



國立臺灣大學電機資訊學院電機工程學研究所

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

使用合成資料搭配領域適應學習無關視角姿勢特徵

進行跨視角動作辨識

Cross-View Action Recognition Using View-Invariant Pose Feature
Learned from Synthetic Data with Domain Adaptation

楊侑寰

Yu-Huan Yang

指導教授：傅立成 博士

Advisor: Li-Chen Fu, Ph.D.

中華民國 107 年 8 月

August, 2018

國立臺灣大學碩士學位論文
口試委員會審定書

使用合成資料搭配領域適應學習無關視角姿勢特徵
進行跨視角動作辨識

Cross-View Action Recognition Using View-Invariant Pose Feature
Learned from Synthetic Data with Domain Adaptation

本論文係 楊侑宸 君（學號 R05921001）在國立臺灣大學電機工程學系完成之碩士學位論文，於民國 107 年 07 月 31 日承下列考試委員審查通過及口試及格，特此證明。

口試委員：

傅立成

（簽名）

（指導教授）

黃正民

王鈺倫

陳弘海

蔡欽倫

系主任

劉志文

（簽名）

誌謝



回首過去的自己，碩班的這兩年來，有了太多的改變。首先要感謝我的父母支持我做的轉系決定，從機械跨足到電機領域，即使比別人多花了一年的時間，也能夠追求自己更有興趣的事物。

再來也要謝謝我的指導老師 傅立成教授，讓我對於學術研究的態度、方法和解決問題的能力有了顯著的成長，老師對於學術技術的好奇心以及堅持是我最好的效仿對象，除此之外，老師平時的待人處事之道也值得我學習，謝謝老師。感謝電機及資工影像組碩二的同學：佐新、子駿、柏文、翔宇、興宇，一起度過了許多次考驗，參與了無數次 meeting，彼此分享技術、心得；也謝謝 ACL 其他的夥伴：孟皓、緯軒、嘉梁、力愷、宗澤、秉蒼、彥程、竣棠、Vicente、Cesar，碩班的兩年來有你們的陪伴，大家一起玩樂、學習，真的很棒，也很謝謝彥成，即使離開了實驗室依然願意花時間陪我討論和給我建議，增加我的信心，讓我的碩論更完整。

另外謝謝安陞學長默默地幫我們做了許多事，減輕了影像組的負擔，讓我們可以更專心在研究上面，還有感謝電機影像組的學弟宇閔，在我最需要幫忙的時候二话不說地答應我的要求，未來的路上一定也會充滿挑戰，相信你可以運用從我們身上學習到的經驗迎刃而解。

轉系不是件輕鬆的事情，為了彌補自己的不足，比別人付出更多的努力，很慶幸自己都撐過來了，希望我的堅持沒有辜負身旁人對我的期待，未來也要繼續面對眼前的挑戰。

侑寰 July 8, 2018

摘要



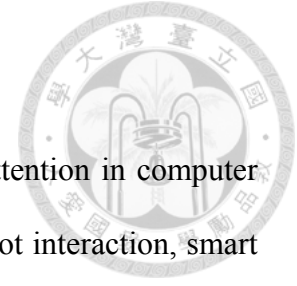
近年來，根據影片了解人們動作的技術獲得越來越多的關注，因為其有廣大的應用場域像是人機互動、智慧家庭、健康照顧以及監視系統。但是隨著視角的不同，人的輪廓外觀也會跟著不同，這造成了從不同的視角進行動作辨識仍然是個挑戰。

在本論文中，我們學習了一個無關視角的姿勢特徵以進行跨視角動作辨識，另一方面，考慮到人們的隱私問題，我們捨棄了彩度影像而只採用深度影像當作我們系統的輸入。此提出的特徵提取模型運用深度卷積神經網路將來自不同視角的人物姿勢轉換到共享的特徵空間中，然而訓練這樣的深度模型需要龐大的多視角影像資料，人為蒐集和標注這樣的資料將會耗費不少的成本與精力，因此我們藉由合成的方式產生了一個多視角的姿勢資料庫，在模擬環境中我們將人體的立體幾何模型貼合到真實的動作捕捉資料上並且在虛擬環境裡進行多視角的深度影像拍攝。

我們以無監督的方式在所創造的合成資料庫上進行無關視角姿勢特徵的學習，此外，為了確保從合成資料到真實資料上的模型遷移性，我們採用了領域適應的技巧去降低彼此的領域差異性。一個動作可以視為一連串的姿勢序列所組成，藉由長短期記憶網路我們可以習得動作的時序模型。在實驗的部分，我們將所提出的方法實作在兩個公開的多視角動作資料庫，其表現超越了幾個基本比較模型，並且同時超越了許多當前最好的方法。

關鍵字：動作辨識、跨視角、合成資料、領域適應

ABSTRACT



Human action understanding from videos has raised lots of attention in computer vision recently because of its wide applications, such as human-robot interaction, smart home, health care, and surveillance systems. Recognizing human activities from different viewpoints is still a challenging problem since human shapes appear quite differently from different viewpoints.

In this thesis, we learn a View-Invariant Pose (VIP) feature representation for cross-view action recognition. Besides, considering privacy issue, we adopt depth video rather than RGB video as input to our system. The proposed VIP feature encoder is a deep Convolutional Neural Network (CNN) that transfers human poses from different viewpoints to a shared high-level feature space. Learning such a deep model requires a large corpus of multi-view data which is very expensive to collect and label. Therefore, we synthesize a Multi-View Pose (MVP) dataset by fitting human physical models with real motion capture data in the simulators and then render depth images from multiple viewpoints.

The VIP feature is learned from the synthetic MVP dataset in an unsupervised way. Moreover, domain adaptation is employed to ensure the transferability from synthetic data to real data such that the domain difference is minimized. An action can be considered as a sequence of poses and the temporal progress is learned and modeled by Long Short-Term Memory (LSTM). In the experimental parts, our method is applied on two benchmark datasets of multi-view 3D human action and achieves superior performance when compared with baseline models as well as promising results when compared with several state-of-the-arts.

Keywords: action recognition, cross-view, synthetic data, domain adaptation

TABLE OF CONTENTS



口試委員會審定書.....	#
誌謝.....	I
摘要.....	II
ABSTRACT.....	III
TABLE OF CONTENTS.....	IV
LIST OF FIGURES.....	VI
LIST OF TABLES.....	X
Chapter 1 Introduction.....	1
1.1 Motivation.....	1
1.2 Literature Review.....	3
1.2.1 Action Recognition with Deep Neural Networks.....	3
1.2.2 Cross-View Action Recognition.....	4
1.2.3 Domain Adaptation.....	7
1.3 Contributions.....	8
1.4 Thesis Organization.....	8
Chapter 2 Preliminaries.....	10
2.1 Cluster Analysis and HDBSCAN.....	10
2.2 Convolutional Neural Network.....	14
2.2.1 Convolutional Layer.....	16
2.2.2 Xception Network.....	17
2.3 Recurrent Neural Network and Long Short-Term Memory.....	19
2.4 Generative Adversarial Network.....	21
2.5 Domain Adaptation.....	22

Chapter 3	Methodology	25
3.1	Synthesize a Multi-View Pose Dataset	25
3.1.1	Build a Pose Dictionary	26
3.1.2	Create 3D Human Models	29
3.1.3	Render Depth Images	31
3.2	Learn a View-Invariant Pose Feature	34
3.2.1	Unsupervised Learning.....	35
3.2.2	Adversarial Domain Adaptation	38
3.3	Model Temporal Information	43
Chapter 4	Experiments	45
4.1	Action Datasets	45
4.1.1	NTU RGB+D Action Recognition Dataset	45
4.1.2	UWA 3D Multi-View Activity II Dataset	47
4.2	Implementation Details	48
4.2.1	Synthesize a Multi-View Pose Dataset.....	49
4.2.2	Architecture Design.....	50
4.2.3	Training Details	51
4.3	Cross-View Pose Classification	53
4.4	Action Recognition Results	55
4.4.1	Action Recognition Pipeline.....	57
4.4.2	The Result of NTU RGB+D Action Recognition Dataset.....	58
4.4.3	The Result of UWA 3D Multi-View Activity II Dataset	59
Chapter 5	Conclusion and Future Works	62
	REFERENCE.....	63



LIST OF FIGURES

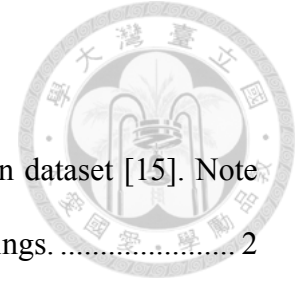


Figure 1-1 Viewpoint variations in NTU RGB+D action recognition dataset [15]. Note that (b) and (c) show the difference between camera settings. 2

Figure 1-2 Long-term recurrent convolutional network proposed in [10]..... 4

Figure 1-3 Transferring source view and target view into a predefined canonical view through non-linear transformations [26]. 6

Figure 1-4 The framework of Robust Non-linear Knowledge Transfer Model (R-NKTM) proposed in [25]. 6

Figure 1-5 Domain adaptation using gradient reversal layer [35]. 7

Figure 2-1 A clustering example containing 2D data points from 3 clusters..... 10

Figure 2-2 Core distance and mutual reachability distance where $mpts = 5$ 11

Figure 2-3 Minimum spanning tree and cluster hierarchy constructed by HDBSCAN [38].
..... 12

Figure 2-4 A condensed tree from the cluster hierarchy and the clustering result. The widths of lines represent the number of points in each cluster. 13

Figure 2-5 A comparison of clustering results between k-means and HDBSCAN. The gray points in (b) are considered as noise. 14

Figure 2-6 A common CNN architecture comprised of several convolutional layers. Every layer transforms the 3D input volume to a 3D output volume of neurons... 15

Figure 2-7 Forward pass of a 3×3 convolutional operation with stride equal to 1. 16

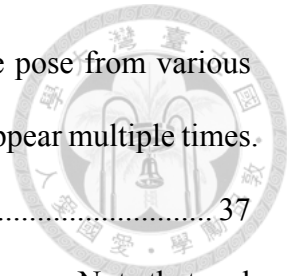
Figure 2-8 The family of Inception modules. 17

Figure 2-9 The difference between traditional and depth-wise convolution using 3×3 convolution kernels. 18

Figure 2-10 The Xception network architecture [41]. 19

Figure 2-11 Structures of the neurons in Recurrent Neural Networks (RNNs).....	19
Figure 2-12 The goal of the generator in GAN [22]. The generator tries to produce images with the similar distribution to real ones.....	22
Figure 2-13 Adversarial training process in GAN [22]. GAN is trained by simultaneously updating the discriminative function (blue, dashed line) to make it tell the difference between real data distribution (black, dotted line) and the generative distribution (green, solid line).	22
Figure 2-14 Domain adaptation attempts to minimize the domain shift.	23
Figure 2-15 The general framework for adversarial domain adaptation proposed in [34].	23
Figure 3-1 Proposed pipeline for synthesizing a Multi-View Pose (MVP) dataset.	25
Figure 3-2 The setting of body markers in CMU motion capture database [20].	26
Figure 3-3 Some representative poses in the pose dictionary. Left column: Skeleton data from CMU motion capture database [20]; Center column: Human models fitted with poses; Right column: Rendered depth images.....	29
Figure 3-4 The GUI of MakeHuman [44].	30
Figure 3-5 3D human models with different combinations of gender, body shape, hair style, and clothes created by MakeHuman [44].	31
Figure 3-6 The GUI of Blender [45].	32
Figure 3-7 Multiple virtual cameras are uniformly placed around the subject.	33
Figure 3-8 Some depth images in the synthetic Multi-View Pose (MVP) dataset. Note that the pixel value indicates the distance. The darker the pixel, the closer it is.	34
Figure 3-9 Unsupervised training architecture for learning View-Invariant Pose (VIP) feature from synthetic data.	35
Figure 3-10 2D t-SNE visualization of View-Invariant Pose (VIP) features extracted from	

synthetic MVP dataset. Each cluster represents the same pose from various views. Due to the limit of palettes, the same color might appear multiple times.



..... 37

Figure 3-11 The visual difference between real and synthetic depth images. Note that real images contain blurred contours caused by noise..... 38

Figure 3-12 2D t-SNE visualization of the domain shift between View-Invariant Pose (VIP) features extracted from synthetic and real images. Orange points represent VIP features from synthetic data while blue points denote the VIP features from real data..... 38

Figure 3-13 Unsupervised training architecture for learning View-Invariant Pose (VIP) feature with domain adaptation..... 39

Figure 3-14 The process of learning View-Invariant Pose (VIP) feature with domain adaptation by Algorithm 1. Orange points represent VIP features from synthetic data while blue points denote the VIP features from real data. Note that the domain shift is minimized during the learning process. 42

Figure 3-15 Modeling temporal information in an action sequence using LSTM. 43

Figure 4-1 Multi-view RGB and depth images of *drinking water* from NTU RGB+D action recognition dataset [15]..... 47

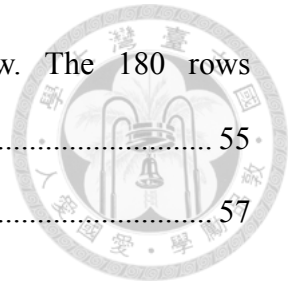
Figure 4-2 Multi-view RGB and depth images of *one-hand waving* from UWA 3D multi-view activity II dataset [27]. Note that some body parts might not be fully captured by the camera. 48

Figure 4-3 The training stages in our proposed method. DA is an abbreviation for domain adaptation..... 51

Figure 4-4 Qualitative visualization of view-invariant property. Each image represents a pose in the synthetic MVP dataset. Each row in an image denotes a VIP

feature $f_x^{VIP} \in \mathbb{R}^{256}$ extracted from a specific view. The 180 rows correspond to 180 viewpoints of the same pose. 55

Figure 4-5 The pipeline of action recognition from input video. 57

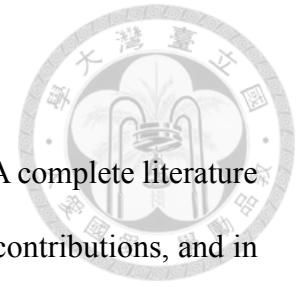


LIST OF TABLES



Table 4-1 The camera settings of NTU RGB+D action recognition dataset [15].....	46
Table 4-2 Pose classification accuracy on the synthetic MVP dataset.....	53
Table 4-3 Comparison of action recognition accuracy (%) on the NTU RGB+D Action Recognition Dataset [15]. Our proposed method is denoted as VIP w/ DA. Our defined baseline models are Xception + LSTM and VIP w/o DA.	59
Table 4-4 Comparison of action recognition accuracy (%) on the UWA 3D multi-view activity II dataset [27]. Each column represents a different cross-view setting. For example, <i>V123</i> means the model is trained on view 1 and view 2 while tested on view 3. Our proposed method is denoted as VIP w/ DA. Our defined baseline models are Xception + LSTM and VIP w/o DA.....	60

Chapter 1 Introduction



In this chapter, we first describe our motivation in Section 1.1. A complete literature review is presented in Section 1.2. In Section 1.3, we highlight our contributions, and in the last, we give the organization of this thesis in Section 1.4.

1.1 Motivation

Recently, human action recognition [1] from videos has raised lots of attention in computer vision because of its wide applications. The objective of action recognition is to automatically identify human activities from a given video. Many applications can benefit from such algorithms in real-world scenarios, such as human-robot interaction, smart home, health care, and surveillance systems. Over the last decade, action analysis evolved from earlier hand-crafted schemes which were limited to controlled environment settings to nowadays advanced algorithms that are learned from large-scale data and can be applied to complexed daily activities. Encouraged by the huge success of deep learning for image recognition problems [2-6], several works also tried to employ deep neural networks to tackle the problem of color-based action recognition [7-10].

As for the indoor surveillance systems applied to smart home and health care environments, there are other issues needed to be concerned. Among them, privacy issue attracts more and more attention recently. Hsu *et al.* [11] proposed a privacy free indoor action detection system using only depth videos which are less privacy sensitive. Moreover, compared with color sensors, depth cameras offer several advantages such as working under varying illumination conditions and being invariant to different colors and textures. Given such considerations, in this thesis we only adopt depth sensors to deal with the action analysis problem.

With the high popularity of depth sensors, several methods [12-14] had been



(a) Human orientations



(b) Side-view camera setting



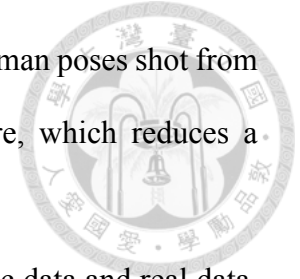
(c) Surveillance-view camera setting

Figure 1-1 Viewpoint variations in NTU RGB+D action recognition dataset [15]. Note that (b) and (c) show the difference between camera settings.

proposed to address depth-based action recognition in the last few years. Among different types of feature representations, silhouettes and spatio-temporal interest points are the most commonly used ones. However, despite the impressive results achieved by these approaches, their performances drop sharply when the viewpoint changes. This is because these features are view-dependent and human shapes significantly vary due to different human orientations or camera settings (including height and distance). Figure 1-1 illustrates some scenarios. The same action may look quite different when observed from different viewpoints, which limits the applicability of these methods when the recognition is performed from unseen views. Designing robust feature representations for video sequences is an important task. In this thesis, we propose a View-Invariant Pose (VIP) feature representation which is robust to viewpoint variations.

Besides, we have noticed a coming trend, that instead of manually collecting large-scale datasets, training on synthetic data could also bring competitive results while evaluating on real data [16-19]. This can bring lots of benefits in many cases. For instance, depth estimation based on color images usually requires a large quantity of accurately labeled data which however is impractical to be annotated manually. Varol *et al.* [18] automatically synthesized training images with ground truth depth maps by integrating human models with CMU motion capture data [20] and rendering with simulators. In this

thesis, we synthesize a large-scale depth image dataset containing human poses shot from multiple viewpoints to help us subsequently learn the VIP feature, which reduces a significant amount of effort in collecting and annotating real data.



However, there is an inevitable visual gap between the synthetic data and real data. The learned features from two different datasets reside in different domains. Thus, the so-called domain adaptation [21] plays a crucial role to bridge the gap by mapping two domains into a common space. In this thesis, we develop a strategy to learn a domain-invariant feature representation by the idea of adversarial training [22].

Recurrent Neural Network (RNN) has demonstrated its superior capability of modeling complex temporal dynamics in sequence-learning problems [23]. Given that an action can be seen as a sequence of poses, we adopt Long Short-Term Memory (LSTM) [24] to describe the temporal dependencies by feeding pose-related features sequentially.

In summary, we propose a learning-based action recognition system which takes depth videos as input to identify human activities from different viewpoints. The cross-view knowledge is learned from synthetic data in an unsupervised way and is also transferred to real data through adversarial domain adaptation.

1.2 Literature Review

We first give an introduction to action recognition using deep neural networks in Section 1.2.1, followed by several works tackling cross-view action recognition problem in Section 1.2.2. In Section 1.2.3, we introduce the domain adaptation and some related works.

1.2.1 Action Recognition with Deep Neural Networks

Due to the impressive results achieved by deep learning on image classification [2, 3], image segmentation [5], object detection [4, 6], etc., so far there have been several

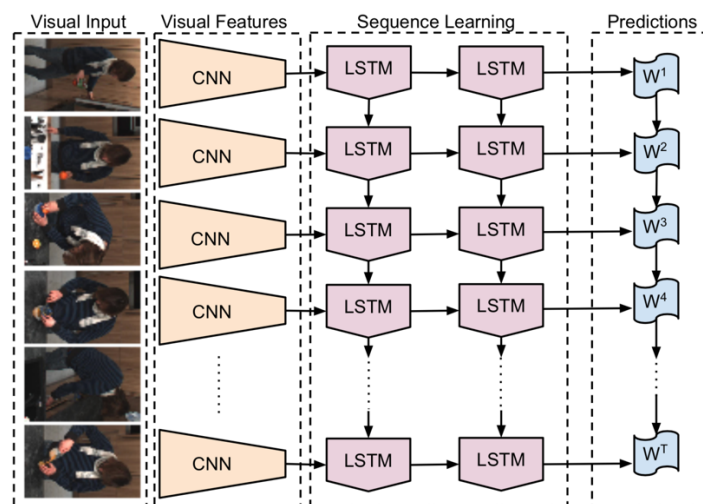


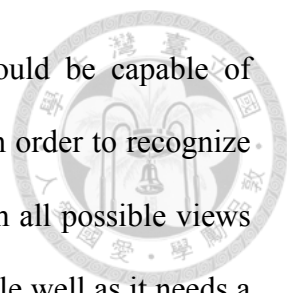
Figure 1-2 Long-term recurrent convolutional network proposed in [10].

works which adopted deep neural networks to learn spatial and temporal information for action recognition [7-10].

Ji *et al.* [7] extended the traditional 2D Convolutional Neural Networks (CNNs) to 3D CNNs where the temporal dimension is involved in convolutions. Tran *et al.* [8] designed a Convolutional 3D (C3D) architecture and found $3 \times 3 \times 3$ the best kernel size to extract spatio-temporal feature. On the other hand, instead of extracting features with a single CNN, Simonyan and Zisserman [9] trained two CNNs, one for RGB image and the other one for optical flow, and combined the learned information with late fusion. Donahue *et al.* [10] proposed an end-to-end recurrent convolutional network which processes the input video frame with a CNN and learns the temporal information with a Recurrent Neural Network (RNN) by feeding in the CNN features sequentially (see Figure 1-2). However, these methods are not designed for cross-view action recognition as they cannot be generalized to different viewpoints.

1.2.2 Cross-View Action Recognition

Most existing action recognition methods have mainly focused on videos captured from a fixed viewpoint. However, the same human action may appear quite differently



when observed from different viewpoints. A practical system should be capable of recognizing human activities from different or unseen viewpoints. In order to recognize actions across various views, a direct solution is to collect data from all possible views and train a separate model for each view. This approach does not scale well as it needs a large amount of labeled data for each view and this is infeasible as the number of action types increases.

To address this issue, knowledge-transfer based methods [16, 25-33] become popular recently. They tried to find a view-invariant latent space where the learned features could be compatible among all different views. Rahmani *et al.* [27] designed a hand-crafted feature called Histogram of Oriented Principal Components (HOPC), which is robust to viewpoint variations, to detect and describe spatio-temporal interest points. Zheng and Jiang [31] built a transferable dictionary pair by forcing the videos of the same action from different views to have similar sparse representations. This method needs video-level feature-to-feature correspondence across different views, thereby limiting the scalability. Zhang *et al.* [30] assumed that there exists a smooth virtual path between two viewpoints and connected cross-view action descriptors by applying an infinite sequence of linear transformations on view-dependent features. Although this method can operate in the absence of paired features between source and target views, they still require some samples from target view for training. Wang *et al.* [28] proposed a cross-view action representation by expressing the appearance and motion variations with a hierarchical compositional tree structure. They learned a separate linear transformation for each body part and used samples from training views to interpolate unseen views. Even though this method can recognize actions from unseen views, it requires 3D skeleton data while training which is not always available.

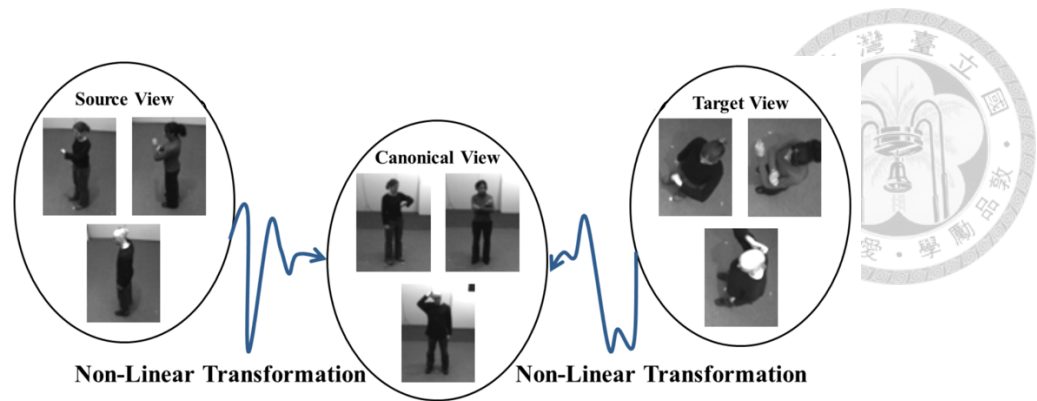


Figure 1-3 Transferring source view and target view into a predefined canonical view through non-linear transformations [26].

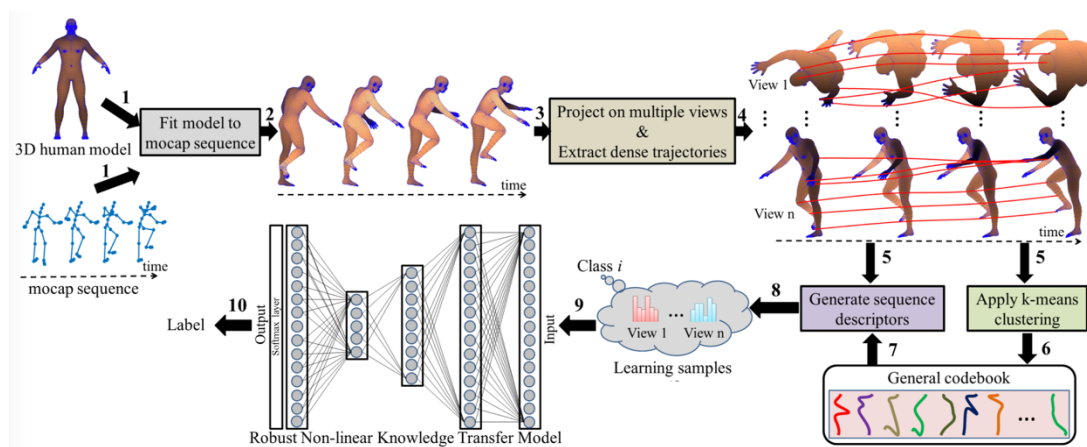


Figure 1-4 The framework of Robust Non-linear Knowledge Transfer Model (R-NKTM) proposed in [25].

Some methods relied on motion capture data to learn cross-view features. For example, Gupta *et al.* [29] used non-linear Circulant Temporal Encoding (nCTE) to find the best match for each training video in a large mocap database and synthesized multi-view data for augmentation. Rahmani and Mian [26] proposed an Non-linear Knowledge Transfer Model (NKTM) such that knowledge from multiple views is transferred to a single predefined canonical view (see Figure 1-3). Rahmani *et al.* [25] further extended NKTM to Robust Non-linear Knowledge Transfer Model (R-NKTM) as shown in Figure 1-4 that removes the need for pre-defining a canonical view which is actually action-dependent. However, these methods cannot be applied to depth videos as they depend on dense trajectories which are not reliable in depth maps.

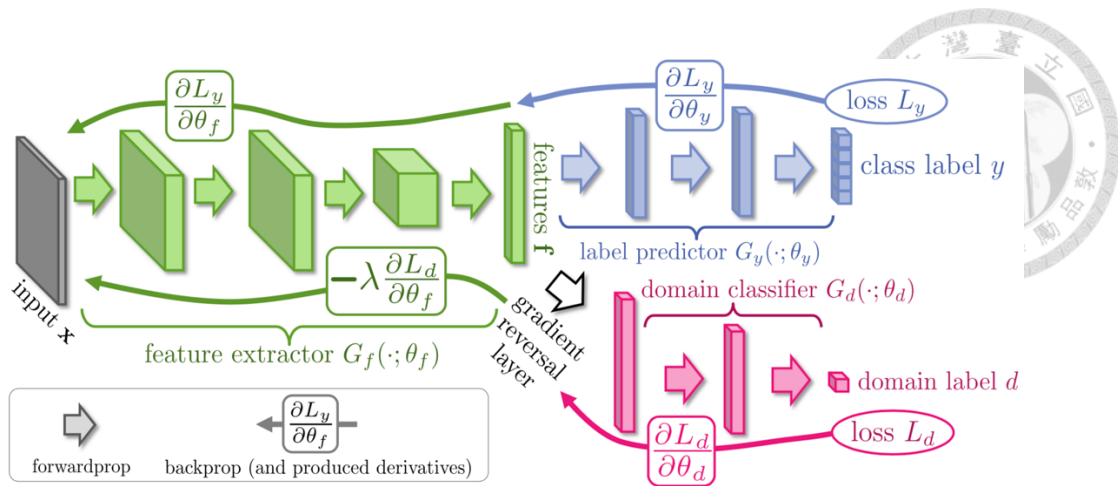


Figure 1-5 Domain adaptation using gradient reversal layer [35].

1.2.3 Domain Adaptation

Several works [16, 18, 19, 25, 26] synthesized data for training and applied the learned knowledge to real data. However, there may exist a domain shift as the synthetic data are generated or sampled within a different distribution from real data. As a result, domain adaptation [21] plays an essential role as it bridges the gap by moving two domains toward a shared space. Most previous methods worked on a fixed feature representation. Recently, there is a trend to combine feature learning and domain adaptation into a unified training process [34].

In addition to the main task, Ganin and Lempitsky [35] added a domain classifier as shown in Figure 1-5 to tell the domain from which the data came. They further proposed a gradient reversal layer to jointly learn the feature representation and align two domains by backpropagation. Other methods have integrated the adversarial training [22] and chosen an adversarial loss to minimize the domain shift by learning a feature representation that is not distinguishable between domains while discriminative of source labels. Tzeng *et al.* [36] designed a domain confusion loss to encourage the prediction of domain classifier to be a uniform distribution over domain labels. Chen *et al.* [19] synthesized training images for 3D human pose estimation based on 2D color images.

They added a domain mixer along with a pose regressor and showed that such a domain adaptation technique could significantly improve the main task on target data.



1.3 Contributions

In this thesis, we propose a learning-based action recognition system which takes depth video as input to identify human activities from different viewpoints. To our best knowledge, the presented work is the earliest one incorporating domain adaptation to address action recognition problem by learning from synthetic data. The contributions of this work are listed as follows:

- I. We propose a simple but efficient pipeline to synthesize a large-scale Multi-View Pose (MVP) dataset. It consists of paired depth images containing human poses captured from multiple viewpoints.
- II. We design a framework to learn a View-Invariant Pose (VIP) feature representation from the synthetic MVP dataset in an unsupervised way. The VIP feature encodes human poses observed from different views into a shared view-invariant feature space, thus benefits recognizing human activities from different views.
- III. We transfer the learned VIP knowledge from synthetic data to real data through adversarial domain adaptation.
- IV. We demonstrate that the proposed method can efficiently tackle the problem of cross-view action recognition while significantly reducing the amount of manual effort in collecting and annotating multi-view real data.

1.4 Thesis Organization

This thesis is organized as follows. In Chapter 2, we build up some prerequisite knowledge related to this work. In Chapter 3, we firstly describe how to synthesize a

dataset using simulators. Then, we present the framework of unsupervised learning for a pose feature, which is invariant to viewpoint variations, from the synthetic dataset. In addition, we describe how to transfer the learned knowledge from synthetic data to real data. Modeling temporal information is included in the end. In Chapter 4, the experimental results consolidate the effectiveness of our proposed method. This thesis is concluded in Chapter 5 with some future works.

Chapter 2 Preliminaries

In this chapter, some prerequisite knowledge is introduced. Firstly, we describe about cluster analysis. Secondly, we briefly give the background of deep neural networks which includes the convolutional neural network and the recurrent neural network. Thirdly, concepts about generative adversarial network and domain adaptation are presented.



2.1 Cluster Analysis and HDBSCAN

Cluster analysis is the task of grouping a set of data in such a way that data belong to the same group (or called cluster) are more similar to each other while distinct to those in other groups (or clusters). In most cases, the data points live in a high-dimension space, and the similarity is defined by different distance measurements.

There is a diversity of clustering algorithms including hierarchical clustering, centroid-based clustering, distribution-based clustering, and density-based clustering. Choosing an appropriate algorithm and parameter settings like which distance function to use or the expected number of clusters depends on the individual dataset. Usually we cluster data in an unsupervised way such that we do not use any label information. Figure 2-1 shows a clustering example.

The most popular clustering algorithm is k-means because of its simplicity and intuitiveness. However, k-means does not always find the best result due to the fact that

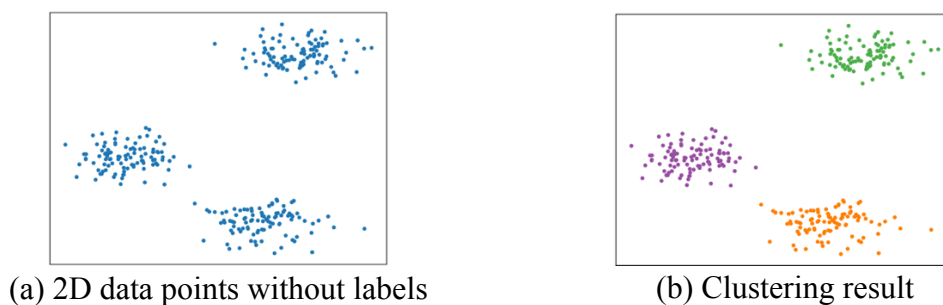
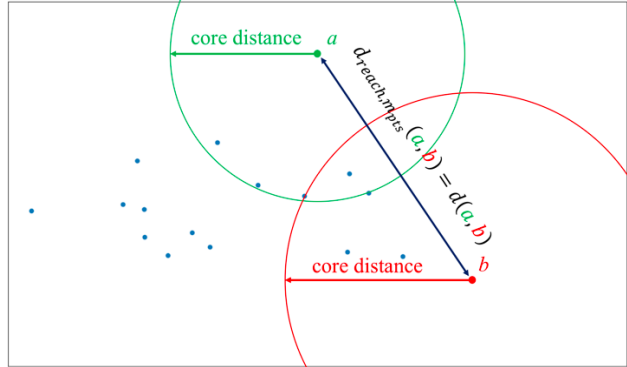
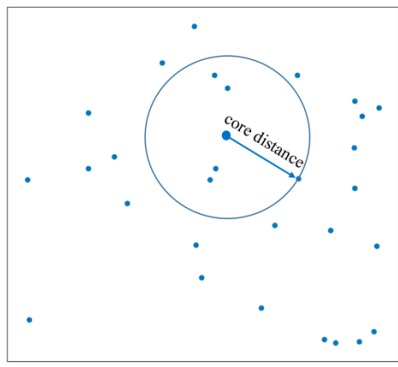


Figure 2-1 A clustering example containing 2D data points from 3 clusters.



(a) Core distance of a point (b) Mutual reachability distance between two points

Figure 2-2 Core distance and mutual reachability distance where $m_{pts} = 5$.

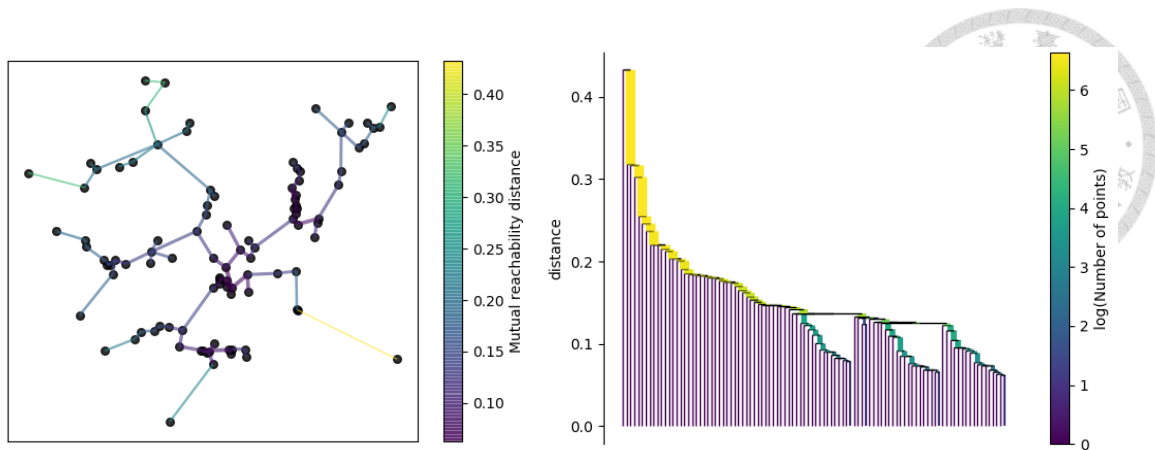
it is extremely sensitive to the k value which means the number of clusters, and it is also affected by noisy data and the initialization.

In density-based clustering, clusters are defined as areas of higher density separated by areas of lower density. Those sparse areas are usually considered as noise or border points. The most popular algorithm is Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [37]. The found clusters can be any shape, as opposed to k -means which assumes that clusters are convex shaped.

Moreover, Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [38] is extended from DBSCAN by transforming into a hierarchical clustering algorithm, and it integrates the clustering results over the varying threshold value ϵ . HDBSCAN algorithm consists of the following steps.

First, it identifies the dense and sparse regions. It estimates density by defining $d_{core,m_{pts}}(a)$ as core distance meaning the distance to the m_{pts} -th nearest neighbor of a data point a . Then it further defines mutual reachability distance $d_{reach,m_{pts}}(a,b)$ between data points a and b as follows:

$$d_{reach,m_{pts}}(a,b) = \max \{ d_{core,m_{pts}}(a), d_{core,m_{pts}}(b), d(a,b) \} \quad (2.1)$$



(a) Minimum spanning tree

(b) Cluster hierarchy

Figure 2-3 Minimum spanning tree and cluster hierarchy constructed by HDBSCAN [38].

where $d(a, b)$ is the original distance metric between a and b . Under this mutual reachability distance, the dense points (with low core distance) remain the same distance from each other while those sparser (or noisy) points are pushed away to be at least their core distance from any other point. Figure 2-2 illustrates an example.

Secondly, it considers the whole dataset as a weighted graph described by points as vertices and edges between any two points with weight equal to the mutual reachability distance. To speed up the algorithm, it builds a minimum spanning tree (MST) via Prim's algorithm, resulting a minimal set of edges such that dropping any edge causes a disconnection of components, as shown in Figure 2-3 (a).

Thirdly, it converts the MST into a hierarchy of connected components. It starts to drop edges with weights higher than a threshold ϵ to split the graph into connected components. By varying the threshold ϵ from high value to low value, it discovers a hierarchy of connected components (from completely connected to completely disconnected). Figure 2-3 (b) shows an example of hierarchical result.

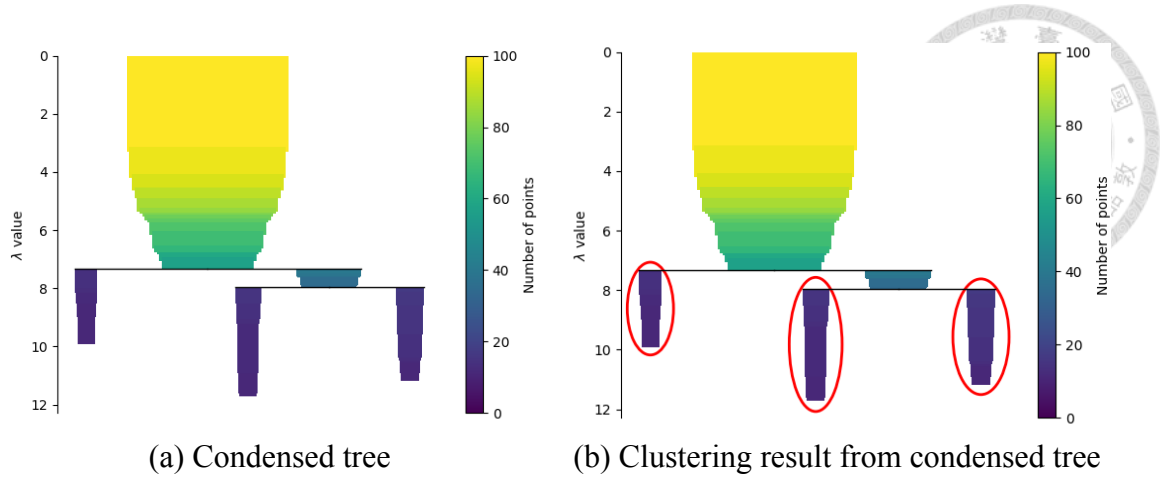


Figure 2-4 A condensed tree from the cluster hierarchy and the clustering result. The widths of lines represent the number of points in each cluster.

The next step is to condense down the large and complicated cluster hierarchy into a smaller tree with more data attached to each node. Let us denote m_{clSize} as the minimum cluster size. Walk through the hierarchy and check at each split if one of the new clusters created by the split has fewer points than m_{clSize} . If it does, discard those points as noisy data. Figure 2-4 (a) shows an example of a condensed tree with a small number of nodes.

The last step is to extract the flat clustering result. HDBSCAN combines all possible results from DBSCAN with respect to a given value of m_{pts} and all density levels $\lambda = 1/\epsilon$ in $[0, \infty)$. Gradually increasing λ (decreasing ϵ) and assume cluster \mathbf{C}_i appears at level $\lambda_{min}(\mathbf{C}_i)$. It defines the stability of a cluster \mathbf{C}_i as:

$$S(\mathbf{C}_i) = \sum_{a \in \mathbf{C}_i} (\lambda_{max}(a, \mathbf{C}_i) - \lambda_{min}(\mathbf{C}_i)) \quad (2.2)$$

where $\lambda_{max}(a, \mathbf{C}_i)$ is the density level beyond which data point a no longer belongs to cluster \mathbf{C}_i . Let $\{\mathbf{C}_2, \dots, \mathbf{C}_m\}$ be the collection of all clusters except the root \mathbf{C}_1 in the condensed tree as shown in Figure 2-4 (a). The final clustering result can be seen as a flat and non-overlapping partition by solving the following optimization problem of maximizing the sum of stabilities of the selected clusters.

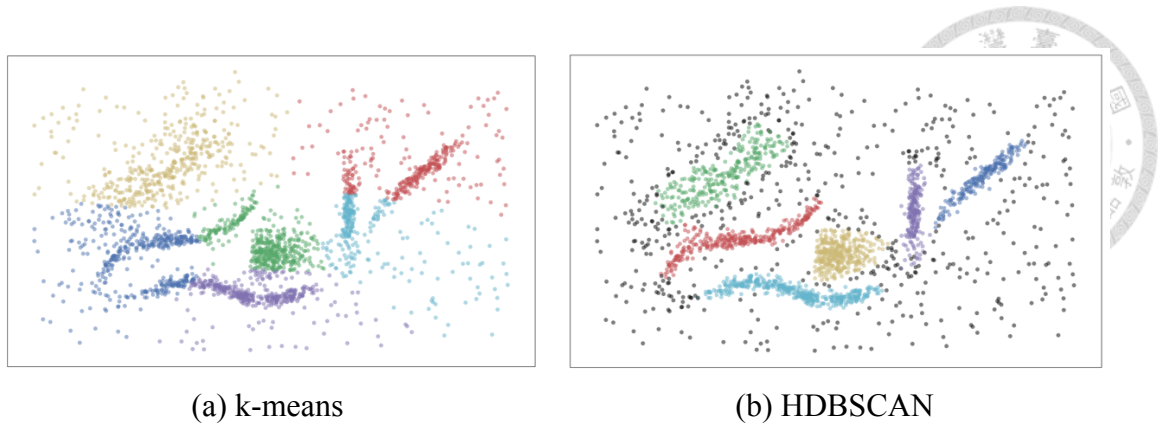


Figure 2-5 A comparison of clustering results between k-means and HDBSCAN. The gray points in (b) are considered as noise.

$$\begin{aligned}
 \max_{\delta_2, \dots, \delta_m} J &= \sum_{i=2}^m \delta_i S(\mathbf{C}_i) \\
 \text{subject to } &\begin{cases} \delta_i \in \{0, 1\}, & i = 2, \dots, m \\ \sum_{j \in J_l} \delta_j = 1, & \forall l \in \mathbf{L} \end{cases}
 \end{aligned} \tag{2.3}$$

where δ_i indicates whether cluster \mathbf{C}_i is selected in the solution, $\mathbf{L} = \{l | \mathbf{C}_l \text{ is leaf cluster}\}$ denotes the indexes of leaf clusters, and $J_l = \{j | j \neq 1 \text{ and } \mathbf{C}_j \text{ is ascendant of } \mathbf{C}_l\}$ is the set of indexes of all clusters on the path from the root to \mathbf{C}_l . Figure 2-4 (b) illustrates an example of the optimization result. Besides, as a comparison between k-means and HDBSCAN shown in Figure 2-5, HDBSCAN can discard the noise from clustering result while k-means assigns each point including noise to a cluster.

As a density-based clustering algorithm with few assumptions about data distribution and a small number of intuitive parameters, HDBSCAN is ideally suitable for exploratory data analysis. We utilize HDBSCAN to build a pose dictionary as described in Section 3.1.1.

2.2 Convolutional Neural Network

In recent years, Convolutional Neural Networks (CNNs) have brought an

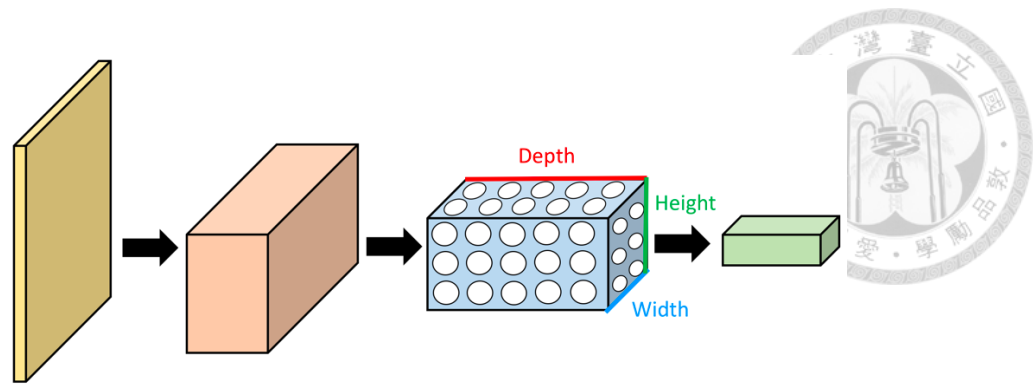


Figure 2-6 A common CNN architecture comprised of several convolutional layers. Every layer transforms the 3D input volume to a 3D output volume of neurons.

overwhelming success in the computer vision field, including image classification [2, 3], image semantic segmentation [5], object detection [4, 6], *etc.*

Take the classification task for example, traditional frameworks consist of hand-crafted feature extractors like HOG feature, and learnable classifiers such as Support Vector Machine (SVM). While the classifiers can learn by themselves to solve the optimization problems, usually we need to predefine the parameters of the feature extractors and it requires some domain knowledge and several trial and error for tuning. On the other hand, CNN-based frameworks are comprised of learnable feature extractors and learnable classifiers. Not only classifiers but also feature extractors can automatically tune their parameters by backpropagation in an end-to-end way, which increases the learning ability of the feature representations.

CNNs are similar to traditional neural networks, where the convolution kernels are made up of learnable weights and biases, just like neurons. Each kernel receives some inputs, performs a dot product and is optionally followed by a non-linearity activation function. Figure 2-6 shows a common CNN architecture. In general, earlier layers extract low-level features, such as edge and corner, while following layers are responsible for high-level features, such as meaningful structures like objects, animals, human faces, *etc.*

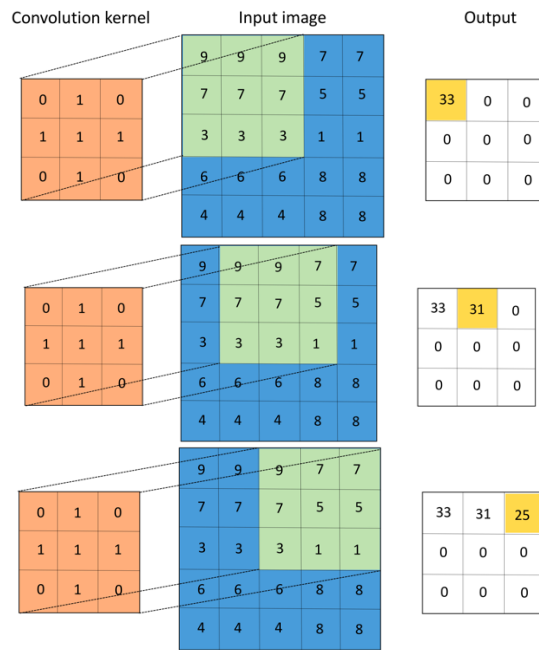
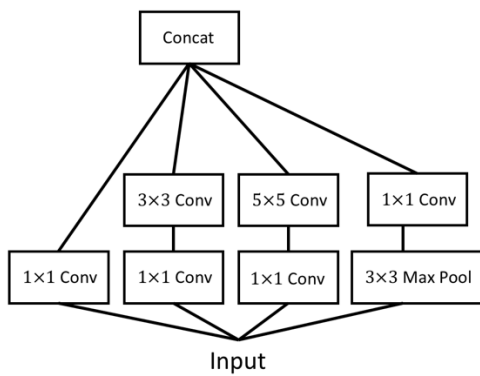


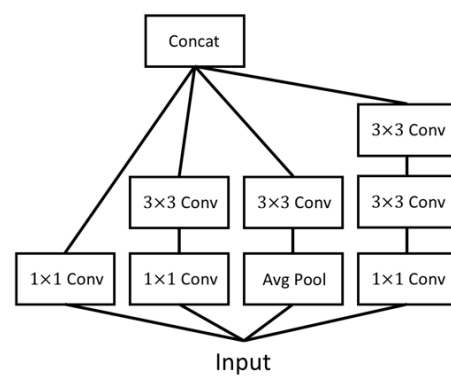
Figure 2-7 Forward pass of a 3×3 convolutional operation with stride equal to 1.

2.2.1 Convolutional Layer

Each convolutional layer has multiple convolution kernels which perform a dot product with a specific input patch called receptive field. Each layer can have different number of kernels, which makes layers' outputs differ in depth (or channel) dimension. Sometimes zero-padding around the border of the input makes the size of output feature map consistent with the input. During the forward pass, the convolution is conducted along the height and width dimensions with a pre-specified stride as illustrated in Figure 2-7. The weights of each convolution kernel can be randomly initialized when trained from scratch. During training, the weights are updated through backpropagation with gradients of loss function.



(a) Inception module V1 [39]



(b) Inception module V3 [40]

Figure 2-8 The family of Inception modules.

Different CNNs may end up with different number of layers, different kernel size, and different connectivity configurations. In the beginning, people tried to make the structure deeper with more powerful representations. However, considering the fact that models are hard to trained as going deeper, recent researchers focus on how to efficiently expand the model capacity while maintaining or reducing the number of parameters. As depicted in Figure 2-8, Google designed a family of “Inception module” [39, 40] which not only deeper the structure but also makes it wider. In the following, we will introduce one of them used in this thesis called Xception network.

2.2.2 Xception Network

To expand the capacity of Inception modules, Chollet [41] proposed a novel CNN structure called Xception which stands for “Extreme Inception” by replacing Inception modules with depth-wise separable convolutions.

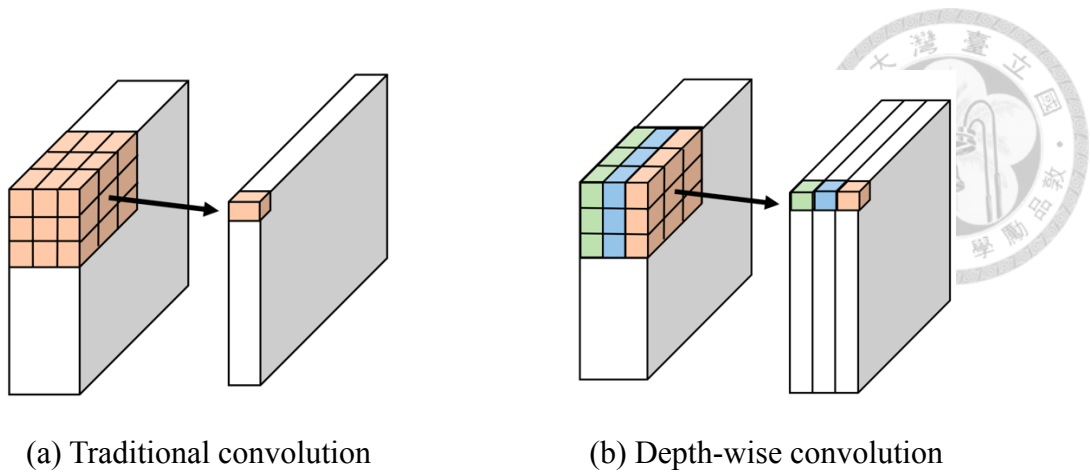


Figure 2-9 The difference between traditional and depth-wise convolution using 3×3 convolution kernels.

A traditional convolution kernel is tasked with simultaneously mapping spatial and cross-depth correlations. On the other hand, the depth-wise separable convolution, commonly called “separable convolution”, tries to decouple these two correlations by independently performing spatial convolution over each channel of input followed by a pointwise (1×1) convolution. Figure 2-9 illustrates the difference between the traditional and depth-wise convolution.

The Xception network, as shown in Figure 2-10, has 36 convolutional layers formed into 14 modules, all of which are equipped with residual connections [3]. Such a design slightly outperforms Inception V3 [40] on the ImageNet dataset and significantly outperforms Inception V3 on JFT dataset while having roughly the same number of parameters. In this thesis, we use Xception network as pose feature extractor as described in Section 3.2.1.

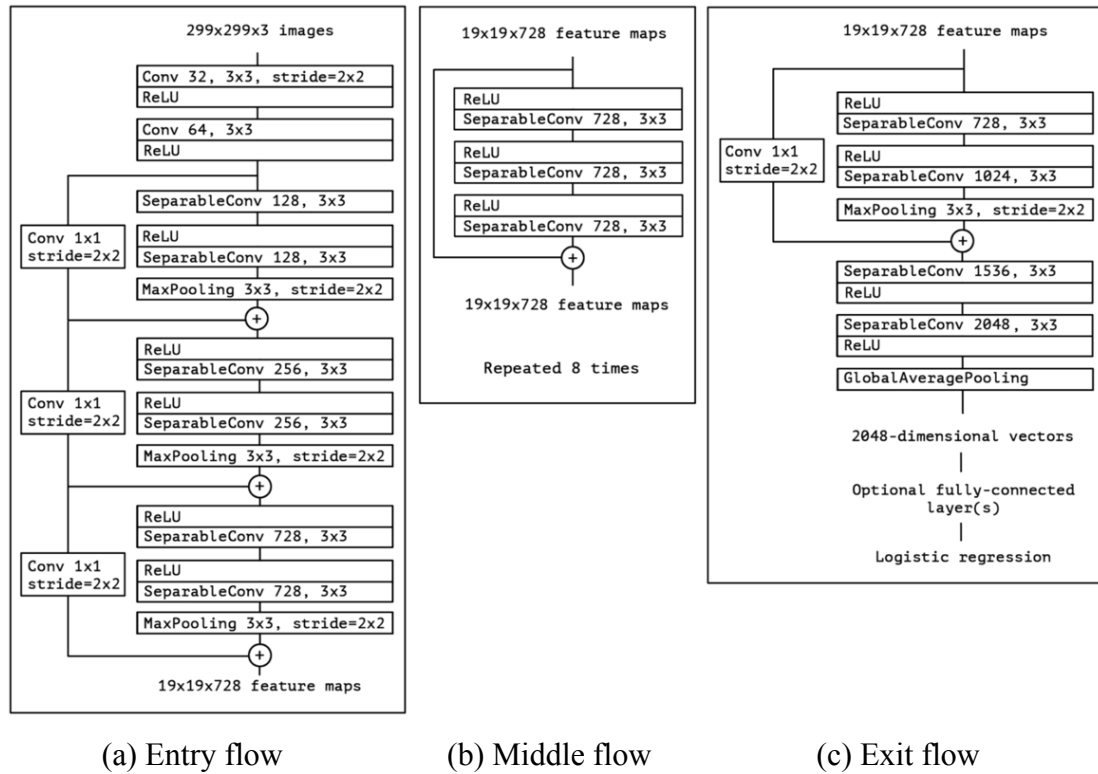


Figure 2-10 The Xception network architecture [41].

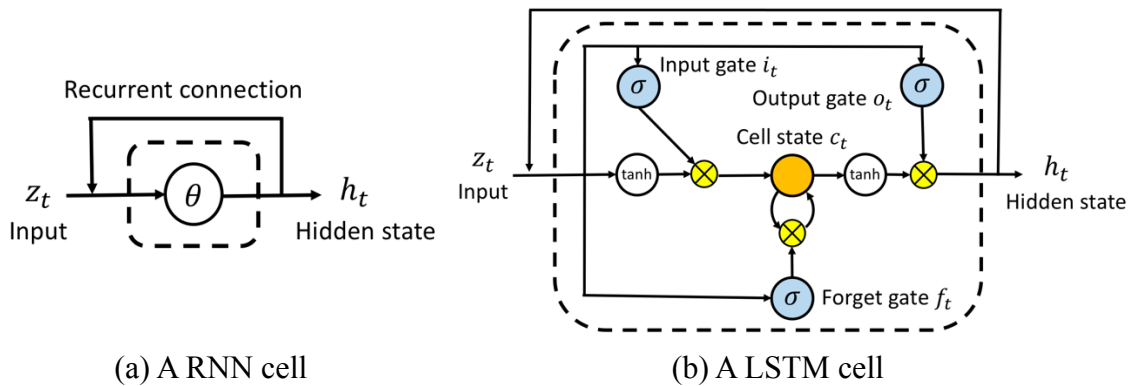


Figure 2-11 Structures of the neurons in Recurrent Neural Networks (RNNs).

2.3 Recurrent Neural Network and Long Short-Term Memory

In this section, we briefly introduce the concept about Recurrent Neural Network (RNN) to make this thesis self-contained.

RNN is a popular method for extracting features and modeling sequential or

temporal data. The main difference between a RNN and a standard feedforward network is the feedback loop, which makes a recurrent connection in an unfolded network. With a self-connected cell as shown in Figure 2-11 (a), RNN is capable of modeling the contextual information from a sequence of length T by the following equation:

$$h_t = \theta(W[h_{t-1}, z_t] + b) \quad (2.4)$$

where z_t and h_t denote the input and hidden state at time step t respectively, W and b represent learnable weights and bias, and θ is the non-linear activation function. Each hidden state h_t is determined by the current input z_t and the previous hidden state h_{t-1} . Theoretically, the last hidden state h_T contains the information about the whole sequence.

However, it is hard to train RNNs due to vanishing gradient and error blowing up problems, especially when the sequential data becomes longer. Long Short-Term Memory (LSTM) [24] remedies this issue by introducing gating mechanism to determine when the input is significant enough to remember, when it should forget information, and when it should output the value. It works tremendously well on a large variety of problems.

As shown in Figure 2-11 (b), a LSTM cell has three gates, including forget gate, input gate, and output gate. In addition, different from the standard RNN, LSTM maintains a cell state c_t along with the hidden state h_t for each time step t . With such a design, it can capture long-term information through the following equations:

$$f_t = \sigma(W_f[h_{t-1}, z_t] + b_f) \quad (2.5)$$

$$i_t = \sigma(W_i[h_{t-1}, z_t] + b_i) \quad (2.6)$$

$$o_t = \sigma(W_o[h_{t-1}, z_t] + b_o) \quad (2.7)$$

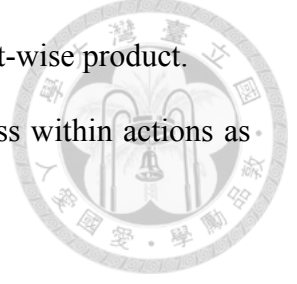
$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh(W_c[h_{t-1}, z_t] + b_c) \quad (2.8)$$

$$h_t = o_t \otimes \tanh(c_t) \quad (2.9)$$

where f , i , o correspond to forget gate, input gate, and output gate, respectively, all W

and b are learnable weights and biases, and \otimes denotes the element-wise product.

In this thesis, we utilize LSTM to capture the temporal progress within actions as described in Section 3.3.



2.4 Generative Adversarial Network

In this part, we briefly describe the Generative Adversarial Network (GAN) [22] to make the completeness of this thesis. Aiming at recovering or synthesizing signals with the similar distribution to the real data, GAN utilize two components, one is generator $G(z): \mathbb{R}^p \rightarrow \mathbb{R}^{m \times n \times 3}$ and the other one is discriminator $D(x): \mathbb{R}^{m \times n \times 3} \rightarrow \mathbb{R}$, to optimize a two-player minimax game with the following objective function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim P_{data}} [\log D(x)] + \mathbb{E}_{z \sim P_z} [\log (1 - D(G(z)))] \quad (2.10)$$

where $x \in \mathbb{R}^{h \times w \times 3}$ is an image sampled from the real data, $z \in \mathbb{R}^p$ is a randomly sampled vector from a prior distribution P_z , and $D(x)$ represents the probability that image x comes from real data.

Generator G is responsible for generating images from a random vector while discriminator D is tasked with distinguishing real images from those generated by G . Note that generator G implicitly defines a generative distribution P_g behind the samples $G(z)$ obtained from $z \sim P_g$.

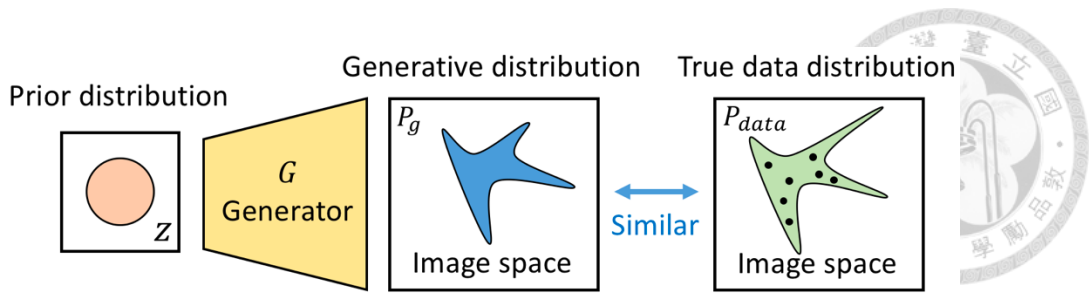


Figure 2-12 The goal of the generator in GAN [22]. The generator tries to produce images with the similar distribution to real ones.

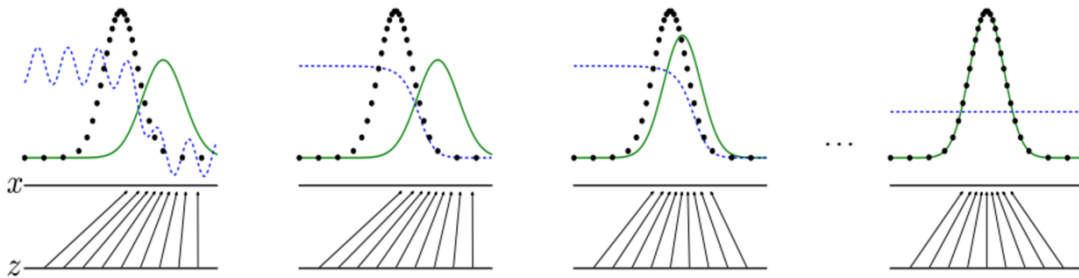


Figure 2-13 Adversarial training process in GAN [22]. GAN is trained by simultaneously updating the discriminative function (blue, dashed line) to make it tell the difference between real data distribution (black, dotted line) and the generative distribution (green, solid line).

As shown in Figure 2-12, with Eqn. (2.10) optimized, generator G learns how to produce images in $\mathbb{R}^{h \times w \times 3}$ lying with the same distribution of real data. Besides, Figure 2-13 illustrates the adversarial training process. From left to right, the generative distribution gradually fits on real data distribution and the discriminator D is unable to differentiate between the two distributions, *i.e.* $D(x) = \frac{1}{2}$.

2.5 Domain Adaptation

Due to the phenomenon called “domain shift”, machine learning models trained on some representations from one dataset do not generalize well to other datasets. In other words, models learned from the training domain (or source domain) cannot perform equally well on the testing domain (or target domain). The simplest solution is to fine-tune the learned models on the task-specific domains. However, it is hardly to obtain enough labeled data to properly fine-tune especially when the models are deep multi-layer networks with a large number of parameters.

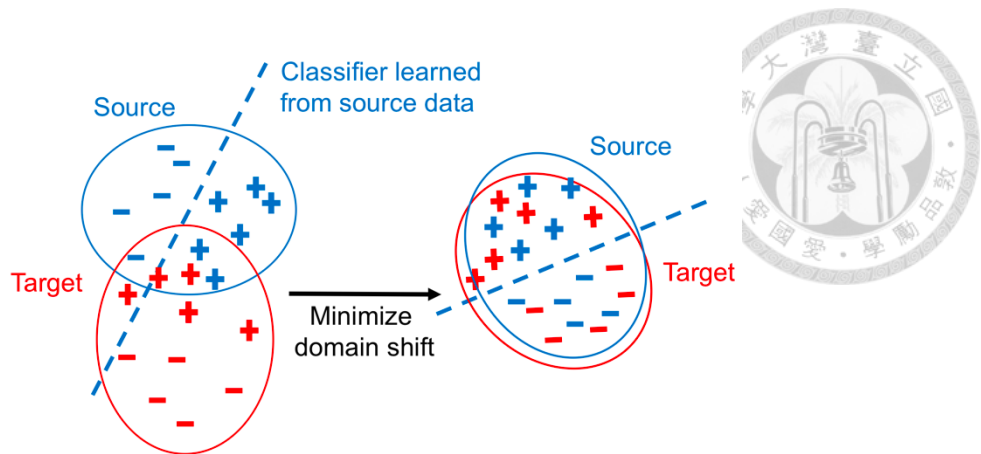


Figure 2-14 Domain adaptation attempts to minimize the domain shift.

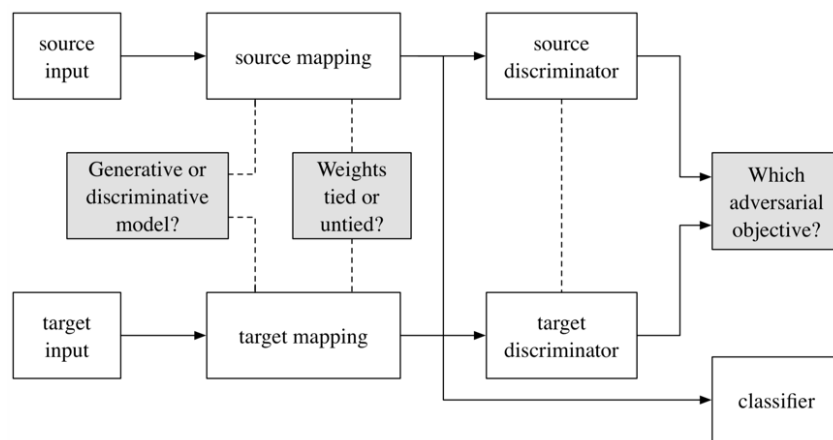


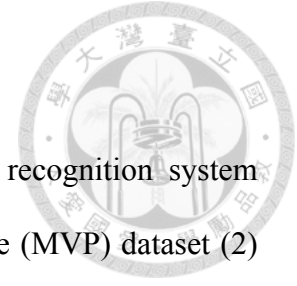
Figure 2-15 The general framework for adversarial domain adaptation proposed in [34].

Domain adaptation is a technique that tackles this problem by adapting data from the source domain into the target domain while the main task such as classification is preserved in the target domain. Among all the methods, a popular way is to reconstruct a new representation space where the data from two domains are projected. It attempts to find a domain-invariant representation which is indistinguishable from domain to domain. Additionally, it learns main-task models with projected data from the source domain. As shown in Figure 2-14, when distributions of representations from two domains become closer, the learned models from source domain perform satisfactorily on the target domain.

Moreover, recent works have focused on transferring deep neural network features from one labeled source domain to another target domain where the labeled data is sparse or non-existent, resulting semi-supervised or unsupervised domain adaptation problems.

Based on the development of GAN [22], several approaches employ adversarial training between the feature encoder (*i.e.* generator) and the domain discriminator so as to find feature space which is not only uninformative about domain but also discriminative to main task. Tzeng *et al.* [34] proposed a universal framework for adversarial domain adaptation as shown in Figure 2-15. In this thesis, we use adversarial domain adaptation to align the features extracted from real data and synthetic data as described in Section 3.2.2.

Chapter 3 Methodology



In this thesis, we propose a depth-based cross-view action recognition system consisting of three parts: (1) collecting a synthetic Multi-View Pose (MVP) dataset (2) learning a View-Invariant Pose (VIP) feature representation from synthetic dataset and transferring the knowledge to real data through domain adaptation, and (3) modeling the temporal information with LSTM.

3.1 Synthesize a Multi-View Pose Dataset

Training deep networks is an optimization problem with respect to millions of parameters, which requires large amounts of labelled data to achieve acceptable performance. In our case, however, learning a deep View-Invariant Pose (VIP) feature representation requires a large corpus of multi-view data which is very hard and expensive to manually collect and label.

Moreover, we have noticed a coming trend that training on synthetic data [16-19] could also bring competitive results while evaluating on real data, which reduces the amount of manual effort in collecting large-scale dataset. Inspired by [16], we design a simple but efficient pipeline to synthesize a Multi-View Pose (MVP) dataset which contains human poses captured from multiple viewpoints. The proposed pipeline can generate an infinite number of possible combinations of human poses, human geometries,

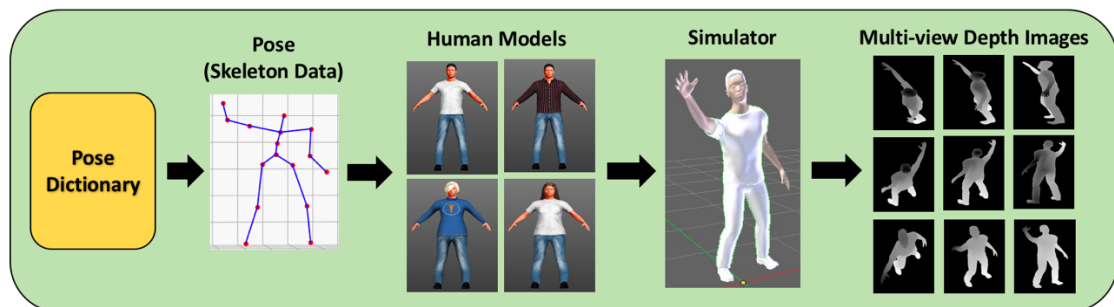


Figure 3-1 Proposed pipeline for synthesizing a Multi-View Pose (MVP) dataset.

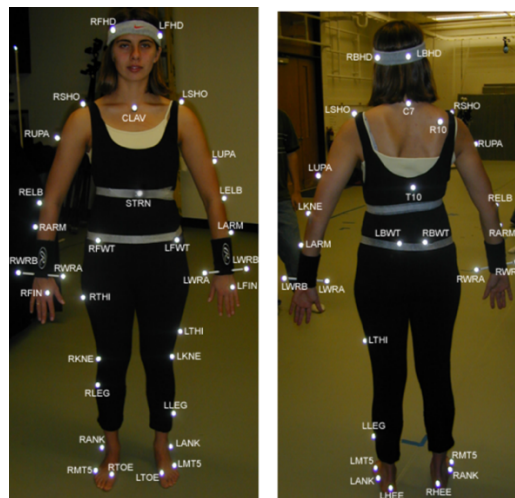


Figure 3-2 The setting of body markers in CMU motion capture database [20].

and viewpoints. Figure 3-1 gives an overview of the involved steps in collecting the synthetic MVP dataset. Firstly, we build a pose dictionary consisting of finite different poses. Secondly, we fit several human models with each pose in the dictionary. Thirdly, we use a simulator to render depth images from numerous viewpoints. The details are described in the following.

3.1.1 Build a Pose Dictionary

Given the fact that an action can be viewed as a sequence of poses, the possible pose space is much smaller than the possible action space. In addition, several bag-of-word based action recognition methods [42, 43] have discriminated actions by the occurrence of each pose-related codeword in an action. Based on these observations, learning a pose-related feature could generalize well to action space with different kinds of sequence combinations.

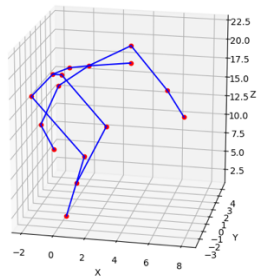


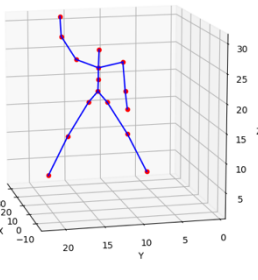


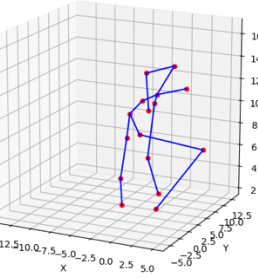
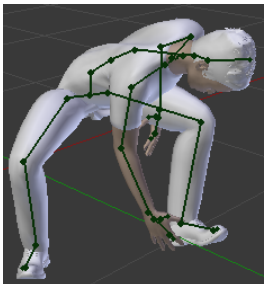

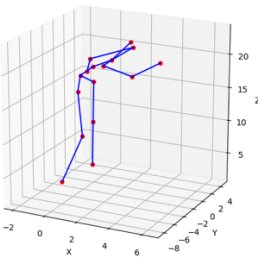


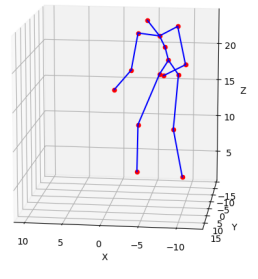
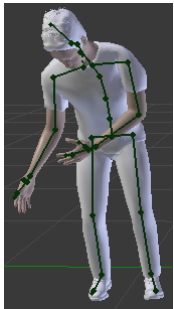

As we need to fit 3D human models with predefined poses in the simulator, we must rely on highly accurate skeleton data such as motion capture database. We choose CMU motion capture database [20] which contains more than 2500 motion sequences (over 4 million poses) covering a variety of actions. It is captured with high-precision camera

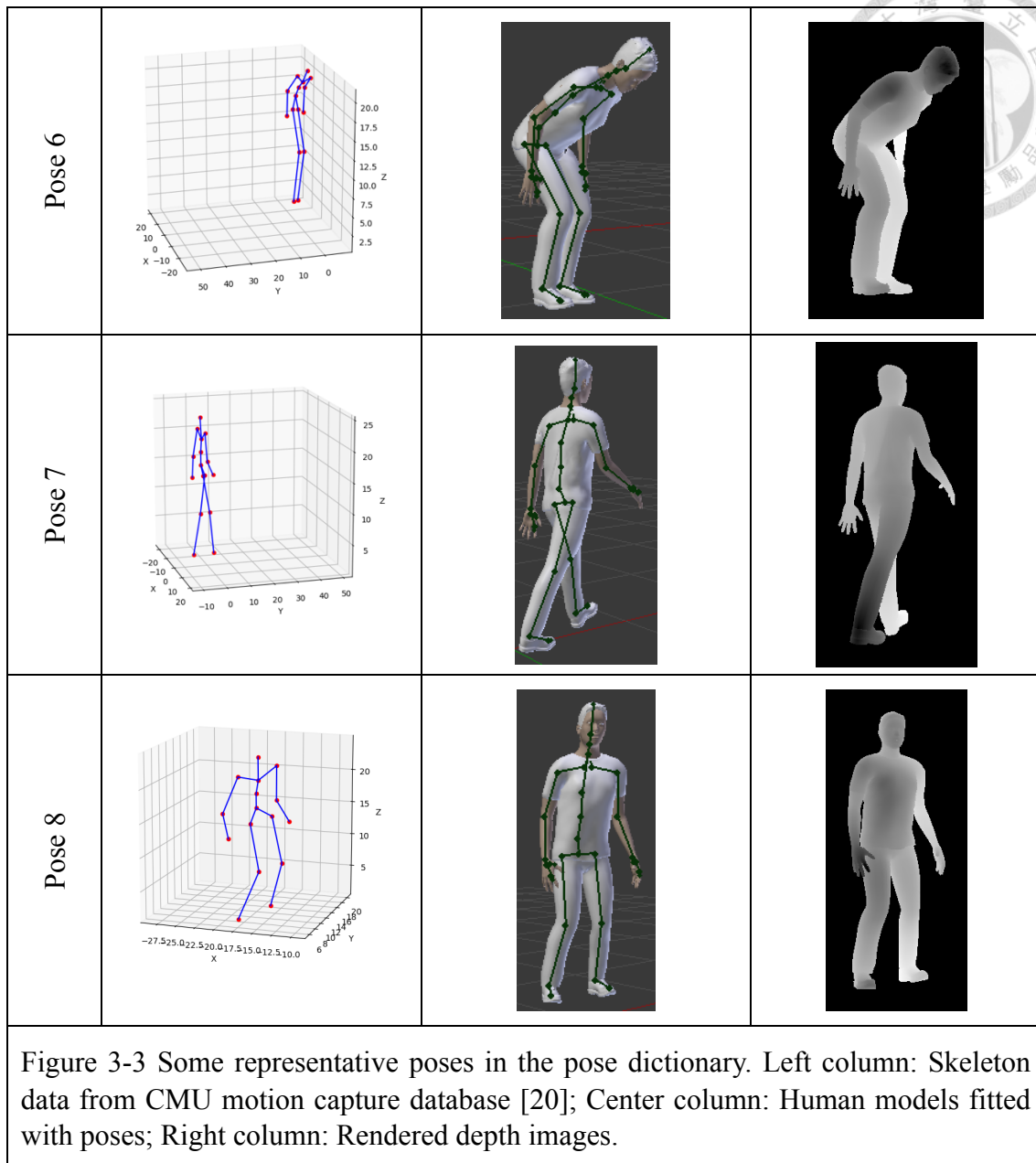
array and body markers (see Figure 3-2), resulting in quite accurate 3D skeletal joint positions. This database is treated as the pose space from which we sample poses.

In order to efficiently learn pose-related features from the pose space, we firstly build a dictionary containing the most representative poses. Unsupervised clustering algorithm is applied to the skeletons sampled from this pose space. Different from [16], HDBSCAN algorithm [38] rather than k-means is used for two reasons. First, the result from k-means is sensitive to the k value which indicates the number of clusters. Actually, we cannot determine in advance how many clusters could well represent the pose space. Second, k-means is a hard-clustering algorithm which means that it assigns each data with a cluster including the noise. HDBSCAN copes with these issues as it can automatically find the k value by tuning two intuitive parameters and discard noisy data from the result by soft clustering.

Besides, we design an orientation-based skeletal feature rather than a displacement-based skeletal feature used in [16] to compare the similarity between two poses. We calculate rotation angles between body limbs. For example, we use 3 degrees of freedoms (DOFs) to represent a shoulder's movement and 1 DOF to describe elbow's. As a result, we use 19 DOFs in total (4 for left arm, 4 for right arm, 4 for left leg, 4 for right leg, 1 for head, and 2 for torso) to describe a skeleton (or pose) and use Euclidean distance to calculate the similarity. Unlike displacement-based feature, the designed feature is not only location-invariant but also scale-invariant and it brings more physical meaning with fewer parameters.

By setting two parameters, minimum cluster size and minimum samples, we get K clusters from HDBSCAN algorithm. The pose with the highest score in each cluster is chosen as representative, thus forming a dictionary containing K representative poses as shown in Figure 3-3 (left column).

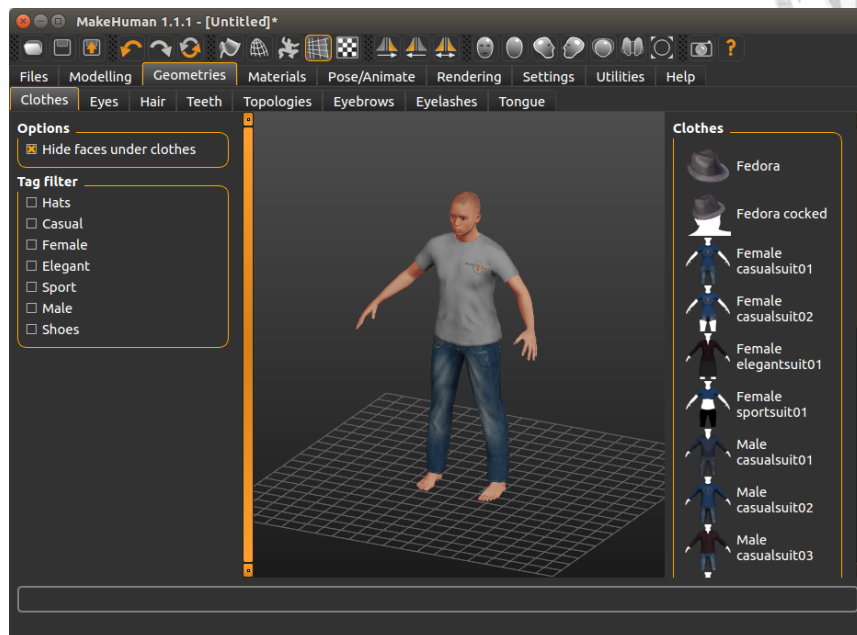
	Skeleton data	Human model	Rendered depth image
Pose 1			
Pose 2			
Pose 3			
Pose 4			
Pose 5			



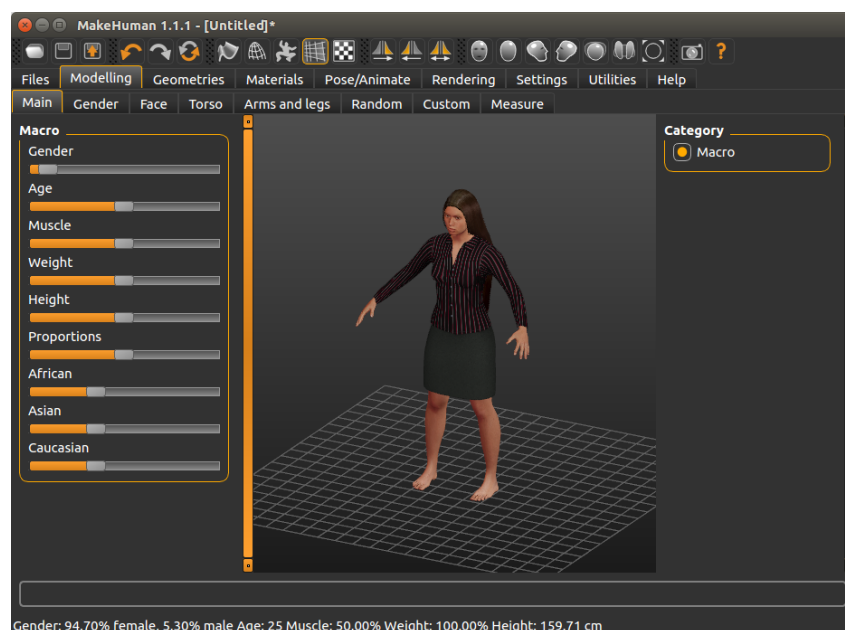
3.1.2 Create 3D Human Models

There are different ways to create 3D human models. In order to cover the variation in subjects, we utilize the open source MakeHuman software [44] to create different 3D human models provided with meta data. The created human models are able to be fit with predefined poses. With the graphical user interface (GUI) of MakeHuman shown in Figure 3-4, we could adjust the gender as well as the body figure such as height, weight, and muscle portion of human models. As depicted in Figure 3-5, we synthesize several

realistic human models with different combinations of gender, body shape, hair style, and clothes.



(a) Choosing clothing style



(b) Choosing gender and body figure

Figure 3-4 The GUI of MakeHuman [44].



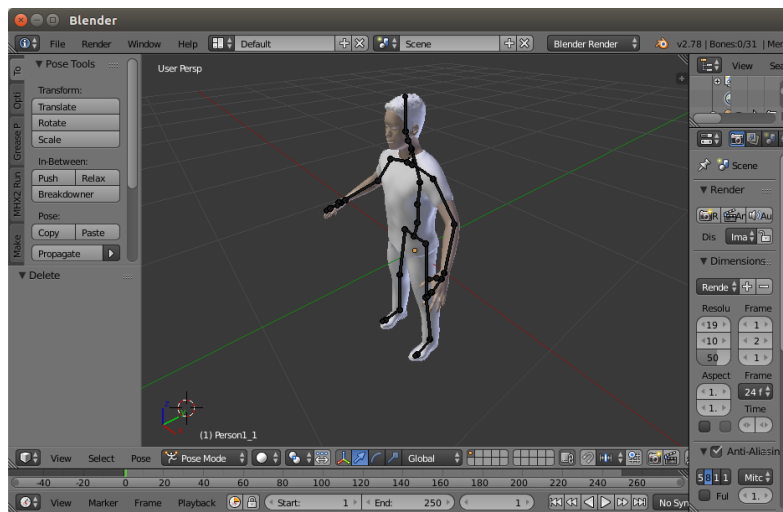
Figure 3-5 3D human models with different combinations of gender, body shape, hair style, and clothes created by MakeHuman [44].

3.1.3 Render Depth Images

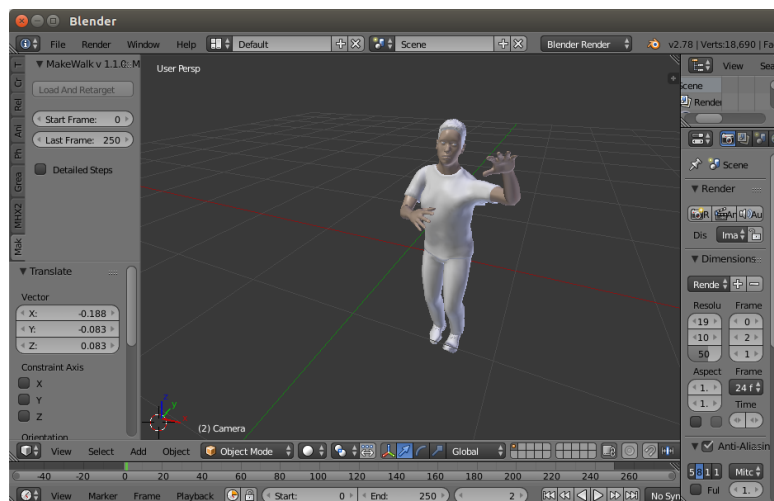
Our goal is to collect synthetic depth images of human poses captured from various viewpoints. We use the Blender software [45] which is an open source package so as to fit 3D human models into mocap data. As shown in Figure 3-6, a simulation environment is created.

Blender normalizes the mocap skeletal data with respect to the size of human models. For each 3D human model, we re-target its rigs to fit all the poses in the pose dictionary. Figure 3-6 (b) illustrates an example of a 3D human model fitted with a pose.

Moreover, in the Blender simulator, we uniformly place numerous virtual cameras with distinct latitudes and longitudes on a hemisphere surrounding the subject (see Figure 3-7) and render depth images with normalized pixel value in $[0, 255]$. This process results in synthetic but realistic depth images. Different from [16], our pipeline does not include hidden point removal and surface fitting. Thus, we synthesize a large-scale MVP dataset containing paired depth images of human poses captured from multiple viewpoints as depicted in Figure 3-8.



(a) The imported 3D human model with its rigs



(b) Re-targeting into a predefined human pose

Figure 3-6 The GUI of Blender [45].

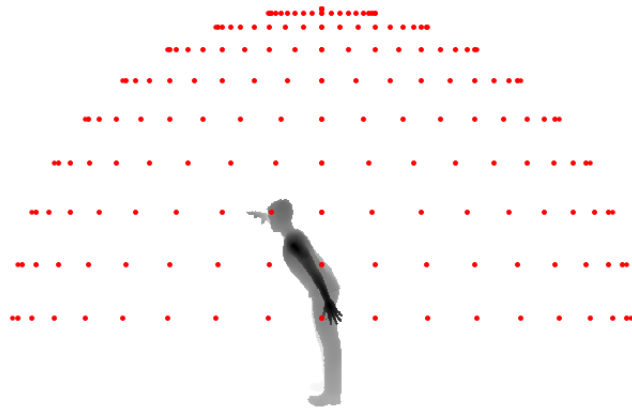
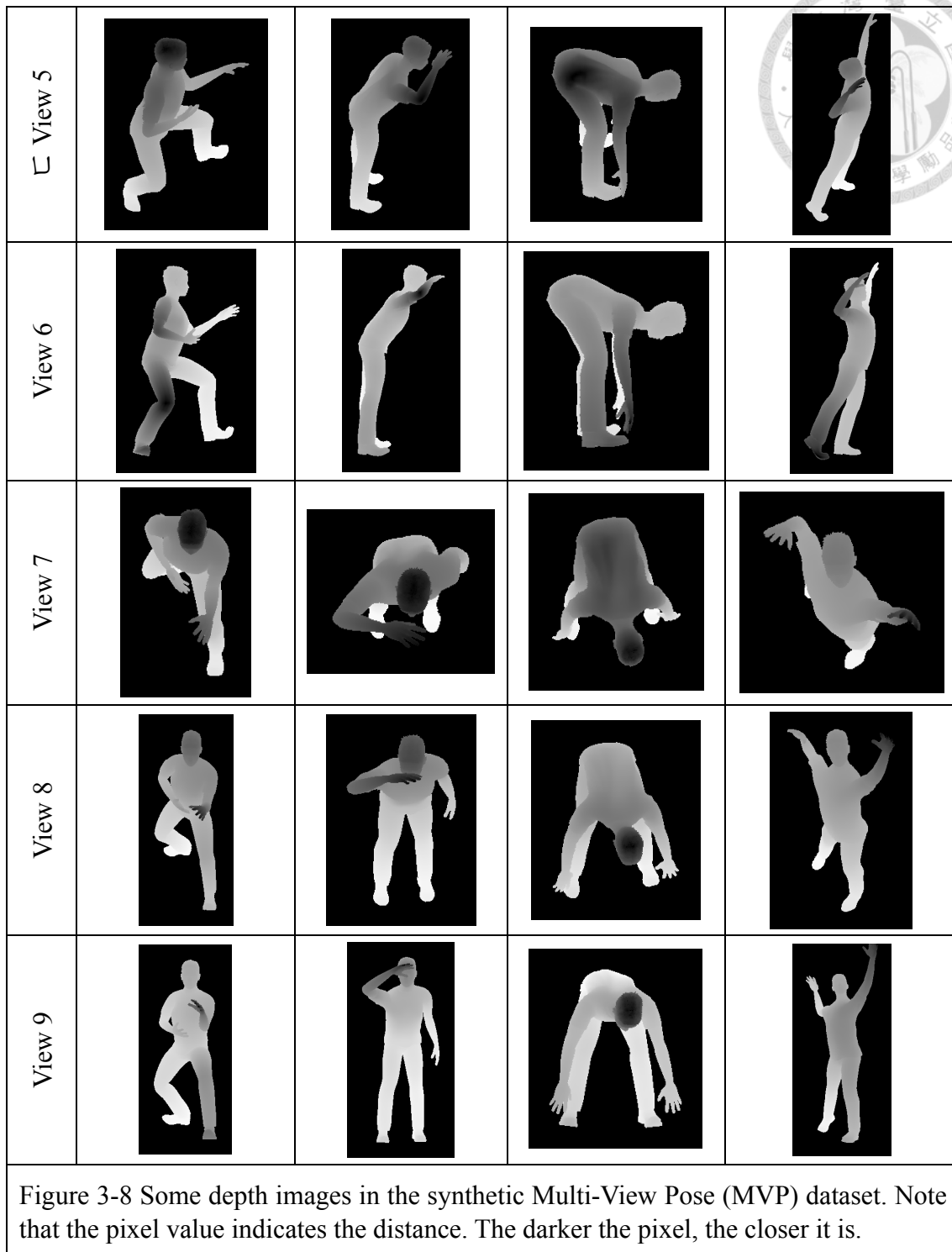


Figure 3-7 Multiple virtual cameras are uniformly placed around the subject.

	Pose 1	Pose 2	Pose 3	Pose 4
View 1				
View 2				
View 3				
View 4				



3.2 Learn a View-Invariant Pose Feature

In this section, we describe how to learn the View-Invariant Pose (VIP) representation that transfers human poses from any view to a shared high-level feature space. Moreover, the VIP knowledge is distilled from the MVP dataset and the

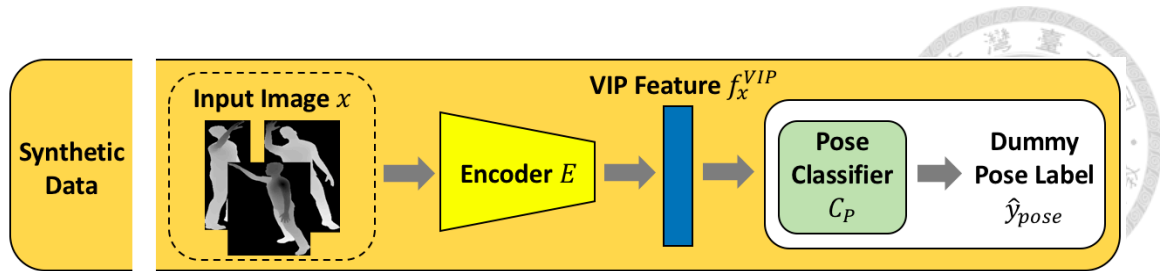


Figure 3-9 Unsupervised training architecture for learning View-Invariant Pose (VIP) feature from synthetic data.

information of viewpoint is not required during the learning process.

3.2.1 Unsupervised Learning

As depicted in Figure 3-9, the proposed architecture for learning VIP feature consists of two parts, one being an encoder E and the other one being a pose classifier C_P . Here, an underlying assumption is that the same human pose observed from different viewpoints share the same high-level feature representation, which we believe should be quite reasonable. Convolutional Neural Networks (CNNs) have demonstrated their powerful capacities of extracting visual features over several image recognition problems [2-6]. We design the encoder E as a CNN-based structure which extracts pose-related feature from different views into a universal feature space and we refer the extracted feature of image x as VIP feature f_x^{VIP} . Then the pose classifier C_P aims to tell which pose appears in the image x based on f_x^{VIP} .

While training such a framework, the input images x come from the synthetic MVP dataset. Besides, we do not use any action label from mocap data but only the dummy pose label, making the training process unsupervised. For each pose $k = 1, 2, \dots, K$ in the pose dictionary, the corresponding synthetic depth images rendered from all viewpoints are assigned the same dummy pose label k . The label contains no physical meaning but the dictionary index.

For the encoder E , we utilize the “Xception network” [41] which is comprised of depth-wise separable convolutions. The last fully-connected layer is replaced with a n -

neuron layer as a bottleneck layer, resulting VIP feature f_x^{VIP} as a n -dimension vector. As for the pose classifier C_P , we design a simple one-layer fully-connected network followed by a softmax activation, consisting of K neurons corresponding to each dummy pose label.

The inputs to our training architecture are synthetic depth images $x \in synthetic$ sampled from synthetic MVP dataset with the corresponding ground truth dummy pose labels y_{pose} . With the goal of achieving view-invariant, the VIP feature f_x^{VIP} is enforced to encode only the pose information regardless of viewpoints. We consider VIP loss L_{VIP} as the standard cross-entropy of pose classification as listed below. Here, *syn* is an abbreviation for *synthetic*.

$$L_{VIP} = - \sum_{x \in syn} \sum_{k=1}^K \mathbb{I}[y_{pose} = k] \log(p_{x,k}) \quad (3.1)$$

$$p_x = C_P(f_x^{VIP}) \quad (3.2)$$

$$f_x^{VIP} = E(x) \quad (3.3)$$

where $x \in \mathbb{R}^{h \times w \times 3}$ is the input image, $f_x^{VIP} \in \mathbb{R}^n$ refers to the encoded VIP feature of image x , $p_x \in \mathbb{R}^K$ indicates the softmax activations of the pose classifier, $p_{x,k} \in \mathbb{R}$ is the k -th element of p_x , meaning the probability of image x being pose k , and $\mathbb{I}[\cdot]$ denotes Iverson bracket. Note that for a particular pose k , we sample the input images from multiple views during training. We use this architecture to find a high-level feature space shared by all possible views. To be more specific, regardless of the input view, we encourage the VIP features to be as similar as possible for all views of the same pose. The detailed architecture design and training process are presented in Section 4.2.2 and Section 4.2.3.

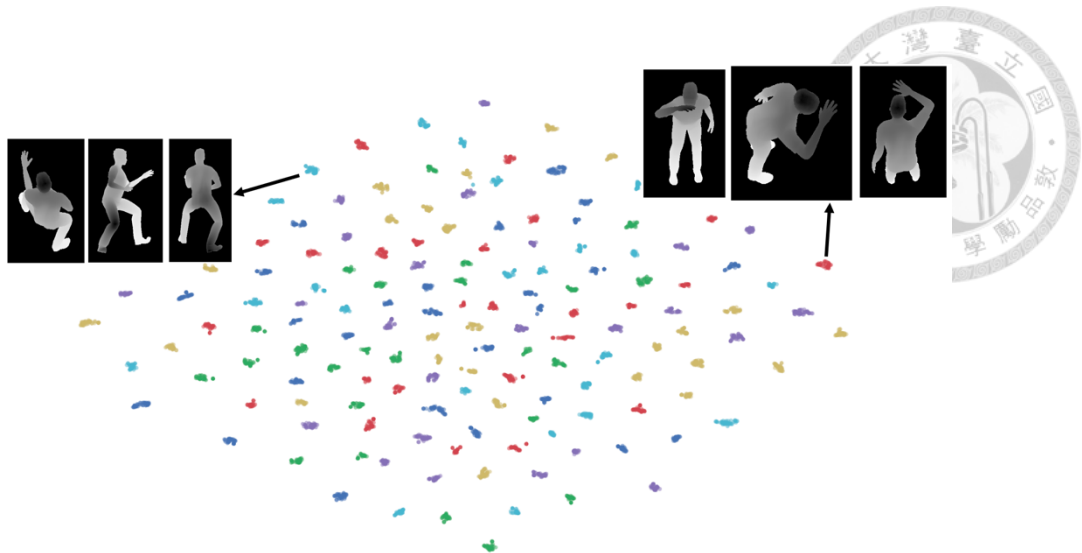


Figure 3-10 2D t-SNE visualization of View-Invariant Pose (VIP) features extracted from synthetic MVP dataset. Each cluster represents the same pose from various views. Due to the limit of palettes, the same color might appear multiple times.

To qualitatively visualize the view-invariant property, we project the learned high-dimension VIP features f_x^{VIP} extracted from synthetic MVP dataset to a 2D space by t-SNE [46]. As we can see in Figure 3-10, each cluster represents VIP features of the same pose from different views. That is to say, the VIP features look quite similar across different viewpoints in the high-dimension space, which demonstrates that the encoder E do extract the pose information excluding viewpoints as we expect.



(a) Real depth images from UWA 3D multi-view activity II dataset [27]. (b) Synthetic depth images from synthetic MVP dataset.

Figure 3-11 The visual difference between real and synthetic depth images. Note that real images contain blurred contours caused by noise.

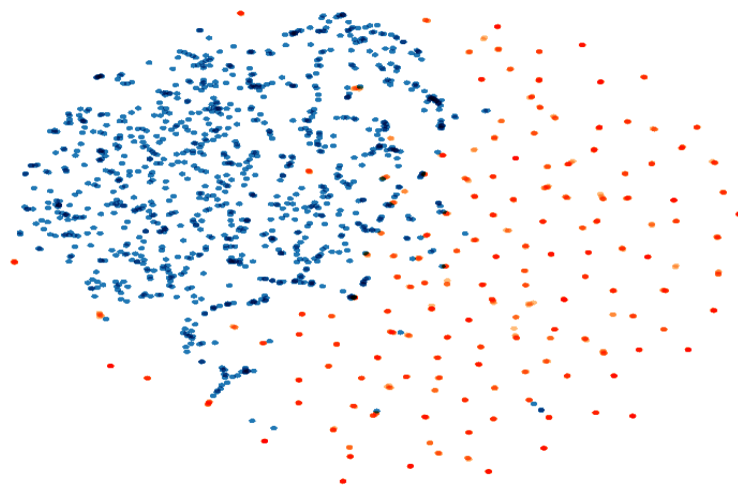


Figure 3-12 2D t-SNE visualization of the domain shift between View-Invariant Pose (VIP) features extracted from synthetic and real images. Orange points represent VIP features from synthetic data while blue points denote the VIP features from real data.

3.2.2 Adversarial Domain Adaptation

After we obtained the VIP feature encoder E learned from the synthetic MVP dataset, here comes a concern about whether we directly adopt it on the real data. In other words, do VIP features of a pose from synthetic MVP dataset and those of a similar pose from real dataset appear similar? Ideally, training and testing data should live in the same domain, such that the model learned from training data can be applied to testing data with no degradations. However, there is a visual gap between the synthetic and real depth images (see Figure 3-11) as well as their corresponding VIP features (see Figure 3-12),

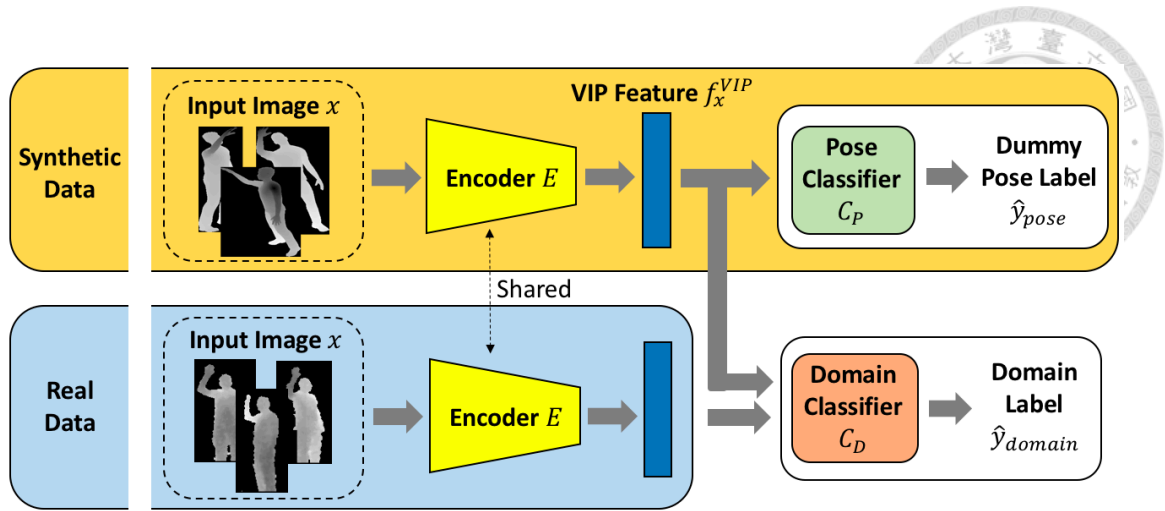


Figure 3-13 Unsupervised training architecture for learning View-Invariant Pose (VIP) feature with domain adaptation.

causing the phenomenon called “domain shift”.

It has been shown that a trained CNN model can be adapted to a new domain through fine-tuning. However, in our case, there is no such a large multi-view real dataset with pose labels, it is not suitable to directly fine-tune the encoder E . As a result, we utilize domain adaptation technique where source (or training) data and target (or testing) data come from similar but different distributions. Inspired by [35], we add a domain classifier C_D along with the existing encoder E and pose classifier C_P , combining domain adaptation and VIP feature learning into a unified training process (see Figure 3-13). The encoded VIP feature f_x^{VIP} is sent to the pose classifier C_P for dummy pose prediction, as well as to domain classifier C_D for domain prediction.

We consider a VIP feature f_x^{VIP} to be domain-invariant if a strong domain classifier C_D cannot distinguish from two domains. In our case, we refer the source domain to synthetic data and target domain to real data. The adversarial training technique [22] is utilized to jointly align the distributions of VIP features across two domains and transfer the view-invariant property to target domain.

Now the inputs to our training architecture are not only synthetic depth images $x \in$

syn with dummy pose labels y_{pose} but also real depth images $x \in real$ without pose labels. It is an unsupervised domain adaptation with the absence of target pose labels. We design two loss functions L_{domain} and L_{conf} for adversarial domain adaptation as follows:

$$L_{domain} = - \sum_{x \in real} \log(d_x) - \sum_{x \in syn} \log(1 - d_x) \quad (3.4)$$

$$L_{conf} = - \sum_x [0.5 \log(d_x) + 0.5 \log(1 - d_x)] \quad (3.5)$$

$$d_x = C_D(f_x^{VIP}) \quad (3.6)$$

where $d_x \in \mathbb{R}$ denotes the probability of image x coming from real domain, L_{domain} is the domain loss, and L_{conf} is the confusion loss. Note that during domain adaptation, all training data are unpaired, which means that there are no one-to-one corresponding images across domains.

Our proposed framework has two major differences from [35]. First, instead of using a gradient reverse layer that encourages a common domain by reversing the gradients, we confuse the domain classifier C_D by enforcing it to output a uniform distribution over domain labels with Eqn. (3.5) for all synthetic and real images. This advantage is also reported in [36]. Second, an adversarial scheme is used to update the networks. The process of learning VIP feature with domain adaptation is summarized in Algorithm 1. We train the networks in three stages. Such a training scheme is observed more stable and less sensitive to the optimization parameters [47]. The architecture design of the domain classifier and training details are described in Section 4.2.2 and Section 4.2.3.

Algorithm 1: Learning VIP feature with domain adaptation

Data : Synthetic depth images $x \in syn$ and corresponding dummy pose labels y_{pose} ;
Real depth images $x \in real$ without labels

Result: VIP feature encoder E for real domain

```
1 for Itrs. of VIP feature with domain adaptation do
2   for Itrs. of VIP feature do
3     | Update encoder  $E$  and pose classifier  $C_P$  to minimize Eqn. (3.1) on  $x \in syn$  and  $y_{pose}$ 
4   end
5   for Itrs. of domain classifier do
6     | Update domain classifier  $C_D$  to minimize Eqn. (3.4) on  $x \in syn$  and  $x \in real$ 
7   end
8   for Itrs. of encoder do
9     | Update encoder  $E$  to minimize Eqn. (3.5) on  $x \in syn$  and  $x \in real$ 
10  end
11 end
```

Figure 3-14 shows the qualitative process of domain adaptation observed from TensorBoard. Each figure illustrates the projected VIP features by t-SNE [46]. Note that the two domains gradually align to each other. Specifically, the domain shift is minimized during the learning process.

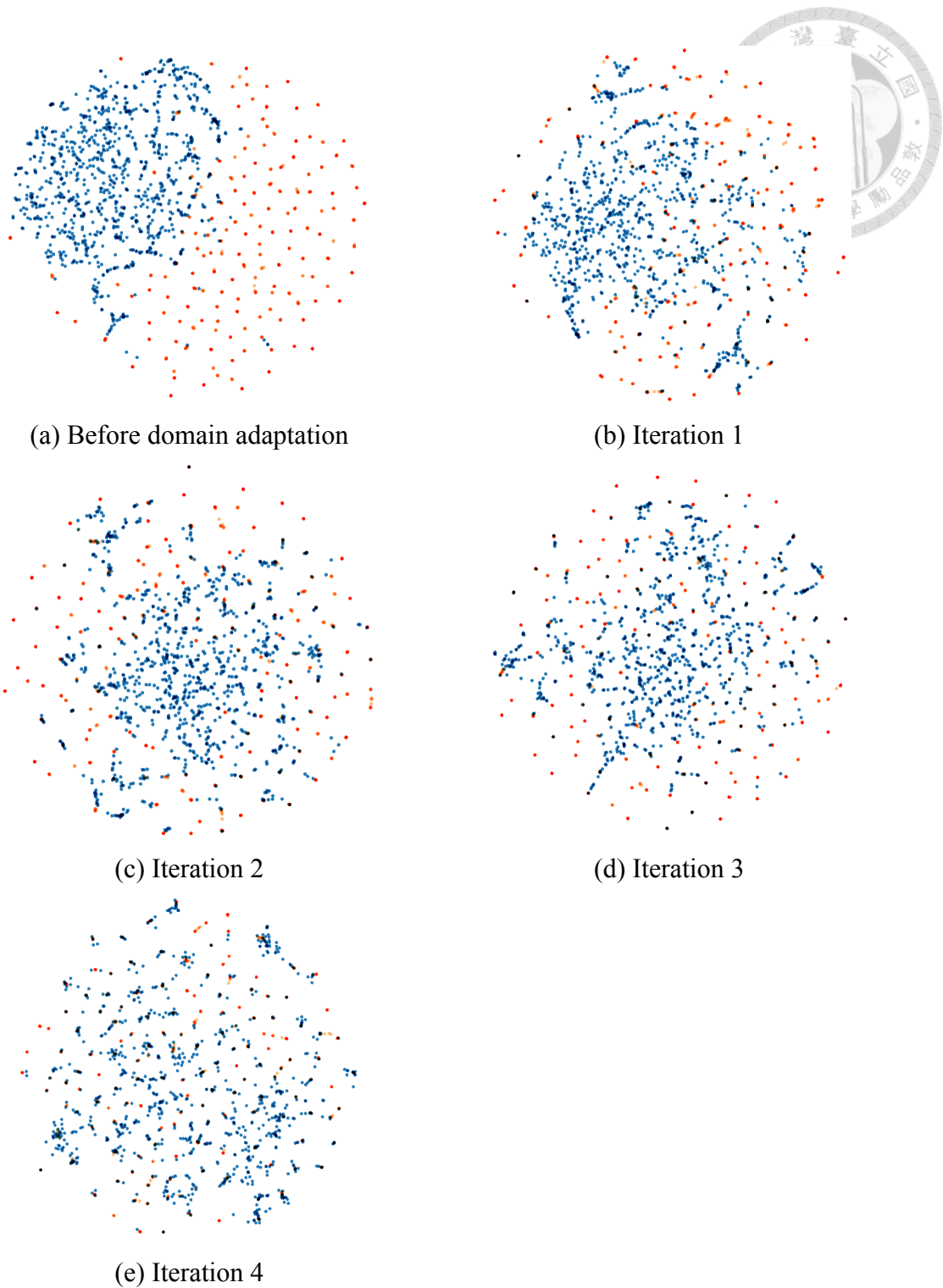


Figure 3-14 The process of learning View-Invariant Pose (VIP) feature with domain adaptation by Algorithm 1. Orange points represent VIP features from synthetic data while blue points denote the VIP features from real data. Note that the domain shift is minimized during the learning process.

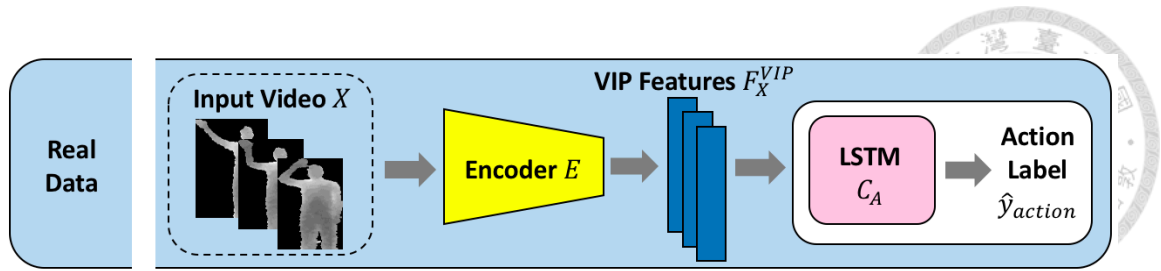


Figure 3-15 Modeling temporal information in an action sequence using LSTM.

3.3 Model Temporal Information

So far we have learned a VIP feature encoder E which maps human pose observed from any arbitrary view into a view-invariant space. However, the encoded VIP feature f_x^{VIP} only contains static and spatial information, the temporal information has to be further combined to describe actions. In this section, we introduce how to model the temporal progress in actions using LSTM [24].

With the temporal variations in each action, Recurrent Neural Networks (RNNs) are appealing in that they can directly map varying-length inputs. Although RNNs have demonstrated superior capabilities of modeling complex temporal dynamics in an input sequence, traditional RNNs are limited to learning long-term temporal dependencies because of the vanishing and exploding gradient problems. LSTM overcomes these challenges by introducing the gating mechanism and demonstrates its power on a large variety of problems.

Figure 3-15 illustrates an overview of the proposed pipeline for recognizing actions in videos captured from different viewpoints. Given an input video of length T , $X = \{x_t | t = 1, 2, \dots, T\}$, we consider an action as a sequence of poses and use the trained encoder E to extract the VIP feature $f_{x_t}^{VIP}$ from each input frame. Thus, we transform the input video X into a sequence of VIP features, $F_X^{VIP} = \{f_{x_t}^{VIP} | t = 1, 2, \dots, T\}$. The LSTM module acts as an action classifier C_A running through the sequence of the

extracted VIP features to model the temporal dependency.

The LSTM module learns the long-term dynamic which is discriminative to the action recognition task. The input to the LSTM module is a sequence of VIP features $F_X^{VIP} = \{f_{x_t}^{VIP} | t = 1, 2, \dots, T\}$ with $f_{x_t}^{VIP} = E(x_t)$ and the output is a probability distribution $a_X \in \mathbb{R}^C$ over all action classes. The objective of the LSTM module is to minimize the action loss L_{action} as the following:

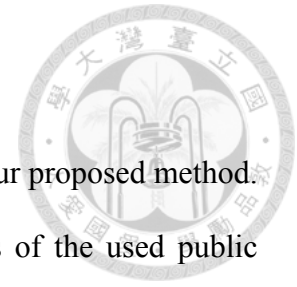
$$L_{action} = - \sum_X \sum_{c=1}^C \mathbb{I}[y_{action} = c] \log(a_{X,c}) \quad (3.7)$$

$$a_X = C_A(F_X^{VIP}) \quad (3.8)$$

where $a_{X,c} \in \mathbb{R}$ is the c -th element of a_X , which indicates the probability of video X belonging to c -th action class, C is the number of action types, and y_{action} denotes the ground truth action label. The action labels from training data are now required to train the LSTM module, which makes the training process supervised.

As the LSTM module is learned based on view-invariant features, we can generalize to different views with the same LSTM module even if the training data come from specific views. The architecture design of the LSTM module and training details are presented in Section 4.2.2 and Section 4.2.3.

Chapter 4 Experiments



In this chapter, we introduce the experiments which validates our proposed method. Before the experimental demonstrations, we give brief summaries of the used public action datasets. Then, the implementation details are presented. To quantify the view-invariant property of the proposed VIP feature, cross-view pose classification is also conducted on the synthetic MVP dataset.

As for cross-view action recognition, we compare our method with various state-of-the-arts on two public datasets, including NTU RGB+D action recognition dataset [15] and UWA 3D multi-view activity II dataset [27]. In order to show the effectiveness of the proposed VIP feature and domain adaptation, we also compare it with some alternative features. The quantitative experimental results confirm the advantages of our proposed method.

4.1 Action Datasets

To verify the performance of the proposed method, it is evaluated on two public benchmark datasets which are NTU RGB+D action recognition dataset [15] and UWA 3D multi-view activity II dataset [27]. Both of them are challenging datasets which have cross-view setting where the training data and testing data come from different viewpoints, thus they are suitable for validating cross-view action recognition methods.

4.1.1 NTU RGB+D Action Recognition Dataset

This dataset [15] is currently the largest publicly available dataset for 3D human action recognition, which contains more than 56,000 videos and 4 million frames. It is collected by Microsoft Kinect v2 and labeled with 60 action classes including daily, health-related, and interactive actions: *drink water, eat meal/snack, brushing teeth, brushing hair, drop, pickup, throw, sitting down, standing up, clapping, reading, writing,*

Table 4-1 The camera settings of NTU RGB+D action recognition dataset [15].

Setup No.	Height (m)	Distance (m)	Setup No.	Height (m)	Distance (m)
1	1.7	3.5	10	1.8	3.0
2	1.7	2.5	11	1.9	3.0
3	1.4	2.5	12	2.0	3.0
4	1.2	3.0	13	2.1	3.0
5	1.2	3.0	14	2.2	3.0
6	0.8	3.5	15	2.3	3.5
7	0.5	4.5	16	2.7	3.5
8	1.4	3.5	17	2.5	3.0
9	0.8	2.0			

tear up paper, wear jacket, take off jacket, wear a shoe, take off a shoe, wear on glasses, take off glasses, put on a hat/cap, take off a hat/cap, cheer up, hand waving, kicking something, put something inside pocket / take out something from pocket, hopping, jump up, make a phone call/answer phone, playing with phone/tablet, typing on a keyboard, pointing to something with finger, taking a selfie, check time, rub two hands together, nod head/bow, shake head, wipe face, salute, put the palms together, cross hands in front, sneeze/cough, staggering, falling, touch head (headache), touch chest (stomachache), touch back (backache), touch neck (neckache), nausea or vomiting condition, use a fan (with hand or paper)/feeling warm, punching/slapping other person, kicking other person, pushing other person, pat on back of other person, point finger at the other person, hugging other person, giving something to other person, touch other person's pocket, handshaking, walking towards each other, and walking apart from each other.

The actions are performed by 40 subjects with different scales. Three cameras are used to capture action videos simultaneously from three horizontal angles: -45° , 0° , and 45° . Every subject performs each action twice while facing the left and right camera respectively.

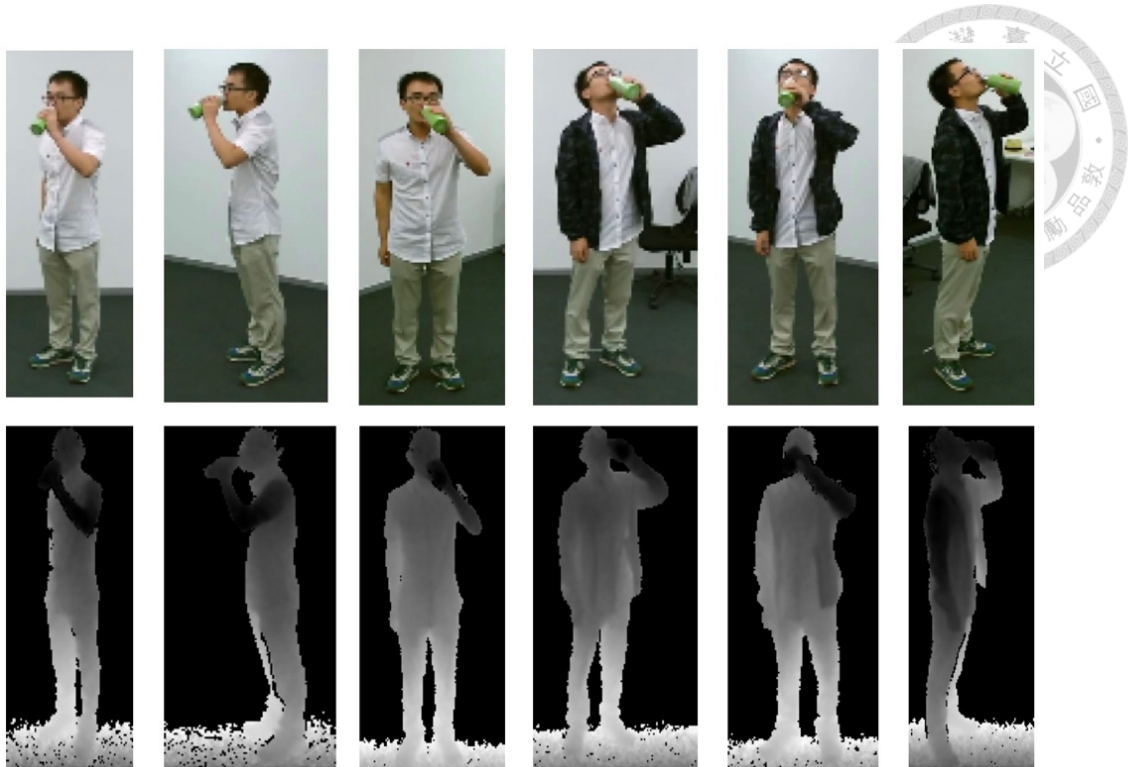


Figure 4-1 Multi-view RGB and depth images of *drinking water* from NTU RGB+D action recognition dataset [15].

Moreover, 17 camera settings listed in Table 4-1 are adopted to get more viewpoint variations. Combined with different human orientations, the dataset is collected from 80 distinct viewpoints. Therefore, it is suitable for validating cross-view action recognition methods. In addition to depth videos, RGB videos, infrared videos, and 3D coordinates of 25 body joints are also provided in this dataset. Figure 4-1 shows some sample frames of *drinking water* from different views. The viewpoint and large intra-class variations make this dataset very challenging. This dataset is highly suitable for data-hungry algorithm for the task of depth-based or skeleton-based human activity analysis.

4.1.2 UWA 3D Multi-View Activity II Dataset

This dataset [27] is comprised of 30 daily-life human actions performed by 10 subjects with different scales: *one-hand waving*, *one-hand punching*, *two-hand waving*, *two-hand punching*, *sitting down on floor*, *standing up*, *vibrating*, *falling down*, *holding chest*, *holding head*, *holding back*, *walking*, *irregular walking*, *lying down*, *turning*

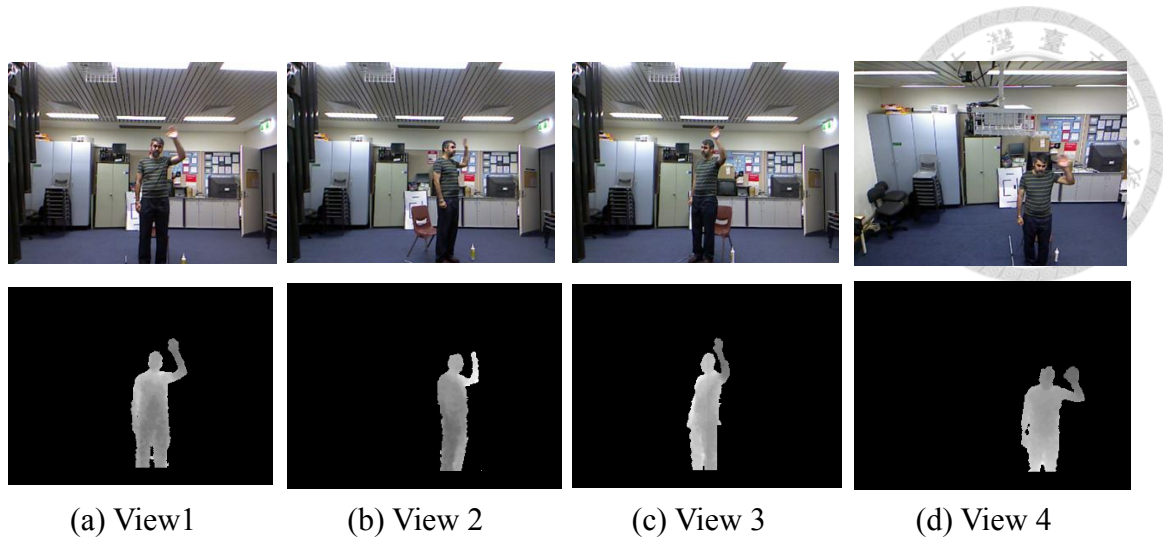


Figure 4-2 Multi-view RGB and depth images of *one-hand waving* from UWA 3D multi-view activity II dataset [27]. Note that some body parts might not be fully captured by the camera.

around, drinking, phone answering, bending, jumping jack, running, picking up, putting down, kicking, jumping, dancing, moping floor, sneezing, sitting down on chair, squatting, and coughing.

Each subject performs all 30 actions in a continuous manner for 4 times. Each time it is captured from a specific viewpoint (front, left, right, and top) using the Microsoft Kinect v1, thus the video acquisition from 4 views is nonsynchronous. Besides, the videos are preprocessed by cropping the continuous sequences of each action.

This dataset is challenging since the videos are recorded at different times from varying viewpoints and the data contains high similarity across action classes. For instance, *phone answering* and *drinking* have very similar arm movements. Moreover, in the top-view setting, the lower body parts are not properly captured due to self-occlusion. Figure 4-2 shows some sample frames of *one-hand waving* from different views.

4.2 Implementation Details

In this section, we describe the implementation details in the experiments including synthesizing a Multi-View Pose (MVP) dataset, architecture designs, training details, and

the pipeline of action recognition.

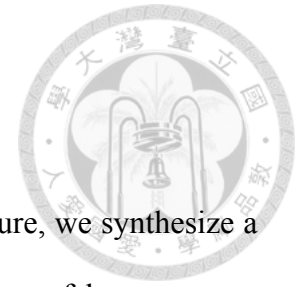
4.2.1 Synthesize a Multi-View Pose Dataset

In order to efficiently learn the View-Invariant Pose (VIP) feature, we synthesize a Multi-View Pose (MVP) dataset. It contains synthetic depth images of human poses captured from multiple viewpoints.

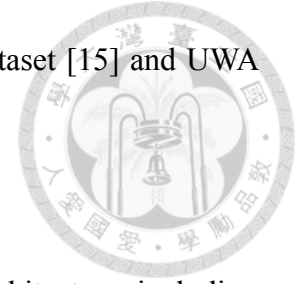
We sample 100,000 poses out of 4 million poses from CMU motion capture database [20] so as to find representative poses in the pose space. Using HDBSCAN clustering algorithm [38], we get $K = 195$ clusters from by setting the minimum cluster size to 20 and minimum samples to 25. Thus, we build a dictionary consisting of 195 poses with each treated as a different pose and given a unique dummy pose label.

Considering the variation of subjects in action datasets, we utilize an open source package, MakeHuman simulator [44], to create 12 human models with different combinations of gender, body shape, hair style, and clothes as shown in Figure 3-5. Moreover, we utilize another open source package, Blender [45], to simulate a virtual environment and re-target each human model with all different poses in the pose dictionary as shown in Figure 3-6 (b).

To render depth images from multiple viewpoints, 180 virtual cameras with distinct polar angles $\phi_{\text{polar}} \in \{0^\circ, 10^\circ, 20^\circ, 30^\circ, 40^\circ, 50^\circ, 60^\circ, 70^\circ, 80^\circ, 90^\circ\}$ and azimuthal angles $\phi_{\text{azim}} \in \{0^\circ, 20^\circ, 40^\circ, 60^\circ, 80^\circ, 100^\circ, 120^\circ, 140^\circ, 160^\circ, 180^\circ, 200^\circ, 220^\circ, 240^\circ, 260^\circ, 280^\circ, 300^\circ, 320^\circ, 340^\circ, 360^\circ\}$ are uniformly placed on a hemisphere around the subject as depicted in Figure 3-7. To sum up, we synthesize a MVP dataset containing depth images of 195 poses with 12 human models captured from 180 viewpoints. Therefore, the dataset includes 421200 ($195 \times 12 \times 180$) depth images. Figure 3-8 illustrates some samples. We utilize the same synthetic MVP dataset to generalize the VIP



feature representation for both NTU RGB+D action recognition dataset [15] and UWA 3D multi-view activity II dataset [27].



4.2.2 Architecture Design

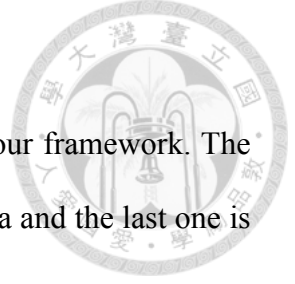
In this part, we describe the detailed design of our proposed architecture, including the encoder, pose classifier, and domain classifier as shown in Figure 3-13 and the LSTM module as depicted in Figure 3-15.

For the encoder E , we use the “Xception network” [41] which consists of depth-wise separable convolutions and the last fully-connected layer is replaced with a 256-neuron layer followed by a \tanh activation. Thus, the extracted VIP feature $f_x^{VIP} \in \mathbb{R}^{256}$ is a 256-dimension vector and each neuron in f_x^{VIP} is activated in $(-1, 1)$.

As for the pose classifier C_p , since we build a pose dictionary containing 195 poses in the synthetic MVP dataset, we design a simple one-layer fully-connected neural network with 195 neurons followed by a softmax activation, corresponding to each pose in the dictionary.

On the other hand, for the domain classifier C_D , a 3-layer neural network is designed. The first two layers are both comprised of 64 neurons followed by a leaky ReLU [48] with slope equal to 0.3 and the last layer aims at binary classification using one neuron with sigmoid activation.

For the LSTM module, we design a 2-layer stacked LSTM with 256 hidden units and aggregate the outputs from all layers with residual connection [3]. Namely, we sum the outputs of all LSTM layers instead of using only the last one. A fully-connected layer with 128 neurons followed by a ReLU activation [49] and a dropout layer [50] are further added on the top of the stacked LSTM. In the last, we add a fully-connected layer with C neurons followed by a softmax activation where C denotes the number of action types.



4.2.3 Training Details

As shown in Figure 4-3, it is a three-stage training process in our framework. The first two stages are used for learning VIP feature encoder for real data and the last one is for learning action information.

In order to have a good initialization for the VIP feature encoder E while learning with domain adaptation as described in Algorithm 1, the encoder is firstly pre-trained without domain adaptation as the framework shown in Figure 3-9. The 180 virtual viewpoints in the synthetic MVP dataset are randomly split into training and testing set which cover 162 and 18 views respectively. The encoder is initialized as the ImageNet pre-trained Xception model [41] and fine-tuned on the synthetic MVP dataset. Given $x \in syn$ and y_{pose} , we update the encoder and pose classifier by using stochastic gradient descent with mini-batch of 32 samples to minimize VIP loss L_{VIP} , Eqn. (3.1), through backpropagation. The network is updated by Adam optimizer [51] with initial learning rate equal to 1×10^{-3} and the training process is monitored with early stopping.

The original input to Xception network is a three-channel RGB image. In our case,

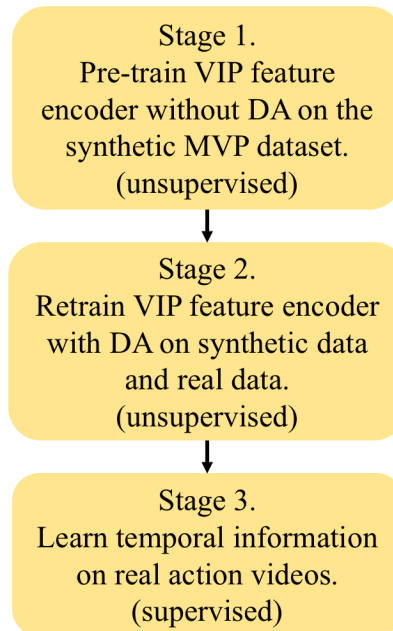


Figure 4-3 The training stages in our proposed method. DA is an abbreviation for domain adaptation.

we duplicate the depth image to fit three channels, making the input dimension consistent. Besides, the input images are augmented by random flipping horizontally and random zooming during training.

Then the VIP knowledge is further adapted to NTU RGB+D action recognition dataset [15] and to UWA 3D multi-view activity II dataset [27] respectively. Note that all training data for learning VIP feature with domain adaptation are unpaired, which means that there are no one-to-one corresponding images across domains. As summarized in Algorithm 1, we set *Iters. of VIP feature with domain adaptation* to 4, *Iters. of VIP feature* to 1, *Iters. of domain classifier* to 2, and *Iters. of encoder* to 1. Stochastic gradient descent with mini-batch of 32 samples is utilized and the learning process is optimized by Adam optimizer with initial learning rate equal to 5×10^{-6} . Each batch contains half of real data and half of synthetic data.

The last step is to learn the temporal information. The input sequence is down-sampled with ratio equal to 5 and each sampled point is randomly shifted along time domain for data augmentation. We use stochastic gradient descent with mini-batch of 8 samples and update the LSTM by backpropagation through time using Adam optimizer with initial learning rate equal to 1×10^{-3} . The encoder E is fixed while training on UWA 3D multi-view activity II dataset. However, since the NTU RGB+D action recognition dataset contains several actions involving object interactions like *drink water*, *eat meal/snack*, *brushing teeth*, *brushing hair*, *drop*, *pickup*, *reading*, *writing*, *tear up paper*, *wear jacket*, *take off jacket*, *wear a shoe*, *take off a shoe*, *wear on glasses*, *take off glasses*, *put on a hat/cap*, *take off a hat/cap*, *put something inside pocket* /*take out something from pocket*, *make a phone call/answer phone*, *playing with phone/tablet*, *typing on a keyboard*, *taking a selfie*, and *use a fan*. We jointly fine-tune the encoder with learning rate equal to 5×10^{-6} while updating LSTM on NTU RGB+D dataset. The

Table 4-2 Pose classification accuracy on the synthetic MVP dataset.

Validation data	Testing data
97.4 %	95.5 %

encoder is expected to further extract object interaction while maintaining view-invariant property. The learning process is also monitored with early stopping.

In an addition, NTU RGB+D action recognition dataset has both one-person actions and two-person interactions. For those samples with two peoples, we predict the action type for each person and average the result.

As for the platform, we utilize Keras, which is a high-level API running on top of TensorFlow, CNTK, and Theano, as the deep learning platform. The whole algorithm is implemented with Python 3.6. For computation resource, we use a personal computer with Intel 4-core i7-6700 3.4 GHz CPU and a single GPU of GTX 1080Ti for all the training processes.

4.3 Cross-View Pose Classification

To quantify the view-invariant property of the proposed VIP feature, we design an experiment for cross-view pose classification. Due to the fact that we only have multi-view data with pose labels on the synthetic MVP dataset. We conduct on synthetic MVP dataset.

All 180 viewpoints are split into 162 training views and 18 testing views. 20 percentage of the data in training views further serve as validation data. Thus, we have 303264 training samples, 75816 validation samples, and 42120 testing samples. Each data is associated with a dummy pose label. Note that training data and validation data share the same viewpoints while testing data have distinct views. We use the same trained VIP feature encoder and pose classifier and report the accuracy.

The result is listed in Table 4-2. Even if the pose is observed from novel viewpoints (testing data), we can still achieve high accuracy by 95.5 %, which demonstrates the robustness of VIP feature to viewpoint variations.

In addition, Figure 4-4 qualitatively visualize the view-invariant property by concatenating the extracted $f_x^{VIP} \in \mathbb{R}^{256}$ of the same pose observed from 180 views as an 180×256 image. Each element in f_x^{VIP} is normalized into $[0, 255]$. As expected, the VIP features appear very similarly across all 180 viewpoints for each pose.

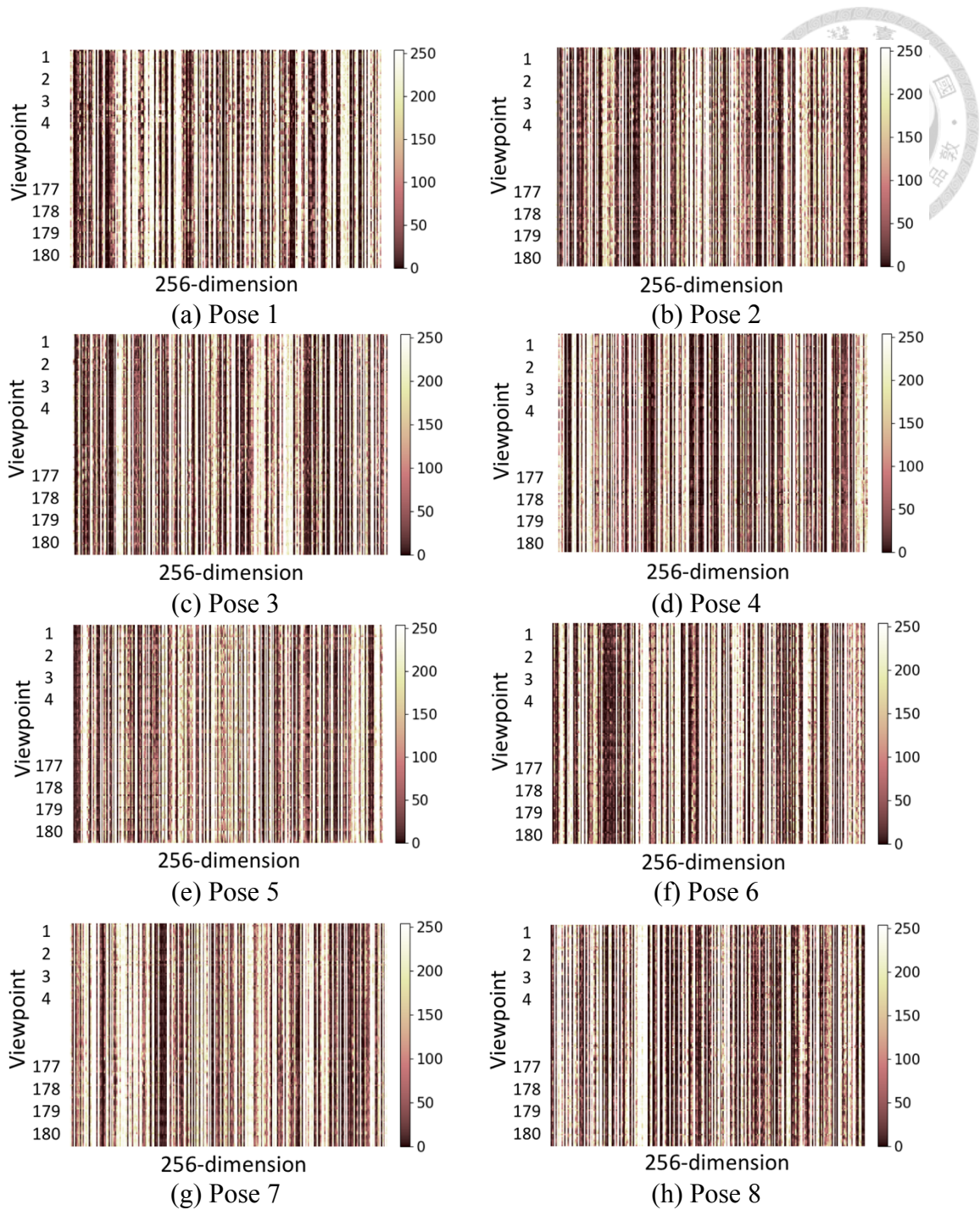


Figure 4-4 Qualitative visualization of view-invariant property. Each image represents a pose in the synthetic MVP dataset. Each row in an image denotes a VIP feature $f_x^{VIP} \in \mathbb{R}^{256}$ extracted from a specific view. The 180 rows correspond to 180 viewpoints of the same pose.

4.4 Action Recognition Results

To validate the effectiveness of our proposed method for action recognition, we compare the performance with several state-of-the-arts. According to the input modalities,

they can be categorized into the following types:

- I. RGB-based methods: Action Tube [52], AND-OR Graph (AOG) [28], Dense Trajectories (DT) [53], Long-term Recurrent Convolutional Network (LRCN) [10], Hannelets [54], Two-stream CNN (Two-Stream) [9], 3D convolution (3D Conv) [7], non-linear Circulant Temporal Encoding (nCTE) [29], Non-linear Knowledge Transfer Model (NKTM) [26], and Robust Non-linear Knowledge Transfer Model (R-NKTM) [25]
- II. Depth-based methods: Histogram of Oriented 4D Normals (HON4D) [12], Super Normal Vector (SNV) [13], 2D Histogram of Oriented Gradients (HOG²) [14], Long-Term Motion Dynamics (LTMD) [55], Comparative Coding Descriptor (CCD) [56], Discriminative Virtual Views (DVV) [32], Continuous Virtual Path (CVP) [30], Histogram of Oriented 3D Point Cloud (HOPC) [27], and Human Pose Model with Temporal Modeling (HPM+TM) [16]
- III. Skeleton-based methods: Skeletal Quads [57], Lie Group [58], Deep RNN, Deep LSTM, Part-aware LSTM [15], Hierarchically Bidirectional RNN (HBRNN) [59], Deep Learning on Lie Group (LieNet) [60], Spatio-Temporal LSTM with Trust Gates (ST-LSTM+TG) [61], Spatio-Temporal Attention LSTM (STA-LSTM) [62], Histograms of 3D joint locations (HOJ3D) [63], and Actionlet Ensemble (AE) [64]
- IV. Multi-modality methods: Deep Shared-Specific Component Analysis and Structure Sparsity Learning Machine (DSSCA-SSLM) [65] and Depth+ Skeleton with Rank Pool (D+S-RP) [66]

The quantitative results of these state-of-the-arts are reported from their original papers and [15, 25, 27, 61]. Our proposed method is denoted as VIP w/ DA (View-

Invariant Pose with Domain Adaptation). In addition to other compared methods, we also report the performance of some baseline models defined by ourselves, including:

- I. Xception + LSTM: We use Xception network [41] as encoder to extract spatial feature and use LSTM module to capture temporal features in actions. We initialize the Xception from the ImageNet pre-trained model. The Xception and LSTM module are jointly trained end-to-end only under the supervision of action labels provided by action datasets. Note that the encoder does not learn the VIP knowledge in advance.
- II. VIP w/o DA: We firstly train the encoder to learn VIP knowledge from the synthetic MVP dataset as the framework depicted in Figure 3-9. No more domain adaptation is conducted. Then we model the temporal information under the supervision of action labels provided by action datasets.

4.4.1 Action Recognition Pipeline

The proposed system pipeline for action recognition is summarized in Figure 4-5. Depth sensors such as Kinect and Xtion PRO are able to segment the human bodies from the input depth video in real-time.

We firstly crop the boundary of human segmentation and resize to a 128×128

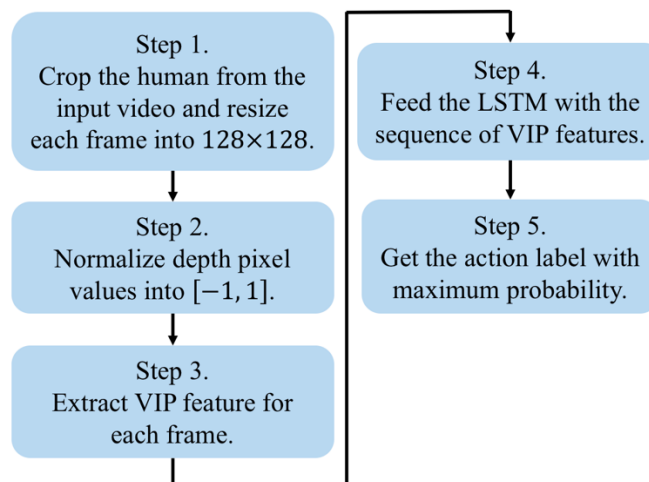


Figure 4-5 The pipeline of action recognition from input video.

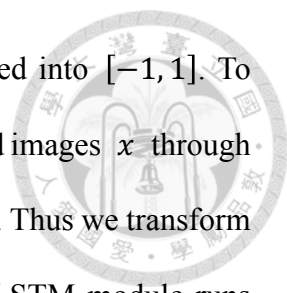


image for each input frame. The depth values are further normalized into $[-1, 1]$. To represent an action as a sequence of poses, we feed forward the resized images x through the encoder E and temporally align the extracted VIP features f_x^{VIP} . Thus we transform the input video X to a sequence of VIP features F_X^{VIP} . Then the LSTM module runs through the VIP features and we take the action label with maximum probability as the predicted output.

4.4.2 The Result of NTU RGB+D Action Recognition Dataset

We follow the cross-subject and cross-view evaluations suggested by [15] and report the classification accuracy in percentage. For the cross-subject protocol, all 40 subjects are split into training and testing sets with each containing 20 subjects. On the other hand, for the cross-view protocol, all the samples recorded by camera 1 and camera 2 are used for training while the samples captured by camera 3 are for testing.

Table 4-3 shows the performance of several methods. Since there are few RGB-based methods implemented on this dataset, we choose skeleton-based methods as alternatives to compare with. Our method (VIP w/ DA) outperforms all existing methods on both cross-subject and cross-view evaluations. Although skeleton data are somewhat immune to viewpoint variations, they cannot include interactions with environmental objects. Our method achieves 86.5 % in cross-view evaluation, which is about 3.4% better than the nearest competitor, D+S-RP [66]. It is worthy to note that our method only depends on depth information that is less privacy sensitive. Besides, while the training and testing data share the viewpoints (cross-subject setting), our proposed method reaches a superior performance by 86.1 %.

Table 4-3 Comparison of action recognition accuracy (%) on the NTU RGB+D Action Recognition Dataset [15]. Our proposed method is denoted as VIP w/ DA. Our defined baseline models are Xception + LSTM and VIP w/o DA.

Methods	Data type	Cross-subject	Cross-view
HON4D [12]	Depth	30.6	7.3
SNV [13]	Depth	31.8	13.6
HOG ² [14]	Depth	32.2	22.3
Skeletal Quads [57]	Skeleton	38.6	41.4
Lie Group [58]	Skeleton	50.1	52.8
Deep RNN [15]	Skeleton	56.3	64.1
HBRNN [59]	Skeleton	59.1	64.0
Deep LSTM [15]	Skeleton	60.7	67.3
LieNet [60]	Skeleton	61.4	67.0
Part-aware LSTM [15]	Skeleton	62.9	70.3
LTMD [55]	Depth	66.2	-
ST-LSTM+TG [61]	Skeleton	69.2	77.7
STA-LSTM [62]	Skeleton	73.4	81.2
DSSCA-SSLM [65]	RGB + Depth	74.9	-
D+S-RP [66]	Depth+ Skeleton	75.2	83.1
Xception + LSTM	Depth	69.8	70.0
VIP w/o DA	Depth	84.3	82.7
VIP w/ DA	Depth	86.1	86.5

From the results of baseline models, pre-learning VIP information (VIP w/o DA) can significantly benefit the results compared with Xception + LSTM by 14.5% higher in cross-subject setting and 12.7% higher in cross-view setting. It also leads to a faster convergence. In addition, domain adaptation can further boost the performance by 1.8% higher in cross-subject setting and 3.8% higher in cross-view setting. Hence, the result shows the effectiveness and applicability of our proposed method.

4.4.3 The Result of UWA 3D Multi-View Activity II Dataset

We follow the cross-view evaluation provided in [27] where videos from two views are used for training and videos from the remaining views are used for testing respectively. It leads to 12 different cross-view combinations in this evaluation protocol.

Table 4-4 Comparison of action recognition accuracy (%) on the UWA 3D multi-view activity II dataset [27]. Each column represents a different cross-view setting. For example, V_{12}^3 means the model is trained on view 1 and view 2 while tested on view 3. Our proposed method is denoted as VIP w/ DA. Our defined baseline models are Xception + LSTM and VIP w/o DA.

Methods	Data type	V_{12}^3	V_{12}^4	V_{13}^2	V_{13}^4	V_{14}^2	V_{14}^3	V_{23}^1	V_{23}^4	V_{24}^1	V_{24}^3	V_{34}^1	V_{34}^2	Avg
CCD [56]	Depth	10.5	13.6	10.3	12.8	11.1	8.3	10.0	7.7	13.1	13.0	12.9	10.8	11.2
HOJ3D [63]	Skeleton	15.3	28.2	17.3	27.0	14.6	13.4	15.0	12.9	22.1	13.5	20.3	12.7	17.7
DVV [32]	Depth	23.5	25.9	23.6	26.9	22.3	20.2	22.1	24.5	24.9	23.1	28.3	23.8	24.1
AOG [28]	Depth	29.3	31.1	25.3	29.9	22.7	21.9	25.0	20.2	30.5	27.9	30.0	26.8	26.7
HON4D [12]	Depth	31.1	23.0	21.9	10.0	36.6	32.6	47.0	22.7	36.6	16.5	41.4	26.8	28.9
SNV [13]	Depth	31.9	25.7	23.0	13.1	38.4	34.0	43.3	24.2	36.9	20.3	38.6	29.0	29.9
Action Tube [52]	RGB	49.1	18.2	39.6	17.8	35.1	39.0	52.0	15.2	47.2	44.6	49.1	36.9	37.0
CVP [30]	Depth	36.0	34.7	35.0	43.5	33.9	35.2	40.4	36.3	36.3	38.0	40.6	37.7	37.3
AE [64]	Skeleton	45.0	40.4	35.1	36.9	34.7	36.0	49.5	29.3	57.1	35.4	49.0	29.3	39.8
AOG [28]	RGB	47.3	39.7	43.0	30.5	35.0	42.2	50.7	28.6	51.0	43.2	51.6	44.2	42.3
LRCN [10]	RGB	53.9	20.6	43.6	18.6	37.2	43.6	56.0	20.0	50.5	44.8	53.3	41.6	40.3
Lie Group [58]	Skeleton	49.4	42.8	34.6	39.7	38.1	44.8	53.3	33.5	53.6	41.2	56.7	32.6	43.4
Hankelets [54]	RGB	46.0	51.5	50.2	59.8	41.9	48.1	66.6	51.3	61.3	38.4	57.8	48.9	51.8
HOPC [27]	Depth	52.7	51.8	59.0	57.5	42.8	44.2	58.1	38.4	63.2	43.8	66.3	48.0	52.2
Two-Stream [9]	RGB	63.0	47.1	55.8	60.6	53.4	54.2	66.0	50.9	65.3	55.5	68.0	51.9	57.6
DT [53]	RGB	57.1	59.9	54.1	60.6	61.2	60.8	71.0	59.5	68.4	51.1	69.5	51.5	60.4
3D Conv [7]	RGB	59.5	59.6	56.6	64.0	59.5	60.8	71.7	60.0	69.5	53.5	67.1	50.4	61.0
nCTE [29]	RGB	55.6	60.6	56.7	62.5	61.9	60.4	69.9	56.1	70.3	54.9	71.7	54.1	61.2
NKTM [26]	RGB	60.1	61.3	57.1	65.1	61.6	66.8	70.6	59.5	73.2	59.3	72.5	54.5	63.5
R-NKTM [25]	RGB	64.9	67.7	61.2	68.4	64.9	70.1	73.6	66.5	73.6	60.8	75.5	61.2	67.4
HPM+TM [16]	Depth	80.6	80.5	75.2	82.0	65.4	72.0	77.3	67.0	83.6	81.0	83.6	74.1	76.9
D+S-RP [66]	Depth + Skeleton	86.8	87.0	80.7	89.1	78.1	80.9	86.5	79.3	85.1	86.9	89.4	80.0	84.2
Xception + LSTM	Depth	42.9	43.8	47.7	48.7	32.0	30.2	68.0	39.7	63.6	28.3	61.0	36.1	45.2
VIP w/o DA	Depth	76.1	79.8	73.7	80.5	71.4	72.8	74.7	73.0	74.7	75.0	77.3	74.4	75.3
VIP w/ DA	Depth	85.8	86.5	85.7	87.3	80.5	81.0	85.9	86.1	85.1	83.2	88.8	82.7	84.9

The quantitative results are summarized in Table 4-4. Our proposed method (VIP w/ DA) outperforms most state-of-the-arts and performs equally well when compared with D+S-RP [66] by achieving 84.9 % average accuracy. D+S-RP depends on not only depth data but also skeleton data which may not always be accurately acquired in real applications due to self-occlusion. Take a skeleton-based method such as Lie group [58] for example, Lie group performs badly on V_{13}^2 , V_{14}^2 , V_{13}^4 , and V_{34}^2 , which means the skeleton data are not reliable. However, VIP w/ DA outperforms D+S-RP on these

evaluation settings, which demonstrates our robustness. Besides, domain adaptation gives a significant improvement by 9.6 % in average compared with VIP w/o DA.

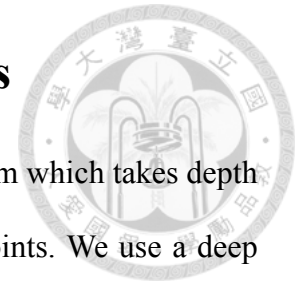
We can see that the deep learning-based methods such as Action Tubes [52] and LRCN [10] achieve low accuracies as they were proposed for recognizing actions from a specific view. DT [53] has a relatively high performance because the dense trajectories of videos obtained from side views are similar.

The overall performances of depth-based methods such as CCD [56], HON4D [12], SNV [13], and HOPC [27] are low because depth appearances of several actions look quite different across views. The performance of skeleton-based methods is worse on this dataset since the skeleton data is not well provided for some actions such as phone answering, drinking, sneezing or is not available for some actions like lying down and falling down.

The result demonstrates that the proposed VIP feature is capable of mapping view-dependent pose information into a view-invariant space. Besides, it also demonstrates that the learned VIP knowledge from synthetic data can be transferred to real data through adversarial domain adaptation.

It is interesting to note that for several actions in this dataset such as *holding back*, *holding head*, *coughing*, and *sneezing*, there are no similar actions included in the CMU motion capture database [20]. However, we can still achieve high performance for these activities, which demonstrates the generalization ability of our proposed VIP feature.

Chapter 5 Conclusion and Future Works



In this thesis, we propose a cross-view action recognition system which takes depth video as input to recognize human activities from different viewpoints. We use a deep Convolutional Neural Network (CNN) to extract View-Invariant Pose (VIP) feature by mapping human poses observed from different viewpoints into a shared view-invariant feature space.

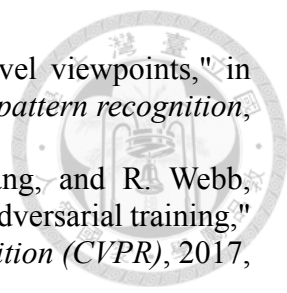
In order to train such a deep network, we design a simple but efficient pipeline to synthesize a Multi-View Pose (MVP) dataset which contains depth images of human poses captured from multiple views. The VIP knowledge is distilled from the synthetic MVP dataset in an unsupervised way and further transferred to real data through adversarial domain adaptation. As for recognizing actions from input video, we adopt Long Short-Term Memory (LSTM) to mine the temporal dependencies. Experimental results show that the proposed method achieves promising performance over state-of-the-arts on two benchmark multi-view 3D human action datasets.

Regarding future works, analyzing the relations between action labels and pose labels in real data will benefit domain adaptation, making it semi-supervised. In addition, we notice that action detection system is more practical in real applications where the input is an untrimmed video sequence. Different from action recognition system, not only the action categories but also the temporal locations are analyzed. We expect that the VIP feature will also be applicable for cross-view action detection.

REFERENCE



- [1] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *Image and vision computing*, vol. 60, pp. 4-21, 2017.
- [2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, 2015.
- [3] 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*, 2016, pp. 770-778.
- [4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.
- [5] 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*, 2015, pp. 3431-3440.
- [6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91-99.
- [7] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, pp. 221-231, 2013.
- [8] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4489-4497.
- [9] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568-576.
- [10] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625-2634.
- [11] T.-W. Hsu, Y.-H. Yang, T.-H. Yeh, A.-S. Liu, L.-C. Fu, and Y.-C. Zeng, "Privacy free indoor action detection system using top-view depth camera based on key-poses," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016, pp. 4058-4063.
- [12] O. Oreifej and Z. Liu, "Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 716-723.
- [13] X. Yang and Y. Tian, "Super normal vector for activity recognition using depth sequences," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 804-811.
- [14] E. Ohn-Bar and M. M. Trivedi, "Joint angles similarities and HOG2 for action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013, pp. 465-470.
- [15] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

- 
- [16] H. Rahmani and A. Mian, "3d action recognition from novel viewpoints," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1506-1515.
- [17] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 6.
- [18] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [19] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen, "Synthesizing training images for boosting human 3d pose estimation," in *International Conference on 3D Vision (3DV)*, 2016, pp. 479-488.
- [20] . *CMU graphics lab motion capture database*. Available: <http://mocap.cs.cmu.edu/>
- [21] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, pp. 151-175, 2010.
- [22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672-2680.
- [23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104-3112.
- [24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735-1780, 1997.
- [25] H. Rahmani, A. Mian, and M. Shah, "Learning a deep model for human action recognition from novel viewpoints," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, pp. 667-681, 2018.
- [26] H. Rahmani and A. Mian, "Learning a non-linear knowledge transfer model for cross-view action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2458-2466.
- [27] H. Rahmani, A. Mahmood, D. Huynh, and A. Mian, "Histogram of oriented principal components for cross-view action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, pp. 2430-2443, 2016.
- [28] J. Wang, X. Nie, Y. Xia, Y. Wu, and S.-C. Zhu, "Cross-view action modeling, learning and recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2649-2656.
- [29] A. Gupta, J. Martinez, J. J. Little, and R. J. Woodham, "3D pose from motion for cross-view action recognition via non-linear circulant temporal encoding," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2601-2608.
- [30] Z. Zhang, C. Wang, B. Xiao, W. Zhou, S. Liu, and C. Shi, "Cross-view action recognition via a continuous virtual path," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2690-2697.
- [31] J. Zheng and Z. Jiang, "Learning view-invariant sparse representations for cross-view action recognition," in *IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 3176-3183.
- [32] R. Li and T. Zickler, "Discriminative virtual views for cross-view action

- recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 2855-2862.
- [33] J. Liu, M. Shah, B. Kuipers, and S. Savarese, "Cross-view action recognition via view knowledge transfer," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3209-3216.
- [34] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 4.
- [35] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the International Conference on Machine Learning*, 2015.
- [36] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4068-4076.
- [37] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Kdd*, 1996, pp. 226-231.
- [38] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*, 2013, pp. 160-172.
- [39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper With Convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.
- [40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818-2826.
- [41] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.
- [42] C. Wang, Y. Wang, and A. L. Yuille, "An approach to pose-based action recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 915-922.
- [43] L. Seidenari, V. Varano, S. Berretti, A. Del Bimbo, and P. Pala, "Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013, pp. 479-485.
- [44] . *MakeHuman: open source tool for making 3D characters*. Available: <http://www.makehuman.org/>
- [45] . *Blender: a 3D modelling and rendering package*. Available: <http://www.blender.org/>
- [46] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, pp. 2579-2605, 2008.
- [47] Y.-C. Liu, W.-C. Chiu, S.-D. Wang, and Y.-C. F. Wang, "Domain-Adaptive generative adversarial networks for sketch-to-photo inversion," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017, pp. 1-6.
- [48] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on*

- Machine Learning*, 2013, p. 3.
- [49] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the International Conference on Machine Learning*, 2010, pp. 807-814.
- [50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.
- [51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2015.
- [52] G. Gkioxari and J. Malik, "Finding action tubes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 759-768.
- [53] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3169-3176.
- [54] B. Li, O. I. Camps, and M. Sznajder, "Cross-view activity recognition using hanklets," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 1362-1369.
- [55] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei, "Unsupervised learning of long-term motion dynamics for videos," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [56] Z. Cheng, L. Qin, Y. Ye, Q. Huang, and Q. Tian, "Human daily action analysis with multi-view and color-depth data," in *European Conference on Computer Vision (ECCV)*, 2012, pp. 52-61.
- [57] G. Evangelidis, G. Singh, and R. Horaud, "Skeletal quads: Human action recognition using joint quadruples," in *International Conference on Pattern Recognition (ICPR)*, 2014, pp. 4513-4518.
- [58] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3d skeletons as points in a lie group," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 588-595.
- [59] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1110-1118.
- [60] Z. Huang, C. Wan, T. Probst, and L. Van Gool, "Deep learning on lie groups for skeleton-based action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6099-6108.
- [61] J. Liu, A. Shahroudy, D. Xu, A. K. Chichung, and G. Wang, "Skeleton-based action recognition using spatio-temporal lstm network with trust gates," *IEEE transactions on pattern analysis and machine intelligence*, 2017.
- [62] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2017, p. 7.
- [63] L. Xia, C.-C. Chen, and J. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012, pp. 20-27.
- [64] J. Wang, Z. Liu, and Y. Wu, "Learning actionlet ensemble for 3D human action recognition," in *Human Action Recognition with Depth Cameras*, ed: Springer, 2014, pp. 11-40.
- [65] A. Shahroudy, T.-T. Ng, Y. Gong, and G. Wang, "Deep multimodal feature analysis for action recognition in rgb+ d videos," *IEEE transactions on pattern*

- analysis and machine intelligence*, 2017.
- [66] H. Rahmani and M. Bennamoun, "Learning action recognition model from depth and skeleton videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5832-5841.

