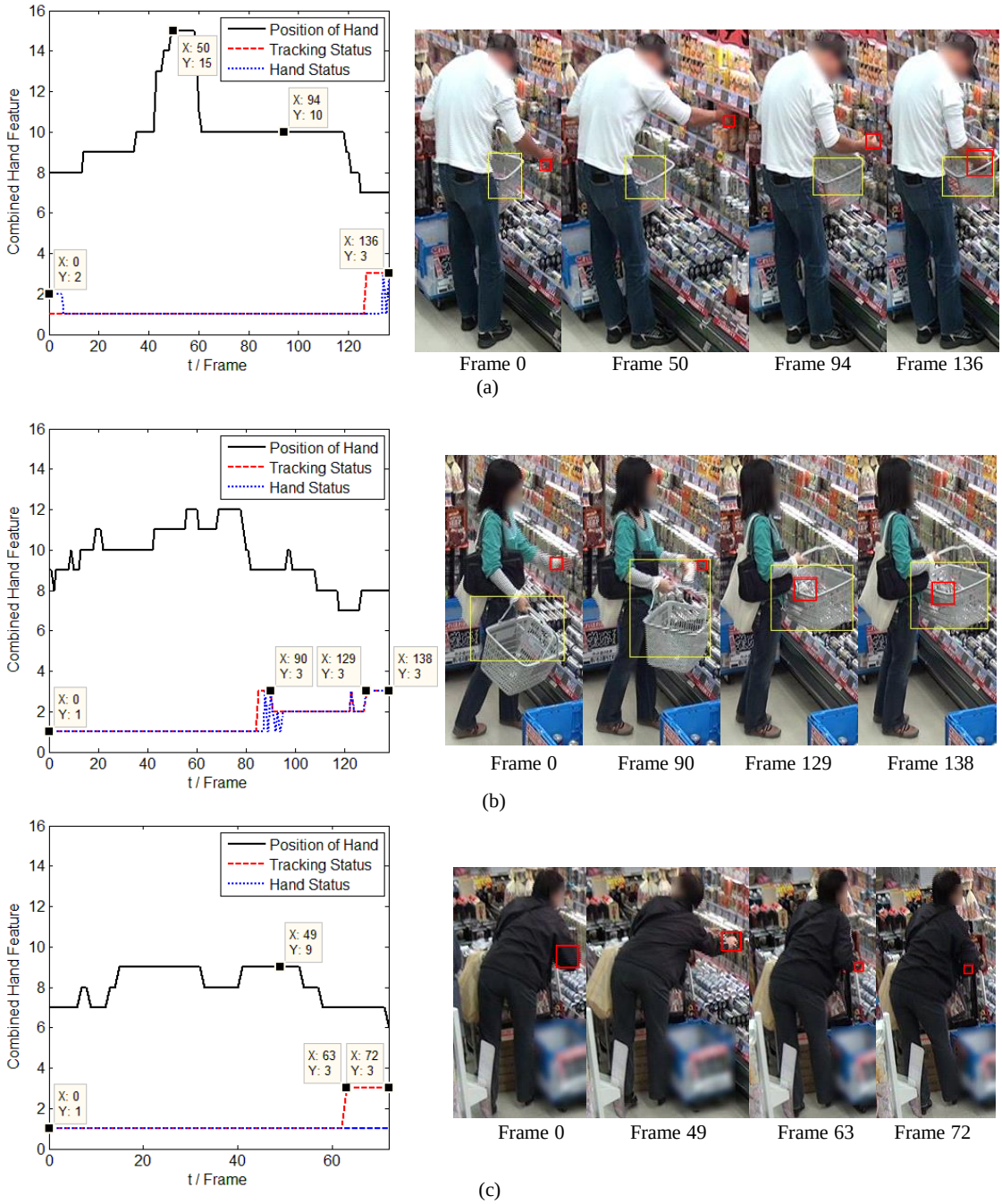


Figure 5.7 Examples of *picking and putting* action classification results. (a) Correct case. The customer picks a product and holds it for a while then puts it into basket. (b) Correct case. The customer picks a product and puts it into basket without hesitation. (c) Wrong case. The basket cannot be detected because of occlusion and the action is wrongly classified as *touching*

A comparison experiment is also conducted as a baseline method which uses appearance-based feature to classify the arm actions. There are two steps in the baseline method: first one is the classification of the appearance of the arm in each frame, second is analysis of the action based on the history of the appearances.

In the first step, the widely-used appearance-based classification method HOG + SVM. Actually, the arm area is cut manually to simulate a perfect arm detection. Then the arm areas are categorized to 6 types according to their appearances. They are given labels {1, 2, …, 6}, which are shown in Figure 5.8. Finally, the HOG features are extracted from each arm area and arm label is classified by SVM. Here, the training and testing data are the same data of arm action in Table 5.6. Therefore, each action can be represented as a series of labels.

In the second step, the same DBN algorithm is applied to classify the arm actions. The difference between the baseline method and the proposed method is that the base line method adopts appearance-based feature to describe the category of each arm action in each frame. The proposed method represents the arm action by CHF feature in each frame. The results are compared in Table 5.6. These results show that the CHF significantly outperforms the appearance-based feature.

Furthermore, by combining the results of customer orientational behavior classification in Chapter 3.5, an overall customer behavior classification results can be list as Table 5.7. The overall recognition rate is 87.4%.

Until here, this dissertation achieves the customer head and body orientation detection, customer pose estimation and customer behavior classification. By using this information,

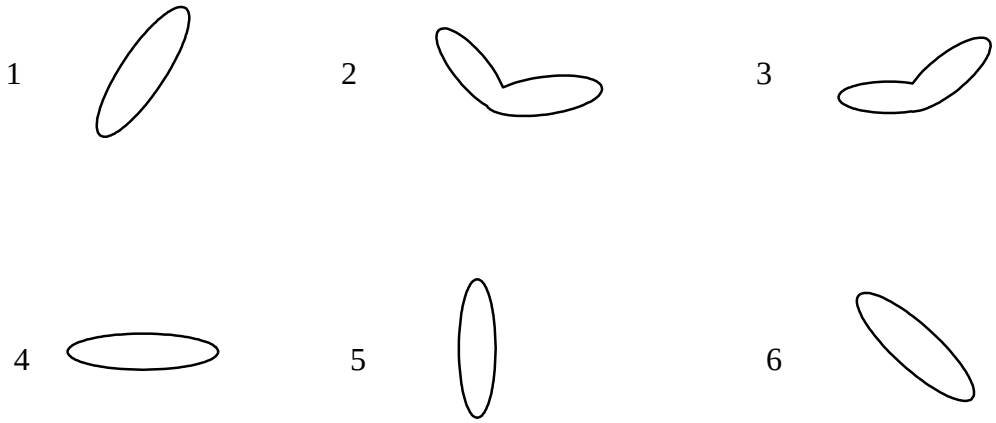


Figure 5.8 The 6 types of arm appearances

retailers can assess the customer interest level. Figure 5.9 shows a demo application for customer interest level estimation. The customer interest is divided into 6 levels from low to high: no interest, just looking, facing, touching, returning and putting. The white arrows mean the head and body orientations of customer. The yellow blocks indicate the shopping basket. Each row in Figure 5.9 indicates the customer interesting level changing with time. This demo shows the essential application of customer interest level estimation for the retail store in the future.

Table 5.6 The recognition rates of arm action by using CHF and HOG + SVM

Arm action	**Proposed method (CHF)**	**HOG + SVM**
Touching	85.0%	70.0%
Picking and Returning	80.0%	70.0%
Picking and Putting	88.2%	70.6%
Overall	**85.1%**	70.3%

Table 5.7 The confusion matrix of overall customer behaviors

	No interest (est.)	Viewing (est.)	Turning to shelf (est.)	Touching (est.)	PR (est.)	PP (est.)
No interest (act.)	95.0%	0.0%	5.0%	0.0%	0.0%	0.0%
Viewing (act.)	5.0%	90.0%	5.0%	0.0%	0.0%	0.0%
Turning to shelf (act.)	6.9%	0.0%	86.2%	6.9%	0.0%	0.0%
Touching (act.)	0.0%	0.0%	0.0%	85.0%	0.0%	15.0%
PR (act.)	0.0%	0.0%	0.0%	10.0%	80.0%	10.0%
PP (act.)	0.0%	0.0%	0.0%	11.8%	0.0%	88.2%

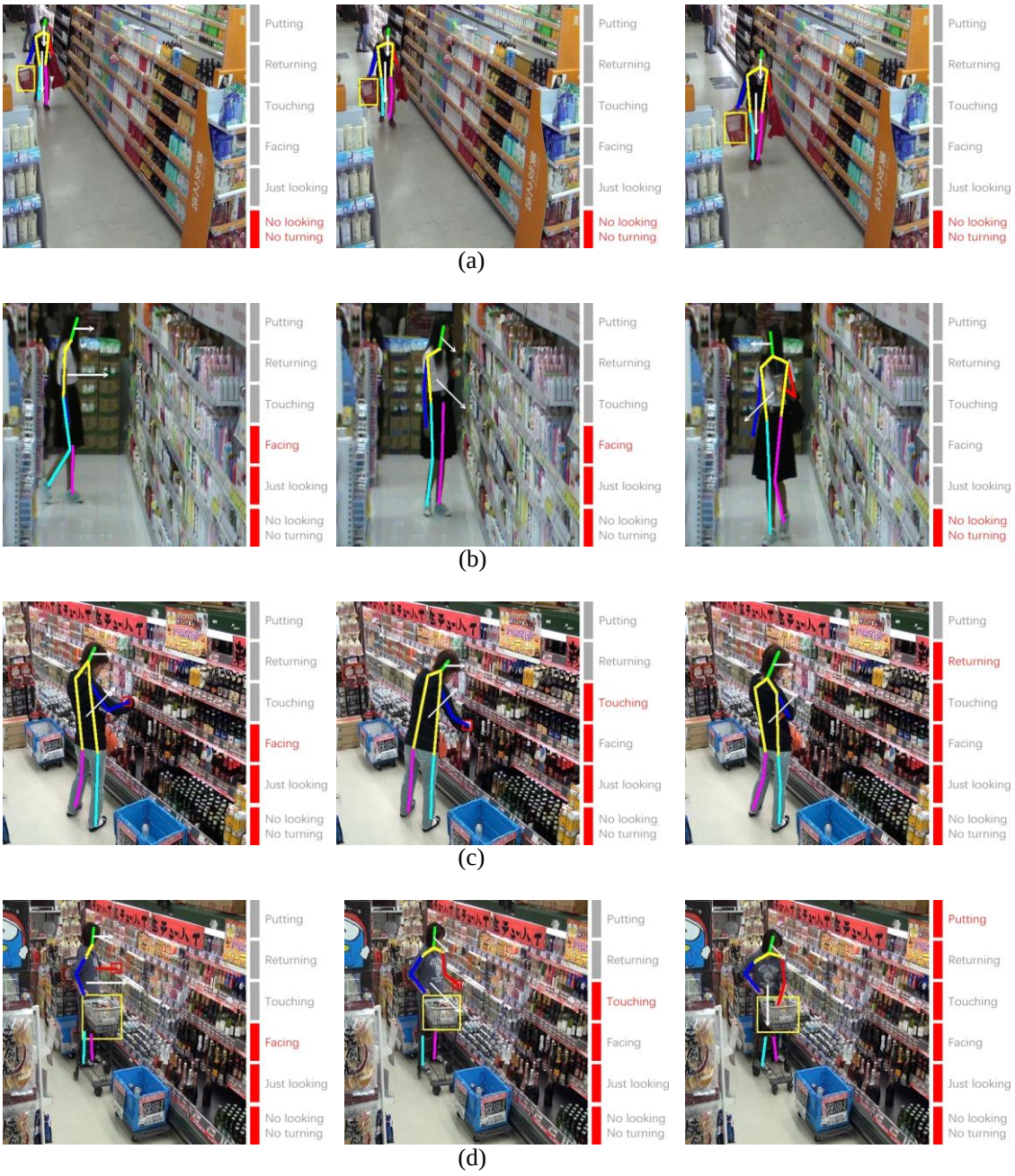


Figure 5.9 A domo application for customer interest level estimation

Chapter 6. Conclusions and Future Works

This dissertation proposes to assess the customer interest level from three main parts: customer head and body orientation detection, customer pose estimation and customer arm action classification.

In the customer head and body orientation detection, a SSL method are proposed to solve the ambiguity problem of training data. Two kinds of classifiers are implemented in the SSL: HOG + SVM classifier and ResNet classifier. The experiments prove that the ResNet based SSL much outperforms the HOG + SVM based SSL and the traditional SL.

In the customer pose estimation, firstly, the DPM + Body Orientation is designed to improve the conventional DPM. Then deep learning method (ResNet) + Body Orientation is introduced to predict one pose and visibility mask per image. In order to increase the candidates of predictions to optimize the estimation, the IntePoseNet is proposed to generate joint heatmaps and visibility mask. It integrally incorporate the joint heatmaps with the global body orientation, local joint connections and visibility mask. Finally, a multi-task neural network is proposed to simultaneously output the joint heatmaps, body orientation, joint connection maps and visibility mask. In the multi-task learning, the tasks can improve the performance other another. Experimental results show that the multi-task neural network achieves the best performance.

In customer arm action classification. A novel Combined Hand Feature (CHF) is proposed to combine hand trajectory, tracking status and the relative position between hand and shopping basket, to describe the arm action in each frame. The CHF outperforms the conventional appearance-base feature in the experiments.

This dissertation only focuses on the single customer without intra-occlusion. As the future works, the occlusion problem should be deal with. Although the occlusion problem is difficult in practice, the future work can start from the evaluation of occlusion. If the occlusion between customer can be corrected estimated, the scene with serious occlusion may be excluded automatically. If the occlusion is partial, parts of customer behaviors could be analyzed by using the methods proposed by this dissertation or by using some novel technologies. Another way is to apply the bottom-up method in the

multi-task neural network. The joint heatmaps and joint connection maps can be directly used for multi-person. Only the body orientation and visibility mask need to adopt new approaches for the multi-person pose estimation.

Reference

[1] “https://service.abeja.asia/.”

[2] “https://www.intelli-vision.com/smart-retail/.” .

[3] “http://www.seqsecurity.com/.” .

[4] I. Haritaoglu and M. Flickner, “Detection and tracking of shopping groups in stores,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. 431–438.

[5] A. Leykin and M. Tuceryan, “Detecting shopper groups in video sequences,” in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2007, pp. 417–422.

[6] A. W. Senior *et al.*, “Video analytics for retail,” in *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2007, pp. 423–428.

[7] M. Popa, L. Rothkrantz, Z. Yang, P. Wiggers, R. Braspenning, and C. Shan, “Analysis of shopping behavior based on surveillance system,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2010, pp. 2512–2519.

[8] Y. Hu, L. Cao, F. Lv, S. Yan, Y. Gong, and T. S. Huang, “Action detection in complex scenes with spatial and temporal ambiguities,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 128–135.

[9] I. Haritaoglu, D. Beymer, and M. Flickner, “Ghost3D: detecting body posture and parts using stereo,” in *Proceedings of the IEEE Workshop on Motion and Video Computing*, 2002, pp. 175–180.

[10] W. Lao, J. Han, and others, “Automatic video-based human motion analyzer for consumer surveillance system,” *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, pp. 591–598, 2009.

[11] I. Haritaoglu and M. Flickner, “Attentive billboards: towards to video based customer behavior understanding,” in *Proceedings of the IEEE Workshop on Applications of Computer Vision*, 2002, pp. 127–131.

[12] C. Migniot and F. Ababsa, “3D Human Tracking from Depth Cue in a Buying

Behavior Analysis Context," in *Proceedings of the International Conference on Computer Analysis of Images and Patterns*, 2013, pp. 482–489.

[13] S. Sae-ueng, A. Ogino, and T. Kato, "Modeling personal preference using shopping behaviors in ubiquitous information environment," in *Proceedings of Data Engineering Workshop*, 2007.

[14] C. E. Stan, D. Dumitrescu, V. Caras, D. E. Tiliute, E. Pop, and L. E. Anghel, "Intelligent store-an innovative technological solution for retail activities with mobile access," in *Proceedings of the International Multi-Conference on Computing in the Global Information Technology*, 2008, pp. 7–11.

[15] K.-D. Lee, M. Y. Nam, K.-Y. Chung, Y.-H. Lee, and U.-G. Kang, "Context and profile based cascade classifier for efficient people detection and safety care system," *Multimedia Tools and Applications (MTAP)*, vol. 63, no. 1, pp. 27–44, Apr. 2012.

[16] S. Abe, M. Morimoto, and K. Fujii, "Estimating face direction from wideview surveillance camera," in *Proceedings of the World Automation Congress (WAC)*, 2010, pp. 1–6.

[17] A. Schulz, N. Damer, M. Fischer, and R. Stiefelhagen, "Combined head localization and head pose estimation for video–based advanced driver assistance systems," in *Proceedings of the Joint Pattern Recognition Symposium*, 2011, pp. 51–60.

[18] A. Schulz and R. Stiefelhagen, "Video-based pedestrian head pose estimation for risk assessment," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2012, pp. 1771–1776.

[19] T. Gandhi and M. M. Trivedi, "Image based estimation of pedestrian orientation for improving path prediction," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2008, pp. 506–511.

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

[21] M. Goffredo, I. Bouchrika, J. N. Carter, and M. S. Nixon, "Performance analysis for automated gait extraction and recognition in multi-camera surveillance," *Multimedia Tools and Applications (MTAP)*, vol. 50, no. 1, pp. 75–94, Oct. 2009.

[22] C. Chen, A. Heili, and J. Odobez, “Combined estimation of location and body pose in surveillance video,” in *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2011, pp. 5–10.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[24] C. Sminchisescu and A. Telea, “Human Pose Estimation from Silhouettes. A Consistent Approach Using Distance Level Sets,” in *Proceedings of the International Conference on Computer Graphics,Visualization and Computer Vision (WSCG)*, 2002.

[25] D. K. Wagg and M. S. Nixon, “Model-Based Gait Enrolment in Real-World Imagery,” in *Proceedings of the Workshop on Multimodal User Authentication*, 2003, pp. 189–195.

[26] F. Tafazzoli and R. Safabakhsh, “Model-based human gait recognition using leg and arm movements,” *Engineering Applications of Artificial Intelligence*, vol. 23, no. 8, pp. 1237–1246, Dec. 2010.

[27] L. Zhao, “Dressed human modeling, detection, and parts localization,” Doctoral Dissertation, Carnegie Mellon University Pittsburgh, PA, 2001.

[28] A. Mittal, L. Zhao, and L. S. Davis, “Human body pose estimation using silhouette shape analysis,” in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*, 2003, pp. 263–270.

[29] A. K. S. Kushwaha, S. Srivastava, and R. Srivastava, “Multi-view human activity recognition based on silhouette and uniform rotation invariant local binary patterns,” *Multimedia Systems*, pp. 1–17, Mar. 2016.

[30] D. Ramanan, D. A. Forsyth, and A. Zisserman, “Tracking People by Learning Their Appearance,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 29, no. 1, pp. 65–81, Jan. 2007.

[31] M. Andriluka, S. Roth, and B. Schiele, “Monocular 3D pose estimation and tracking by detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 623–630.

[32] M. Andriluka, S. Roth, and B. Schiele, “Pictorial structures revisited: People

detection and articulated pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 1014–1021.

[33] A. Moutzouris, J. Martinez-del-Rincon, M. Lewandowski, J. Nebel, and D. Makris, "Human pose tracking in low dimensional space enhanced by limb correction," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2011, pp. 2301–2304.

[34] D. Weiss, B. Sapp, and B. Taskar, "Sidestepping intractable inference with structured ensemble cascades," in *Advances in Neural Information Processing Systems*, 2010, pp. 2415–2423.

[35] B. Sapp, D. Weiss, and B. Taskar, "Parsing human motion with stretchable models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1281–1288.

[36] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari, "2d articulated human pose estimation and retrieval in (almost) unconstrained still images," *International Journal of Computer Vision (IJCV)*, vol. 99, no. 2, pp. 190–214, 2012.

[37] A. Cherian, J. Mairal, K. Alahari, and C. Schmid, "Mixing Body-Part Sequences for Human Pose Estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2353–2360.

[38] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision (IJCV)*, vol. 61, no. 1, pp. 55–79, Jan. 2005.

[39] M. Sun and S. Savarese, "Articulated part-based model for joint object detection and pose estimation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011, pp. 723–730.

[40] M. Dantone, J. Gall, C. Leistner, and L. Van Gool, "Human pose estimation using body parts dependent joint regressors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3041–3048.

[41] Y. Yang and D. Ramanan, "Articulated Human Detection with Flexible Mixtures of Parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 12, pp. 2878–2890, Dec. 2013.

[42] M. Eichner and V. Ferrari, "Appearance Sharing for Collective Human Pose

Estimation," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2012, pp. 138–151.

[43] S. Li, M. Zhang, S. Su, B. Shuai, and R. Ji, "Decomposed human localization from social photo album," *Multimedia Systems*, vol. 22, no. 1, pp. 137–148, Feb. 2016.

[44] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Poselet Conditioned Pictorial Structures," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 588–595.

[45] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Neural Information Processing Systems (NIPS)*, 1989.

[46] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, 1995.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[49] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[50] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1653–1660.

[51] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4733–4742.

[52] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei, "Towards viewpoint invariant 3D human pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016, pp. 160–177.

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

[54] X. Chen and A. L. Yuille, “Parsing occluded people by flexible compositions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3945–3954.

[55] J. 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*, 2014, pp. 1799–1807.

[56] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, “Efficient object localization using convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 648–656.

[57] W. Yang, W. Ouyang, H. Li, and X. Wang, “End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4715–4723.

[58] X. Chu, W. Ouyang, H. Li, and X. Wang, “Structured feature learning for pose estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4715–4723.

[59] A. Jain, J. Tompson, Y. LeCun, and C. Bregler, “MoDeep: A Deep Learning Framework Using Motion Features for Human Pose Estimation,” in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2014, pp. 302–315.

[60] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, “DeepFlow: Large Displacement Optical Flow with Deep Matching,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 1385–1392.

[61] T. Pfister, J. Charles, and A. Zisserman, “Flowing convnets for human pose estimation in videos,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1913–1921.

[62] H. Azizpour and I. Laptev, “Object detection using strongly-supervised deformable part models,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012, pp. 836–849.

[63] G. Ghiasi, Y. Yang, D. Ramanan, and C. C. Fowlkes, “Parsing occluded people,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2401–2408.

[64] U. Rafi, J. Gall, and B. Leibe, “A semantic occlusion model for human pose estimation from a single depth image,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2015, pp. 67–74.

[65] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[66] R. Camana, “Multi-task learning,” *Machine Learning*, vol. 28, pp. 41–75, 1997.

[67] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, “Facial landmark detection by deep multi-task learning,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 94–108.

[68] Y. Tian, P. Luo, X. Wang, and X. Tang, “Pedestrian detection aided by deep learning semantic tasks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5079–5087.

[69] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim, “Rotating your face using multi-task deep neural network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 676–684.

[70] Y. Kao, R. He, and K. Huang, “Visual aesthetic quality assessment with multi-task deep learning,” *Computing Research Repository (CoRR)*, Apr. 2016.

[71] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[72] L. Zelnik-Manor and M. Irani, “Event-based analysis of video,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, vol. 2, pp. 123–130.

[73] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: a local SVM approach,” in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2004, vol. 3, p. 32–36 Vol.3.

[74] J. Liu and M. Shah, “Learning human actions via information maximization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR), 2008, pp. 1–8.

[75] J. C. Niebles and L. Fei-Fei, "A Hierarchical Model of Shape and Appearance for Human Action Classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.

[76] R. Benmokhtar, "Robust human action recognition scheme based on high-level feature fusion," *Multimedia Tools and Applications (MTAP)*, vol. 69, no. 2, pp. 253–275, Mar. 2012.

[77] H. Trinh, Q. Fan, J. Pan, P. Gabbur, S. Miyazawa, and S. Pankanti, "Detecting human activities in retail surveillance using hierarchical finite state machine," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 1337–1340.

[78] M. S. Ryoo and J. K. Aggarwal, "Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 1593–1600.

[79] M. S. Ryoo and J. K. Aggarwal, "Semantic representation and recognition of continued and recursive human activities," *International journal of computer vision*, vol. 82, no. 1, pp. 1–24, 2009.

[80] D. Weinland, M. Özuysal, and P. Fua, "Making action recognition robust to occlusions and viewpoint changes," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010, pp. 635–648.

[81] L. Shao, L. Ji, Y. Liu, and J. Zhang, "Human action segmentation and recognition via motion and shape analysis," *Pattern Recognition Letters*, vol. 33, no. 4, pp. 438–445, 2012.

[82] W. Choi and S. Savarese, "A unified framework for multi-target tracking and collective activity recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012, pp. 215–230.

[83] L. Shao, X. Zhen, D. Tao, and X. Li, "Spatio-temporal Laplacian pyramid coding for action recognition," *IEEE Transactions on Cybernetics*, vol. 44, no. 6, pp. 817–827, 2014.

[84] L. Liu, L. Shao, X. Li, and K. Lu, "Learning spatio-temporal representations

for action recognition: A genetic programming approach," *IEEE transactions on cybernetics*, vol. 46, no. 1, pp. 158–170, 2016.

[85] M. Elmezain, A. Al-Hamadi, and B. Michaelis, "Hand trajectory-based gesture spotting and recognition using HMM," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 3577–3580.

[86] E. Shechtman and M. Irani, "Space-time behavior based correlation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, vol. 1, pp. 405–412 vol. 1.

[87] F. Chen and W. Wang, "Activity recognition through multi-scale dynamic bayesian network," in *Proceedings of the International Conference on Virtual Systems and Multimedia (VSMM)*, 2010, pp. 34–41.

[88] K. P. Murphy, "Dynamic bayesian networks: representation, inference and learning," Doctoral Dissertation, University of California, Berkeley, 2002.

[89] Y. Zhu, N. M. Nayak, and A. K. Roy-Chowdhury, "Context-aware activity modeling using hierarchical conditional random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 37, no. 7, pp. 1360–1372, 2015.

[90] J. Yao and J.-M. Odobez, "Multi-layer background subtraction based on color and texture," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2007, pp. 1–8.

[91] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1999, vol. 2, pp. 1150–1157.

[92] S. Belongie, J. Malik, and J. Puzicha, "Matching shapes," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2001, vol. 1, pp. 454–461.

[93] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Annual Workshop on Computational Learning Theory*, 1992, pp. 144–152.

[94] C. Jin and L. Wang, "Dimensionality dependent PAC-Bayes margin bound," in *Advances in Neural Information Processing Systems*, 2012, pp. 1034–1042.

[95] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, “Pegasos: Primal estimated sub-gradient solver for svm,” *Mathematical programming*, vol. 127, no. 1, pp. 3–30, 2011.

[96] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, “A dual coordinate descent method for large-scale linear SVM,” in *Proceedings of the International Conference on Machine Learning*, 2008, pp. 408–415.

[97] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[99] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[100] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Advances in Neural Information Processing Systems*, 2007, pp. 153–160.

[101] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015, pp. 448–456.

[102] C. Chen, A. Heili, and J.-M. Odobez, “A joint estimation of head and body orientation cues in surveillance video,” in *Proceedings of IEEE International Conference on Computer Vision (ICCV) Workshops*, 2011, pp. 860–867.

[103] E. Rehder, H. Kloeden, and C. Stiller, “Head detection and orientation estimation for pedestrian safety,” in *Proceedings of International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 2292–2297.

[104] D. Park and D. Ramanan, “Articulated pose estimation with tiny synthetic videos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2015, pp. 58–66.

[105] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE Transactions on*

Pattern Analysis and Machine Intelligence (PAMI), vol. 32, no. 9, pp. 1627–1645, 2010.

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

[107] M. P. Kumar, A. Zisserman, and P. H. Torr, "Efficient discriminative learning of parts-based models," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 552–559.

[108] D. Ramanan, "Dual coordinate solvers for large-scale structural SVMs," *arXiv preprint arXiv:1312.1743*, 2013.

[109] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[110] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

[111] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[112] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2758–2766.

[113] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[114] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2625–2634.

[115] S. Johnson and M. Everingham, "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2010.

[116] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, "Towards understanding action recognition," in *Proceedings of the IEEE International*

Conference on Computer Vision (ICCV), 2013, pp. 3192–3199.

[117] S. Johnson and M. Everingham, "Learning effective human pose estimation from inaccurate annotation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1465–1472.

[118] U. Rafi, B. Leibe, J. Gall, and I. Kostrikov, "An Efficient Convolutional Network for Human Pose Estimation.," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2016, vol. 1, p. 2.

[119] X. Yu, F. Zhou, and M. Chandraker, "Deep deformation network for object landmark localization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[120] B. Xiaohan Nie, C. Xiong, and S.-C. Zhu, "Joint action recognition and pose estimation from video," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2015, pp. 1293–1301.

[121] U. Iqbal, M. Garbade, and J. Gall, "Pose for action-action for pose," in *IEEE International Conference on Automatic Face & Gesture Recognition*, 2017, pp. 438–445.

[122] J. Song, L. Wang, L. Van Gool, and O. Hilliges, "Thin-Slicing Network: A Deep Structured Model for Pose Estimation in Videos," in *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2017.

Publication List

International Journal:

Jingwen Liu, Yanlei Gu, and Shunsuke Kamijo. "Integral Customer Pose Estimation Using Body Orientation and Visibility Mask." *Multimedia Tools and Applications (MTAP)* (Submitted).

Jingwen Liu, Yanlei Gu, and Shunsuke Kamijo. "Customer Pose Estimation using Orientational Spatio-Temporal Network from Surveillance Camera." *Multimedia Systems* (Second Review).

Jingwen Liu, Yanlei Gu, and Shunsuke Kamijo. "Customer behavior classification using surveillance camera for marketing." *Multimedia Tools and Applications (MTAP)*, Feb 2016, pp. 1-28.

International Conference:

Jingwen Liu, Yanlei Gu, and Shunsuke Kamijo. "Joint Customer Pose and Orientation Estimation using Deep Neural Network from Surveillance Camera." in *IEEE International Symposium on Multimedia (ISM)*, Dec 2016, pp. 216–221.

Jingwen Liu, Yanlei Gu, and Shunsuke Kamijo. "Customer Behavior Recognition in Retail Store from Surveillance Camera." in *IEEE International Symposium on Multimedia (ISM)*, Dec 2015, pp. 154–159.

Domestic Conference:

Jingwen Liu, Yanlei Gu, and Shunsuke Kamijo. "Orientation based Customer Pose Estimation from Surveillance Camera." in *Pattern Recognition and Media Understanding (PRMU)*, Mar 2016, pp. 167-172.

Jingwen Liu, Yanlei Gu, and Shunsuke Kamijo. "Recognition of Customer Behavior on the Front of Shelf from Surveillance Camera." in *Computer Vision and Image Media*

(CVIM), Sep 2015, pp. 1-6.