ABSTRACT

Title of thesis: HUMAN ROBOT INTERACTION
ON GESTURE CONTROLLED DRONE:
METHODS OF GESTURE ACTION RECOGNITION

Siqin Li
Master of Science, 2018

Thesis directed by: Professor Yiannis Aloimonos
Department of Computer Science

Today, the interaction between robots and human is mostly based on remote controller. However this interaction could be more natural, just like we humans interact with each others through speech, body movements, facial expressions, and so on. We propose gesture and body language as an alternative to interact with robots, particularly Unmanned Aerial Vehicles(UAVs) also known as drones. In this thesis, we developed action recognition methods for the interaction with drones. Specifically, we developed approaches to recognize human gestures for the communication with drones.

Automatic detection and classification of dynamic actions in real-world system intended for human robot interaction is challenging because: *1)* there is large variation in how people perform actions, making detection and classification difficult; *2)* the system must work online in order to avoid a noticeable delay in the interaction between the human and the drone. In this work, we address these challenges through the combination of a real-time skeleton detection library and deep learning

techniques. Our methods perform dynamic actions or gestures classification from skeleton data.

HUMAN ROBOT INTERACTION ON GESTURE CONTROLLED DRONE: METHODS OF GESTURE ACTION RECOGNITION

by

Siqin Li

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2018

Advisory Committee:
Professor Yiannis Aloimonos, Chair/Advisor
Professor Gang Qu
Dr. Cornelia Fermuller

© Copyright by
Siqin Li
2018

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Yiannis Aloimonos for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past year. He has always made himself available for help and advice. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank my co-advisor, Dr. Cornelia Fermuller. Without her extraordinary theoretical ideas and computational expertise, this thesis would have been a distant dream. Thanks are due to Professor Gang Qu for agreeing to serve on my thesis committee and for sparing his invaluable time reviewing the manuscript.

My colleagues at the Autonomy Robotics Cognition laboratory and Computer Vision laboratory have enriched my graduate life in many ways and deserve a special mention. They provided me a relax and pleasant working environment.

I would also like to acknowledge help and support from all of my friends. My discussion and interaction with them has been very fruitful. Without their encouragement, this thesis can't be happened in the past year.

I owe my deepest thanks to my family - my mother and father who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them.

It is impossible to remember all, and I apologize to those I've inadvertently

left out.

Lastly, thank you all.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

TCN	Temporal Convolution Neural Network
Res-TCN	Temporal Convolution Network with Residual Connection
UAV	Unmanned Aerial Vihecle
HRI	Human Robot Interaction

Chapter 1: Introduction

Human-Robot Interaction (HRI) has recently received considerable attention in the academic community, in labs, in technology companies, and through the media. Interaction, also know as communication between human and robot, can be split into two general categories: remote interaction and proximate interaction. These two forms are influenced by whether the human and the robot are in close proximity to each other or not. In this thesis, we decide to develop a remote interaction system between a human and an Unmanned Aerial Vehicle (UAV).

Different from the traditional mouse and keyboard, recent trends on user interface, including multi-touch interfaces and camera-based systems, have gained significant popularity among consumers. They made the interaction between human and computer more natural and effortless. Most of them allow users to use their hands or body actions to directly manipulate virtual objects which is very similar to how humans interact with each other.

Considering the significant potential and demand for natural interaction, we propose using gesture and body language as the purveyor passing information from human to UAVs. Instead of recognizing static gesture and body pose in the old paradigm, our interactions are more versatile and can recognize a sequence of ges-

tures and the trajectory. In this thesis, we addressed the following questions: What types of gestures do people conduct? What are the important features needed for understanding a gesture? How to detect and track on the human body? How should a recognition model be designed and trained?

1.1 Background

To design a gesture controlled system, it is important to understand the definition and form of human gestures. In this section, we are going to discuss the background of gesture research and the different gesture categories.

1.1.1 Definition of Gesture and Gesture Recognition

According to the description in [18], gesture is a form of non-verbal communication or non-vocal communication in which visible bodily actions communicate particular messages, either in place of, or in conjunction with, speech. However, in human robot interaction, human gestures are mostly designed as sign language carrying different meanings. Users can use simple gestures to control or interact with devices without physically touching them.

Gesture recognition is a topic in computer science and robotics with the goal of interpreting human gestures via mathematical algorithms. Many approaches have been developed using cameras and computer vision algorithms. With the popularity of deep learning and neural networks, modern computer vision techniques heavily rely on deep learning to learn from a huge dataset and find patterns which can then

make predictions. In this thesis, deep learning is the primary technique used to solve the problem.

1.1.2 Gesture Category

A way to distinguish gesture is to differentiate between communicative gestures and informative gestures. Communicative gestures are those gestures produced with intentionality of meaning. Speakers consciously use them to communicate something about themselves or someone else. However, informative gesture are passive gestures that the speaker is not about to communicate with. These gestures can occur during speech, but they may also occur independent of communication. In computer interfaces, we consider communicative gesture.

As an emerging type of technology in relation to gesture control, touchless user interface is the process of commanding the computer or robot without physically touching a device. A number of applications have started utilizing this type of interface mostly in gaming software (e.g. Microsoft's Kinect). We believe that establishing such a interface between computer vision and robotics will be a promising application.

1.2 System Overview and Thesis Outline

Our gesture control system consists of two modules: the hand and body pose detection and the gesture action recognition (Figure 1.1). At each time step, the hand and body pose detection module takes a frame from the video as input, then

estimates the human body and hand location, and outputs the keypoints of the body and hand joints to the recognition module. The gesture recognition module estimates the current most likely gesture label based on the input stream of keypoints.

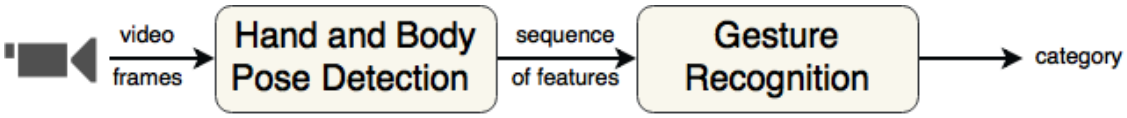

Figure 1.1: System Overview

In the detection module, we adopted a powerful open-source library which provides the body keypoints with accurate and online detection performance. In the recognition module, we investigate a state-of-the-art network, then come up with two different networks to improve the gesture recognition accuracy and user experience. The main focus and contributions of this work are the action and gesture recognition.

1.3 Contribution

The main contribution of this thesis include:

- I created a new dataset called "helicopter marshalling" dataset that includes 2D skeleton sequences for helicopter marshelling. This dataset can be used to evaluate our system and other gesture recognition systems, and it will be used to study drone communication and control in the future.

- I developed two different techniques to recognize human gesture from 2D input which were shown to outperform the state-of-the-art algorithms taking 3D information as input.

thesis

Chapter 2: Related Work

There are three essential components in the gesture control pipeline: input modality, keypoint detection, gesture action recognition. This section discusses some literatures on recent developments in each area.

2.1 Input Modality

The first step in the pipeline is to decide on the input to the system. The sensor used to capture information is closely related to the project's objective and the system performance.

The work from [21] uses stereo RGB cameras to detect hands from background based on skin color and input from registered 3D data. Oka et al. [22] use thermal images for hand segmentation in complex backgrounds relying on the hand's temperature which is always distinct from the background.

In recent years, a number of innovative sensors such as depth camera and the Leap Motion sensor that provides 3D data of the scene, have contributed a lot to body detection and action recognition. Depth cameras allow to obtain a complete 3D description of the scene while the Leap Motion sensor is a device explicitly targeted for hand gesture recognition and provides only a limited set of relevant

points. Molchanov et al. [23] use multi-modal dynamic hand gesture data captured with depth, color and stereo-IR sensors to achieve online detection and classification.

Considering the feasibility of putting cameras on UAVs, we decided to use a regular RGB color camera as sensor for capturing the scene. The input to our system are RGB videos. We demonstrate that the system is competitive with those using extra sensors.

2.2 Hand Detection

The next step in the pipeline is to detect the human bodies and hands in the scene. This step is essential for for the recognition pipeline. However, vision-based hand detection is challenging due to the wide range of configurations and appearances it can assume and possible occlusions.

Hand detection has been studied by a number of works. For instance, Van den Bergh et al. [24] introduced a novel hand detection algorithm based on depth and color. Kölsch et al. [25] presented a frequency analysis-based method for instantaneous estimation of class separability. In [26], Ohn-Bar et al. proposed a framework that provides a robust hand localization by partitioning visible and depth images ito disjoint sub-regions. Later, Mittla et al. [27] proposed a detector using a two-stage hypothesis and classificatoin framework. This method is able to detect hands and their orientation in unconstrained images.

However, the above works mostly focus on detecting the hand palms excluding arms and limbs. In real life, there is no way to attain gesture interaction without

moving arms and limbs. Thus, we consider human upper body detection of significance in this scenario.

The recent development of deep learning based approaches revolutionized research on visual detection. The domain of face detection and pedestrian detection are well researched. A number of works have started solving one of the longest-lasting problems in computer vision, articulated body pose estimation, which aims at recovering the pose of an articulated body. Body pose estimation has applications in many areas including assisted living, character animation, intelligent driver assistance systems and video games. Human pose estimation is closely related to hand detection. The direction pointing from the elbow to the wrist and the length of the fore arm give us adequate knowledge to infer the direction and the size of the hands. Spurred by the MPII and COCO human pose benchmarks, human pose estimation has been advanced significantly in recent years. The convolutional neural network architecture of Toshev et al. [28] regresses 2D cartesian coordinates from color images as input directly. Work from [14] regresses keypoint score maps. More recently, novel convolutional neural network architectures such as the stacked hourglass [29] have been proposed to capture the various spatial relationships associated with a human body. Zimmermann et al. [30] employed the body pose estimation technique in [14] to detect hand poses. We use an open source library called OpenPose that can achieve real-time human pose estimation as well as hand pose estimation on RGB frames.

2.3 Gesture Recognition

Many previous works on gesture recognition have focused on single categories. Gesture can be static or dynamic, and some gestures have both static and dynamic elements, such as sign languages. There are different tools for gesture recognition. Based on statistical modeling, some approaches such as PCA, HMMs [31], [32], Kalman filter, particle filter [34] [35], and condensation algorithms have been employed. Computer vision and pattern recognition techniques, involving feature extraction, object detection, clustering, and classification have been successfully used for many gesture recognition systems.

Deep learning enabled researcher to explore many approaches to action recognition. Hierarchical recurrent neural network of [8] combines features of different body parts hierarchically. The work by [16] introduces a part-aware LSTM model, body joints are grouped together based on their spatial context. The work of [9] leverages on a similar intuition, namely that co-occurrence of joints is a strong discriminative feature. [5] proposed a version of a temporal convolutional network that learns both spatial and temporal attention. These methods demonstrate the success of deep networks on skeleton based action recognition.

Chapter 3: Datasets

In this chapter, we describe the datasets used to perform the experiments in this thesis.

3.1 **NTU-RGBD Dataset**

3.1.1 Introduction

The NTU-RGBD dataset [16] is a publicly available large dataset containing recordings of labeled human activities. This dataset consists of 56,880 action samples containing 4 different modalities (RGB videos, depth map sequences, 3D skeletal data, infrared video) for each sample. Video samples have been captured by 3 Microsoft Kinect v.2 cameras concurrently.

3.1.1.1 Data Modalities

As being introduced before, the dataset consists of four major data modalities: depth maps, 3D joint information, RGB frames, and IR sequences. Depth maps are sequences of two dimensional arrays of depth values in millimeters. The 3D joint information consists of the spacial location of 25 major body point, and the

corresponding pixels on RGB frames and depth maps are also provided for each joint and every frame.

3.1.1.2 Action Classes

There are 60 action classes in total, which are divided into three major groups: 40 daily actions, 9 health-related actions, and 11 mutual actions.

3.1.1.3 Subjects

40 distinct subjects participated in the data collection. The ages of the subjects varied from 10 to 35 years.

3.1.1.4 Views

This dataset was collected by three cameras capturing three different horizontal views at the same time. The three cameras were at the same height but pointing with different angles: $-45°$, $0°$, $45°$. Subjects were asked to perform each action twice facing towards the left and the right cameras separately. For each subject and every action, the data has two front views, one left side view, one right view, one left side $45°$ view and one right side $45°$.

3.1.2 Benchmark Evaluations

The NTU-RGBD dataset provides a standard evaluation protocol for all the reported results. There are two precise criteria for two types of action classification

evaluation, the classification accuracy should be reported in percentage.

3.1.2.1 Cross-Subject Evaluation

In the cross-subject evaluation, the 40 subjects are split into training and testing groups. Each groups consists of 20 subjects. In this thesis, since we only consider the skeleton data that can be detected on the RGB frames, the training and testing sets have 39,315 and 16,148 samples respectively.

3.1.2.2 Cross-View Evaluation

In the cross-view evaluation, all the samples captured by camera 1 are used for testing and samples from cameras 2 and 3 are used for training. Thus the training set consists of front and two side views of the actions, while the testing set includes left and right 45° view of the actions. For this evaluation, the training sets have 36,838 samples, the testing sets have 18,625 samples.

3.2 **Helicopter Marshalling Dataset**

As this project is going to be implemented on unmanned aerial vehicles with several command gestures, we collected a new dataset named Helicopter Marshalling dataset which we generated from the ChaLearn dataset. This dataset contains 9 classes of helicopter signal gestures: $HoldOver$, $Land$, $LiftOff$, $MoveDownward$, $MoveForward$, $MoveLeft$, $MoveRight$, $MoveUpward$, $ReleaseSlingLoad$. The training sets, validation sets and test sets have 1399, 200 and 300 samples respec-

tively.

3.2.1 Description of the ChaLearn Dataset

The ChaLearn dataset [17] consists of two large video multi-modal datasets for RGB and RGBD gesture recognition. One is the Chalearn LAP RGB-D Isolated Gesture Dataset (IsoGD) and the other is the Continuous Gesture Dataset (ConGD). The complete dataset has a total of 249 gesture labels. In this thesis, we only use the IsoGD dataset. There are 47,933 samples. Each RGB-D video features one gesture instance, and the gesture actions are performed by 21 different individuals.

3.2.2 Benchmark Evaluation of ChaLearn

The ChaLearn dataset provides training, validation and test sets. All three sets include data from different subjects, which means that the gestures of a specific subject in the validation and test sets will not appear in the training set. For the IsoGD dataset, a recognition rate r is used as the evaluation criteria which is calculated as:

$$r = \frac{1}{n} \sum_{i=1}^{n} \delta(p_l(i), t_l(i)) \tag{3.1}$$

where n is the number of samples; p_l is the predicted label; t_l is the ground truth; $\delta(j_1, j_2) = 1$, if $j_1 = j_2$, otherwise $\delta(j_1, j_2) = 0$.

3.2.3 Data Collection Procedure

The IsoGD dataset provides us a set of helicopter signals as RGB videos. To generate the 2D skeleton data, we first extracted the helicopter subset from the IsoGD dataset. Secondly, we run OpenPose on each of the RGB video frames and annotated each frame with the video label. Since our task is to recognize gestures, we only require the upper body skeleton and the hand keypoints for classification. Thirdly, we sort out the data with a specific format. The details of the data will be described in the next section.

3.2.4 Data Formats

The data from the OpenPose library is recorded in form of 2D image coordinates for each body parts. The skeleton data contains the joint positions for 18 body keypoints and 2×21 hand keypoints. The order of the body part defined by the COCO keypoints is described in Fig 3.1 (left). We extract $0 - 7$ keypoints for the upper body. The hand keypoints are described in Fig 3.1 (right). We simply concatenate those keypoint coordinates into a vector.

3.2.5 Evaluation Protocol

Since this dataset is obtained from the ChaLearn Dataset, we inherited the protocol provided in ChaLearn. The accuracy in percentage is used to evaluate the system performance.

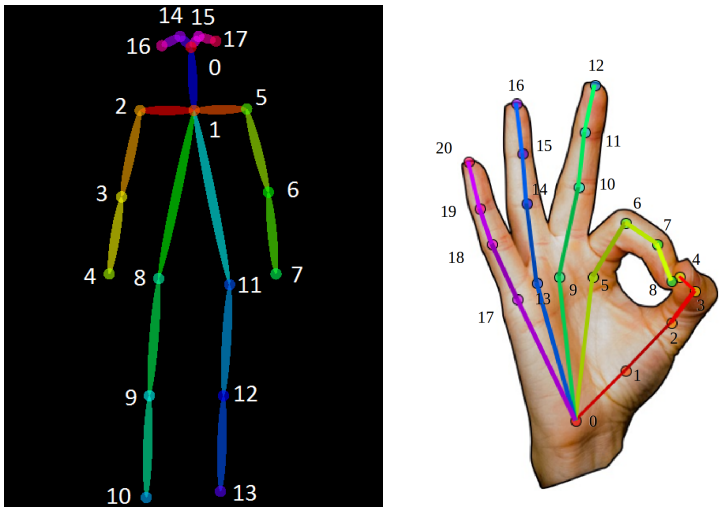

Figure 3.1: Output Format

Left: Pose Output Format; *Right*: Hand Output Format

Chapter 4: Human Pose Estimation and Hand Pose Detection

Human pose estimation is the process of estimating the configuration of the body pose from a single image. Human pose estimation is a key problems in computer vision that has been studied, and many applications can benefit from such a technology.

In this project, the detection and estimation for human body and hand keypoint is of significance. Instead of recognizing human action from image frames, skeleton-based action recognition is more effective and well-behaved, because skeleton keypoint coordinates can be a type of very crucial representation of human pose and movement. In the next section, a powerful techniques adopted here will be analyzed in detail. The evaluation for the performance is discussed later.

4.1 **Introduction to OpenPose Library**

OpenPose is a library that implements a real-time system to jointly detect human body, hand, and facial keypoints in single images. It is able to detect the 2D pose of multiple people efficiently. The algorithm and implementation of the OpenPose system will be discussed here.

4.1.1 Human Pose Estimation

According to the introduction in [13], there are two broad approaches for tackling the multi-person pose estimation problem. *Bottom-up*, in which keypoint proposals are detected and grouped together into person instances, and *top-down*, in which a pose estimator is applied to the bounding box of each detected person. The approach of Cao et al. [12], which is adopted by OpenPose advocates the bottom-up approach. It combines a variation of the unary joint detector architecture from convolutional pose machines [14] with a part affinity field regression to enforce inter-joint consistency. Then they employ a greedy algorithm to generate person instance proposals in a bottom-up fashion. Figure 4.1 illustrates the overall pipeline of this method.

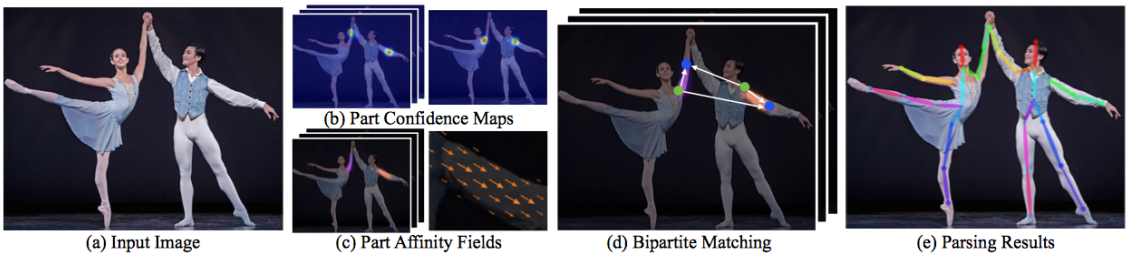

Figure 4.1: Overall pipeline for human pose estimation

(a) The entire image as the input for the network (b) predict confidence maps for body part detection (c) part affinity fields for parts association (d) using bipartite matchings to parse associated body part candidates (d) final output for all people in the image

4.1.1.1 Architecture and Details

The idea of this network architecture is to predict detection confidence maps and affinity fields simultaneously. The network consists of two branchesthesis, as shown in Figure 4.2. The upper branch predicts the confidence maps and the lower branch predicts the part affinity fields. Each branch is an iterative architecture to refine the predictions over successive stages. Intermediate supervision is adopted between every two stages.

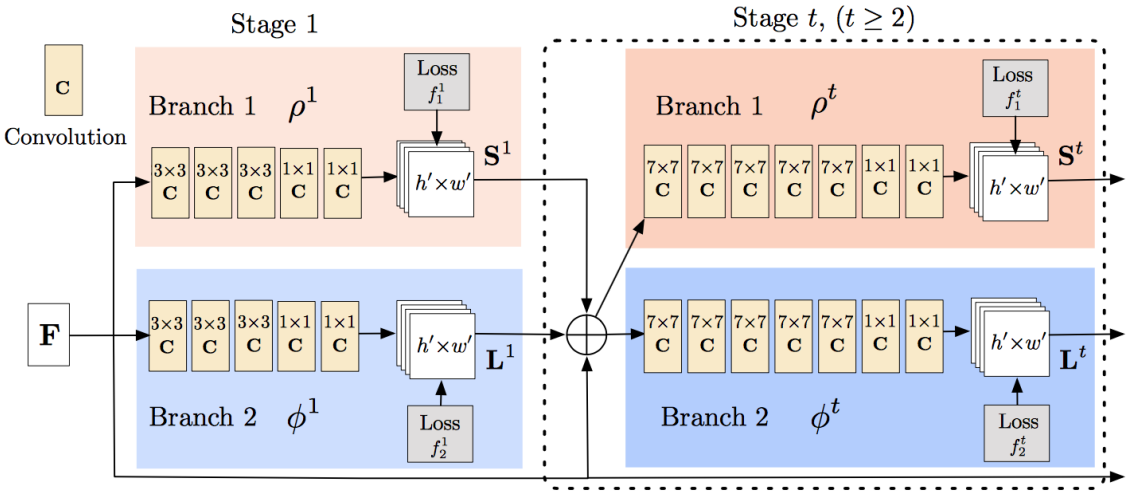

Figure 4.2: Architecture of the two-branch multi-stage CNN

The confidence map is a 2D representation of the probability that a particular body part occurs at each pixel location. To generate the ground truth confidence map for a part, the map is formulated as a Gaussian distribution centered at the true position. At testing time, the body part candidates can be obtained by performing non-maximum suppression on predicted confidence maps.

After the body part locations are provided, the part affinity fields (PAF) is

computed assemble the body part locations. The part affinity field is a 2D vector field for each limb. It can preserve both location and orientation information across the region of support of the limb. Specifically, each pixel in the area belonging to a particular limb will be assigned a 2D vector encoding the direction from one body part to the others. The integral over the PAF measures provides the association between two body parts. During testing, the predicted PAF with the possible connected limb is aligned and measured.

Once the measurement is defined, in this scenario, parsing multiple people in the image is solved as a K-partite graph matching problem which is NP-hard. Later, a greedy relaxation that consistently produces high quality matches is adopted. With two relaxations, the optimization problem is simplified. Therefore, all limb connection candidates can be assembled into full-body poses of multiple people.

4.1.2 Hand Keypoint Estimation

For the hand keypoint estimation, OpenPose implements a robust approach introduced in [15]. This approach is called multiview bootstrapping: which uses a multi-camera system to train fine-grained detectors for keypoints that are prone to occlusion, such as the joints of a hand. This approach boost the performance of a given keypoint detector. It allows a weak detector, trained on a small annotated dataset, to localize subsets of keypoints in good views and use robust 3D triangulation to filter out incorrect detections. It generates an approach which can label images that are difficult to annotate due to occlusion in single views. As for the

detection architecture, the hand bounding box detection is the first step. The hand location can be approximately cropped from the extension along the forearm direction. The position of the wrist and the elbow is decided by the approach described above. After the bounding box is cropped out, the work follows the approach introduced in [14], which can generate a confidence map for each keypoint, representing the keypoint's location as a Gaussian centered at the true position. The location of each keypoint is produced by locating the maximum peak in each confidence map.

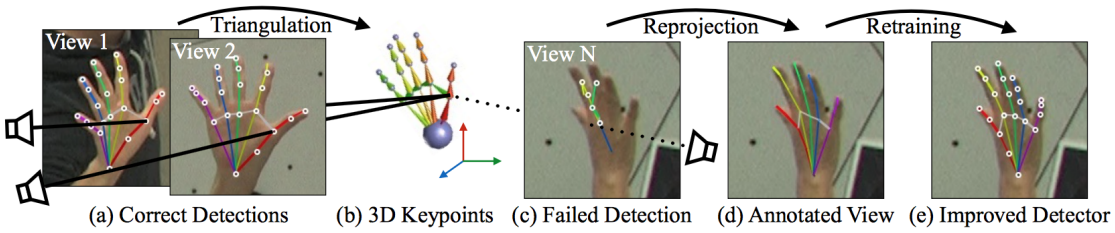

Figure 4.3: Multiview Bootstrapping

(a) A multiview system provides views of the hand (b) the 3D position of the keypoints (c) failure case (d) annotated failure case by 3D reprojection (e) retrained and improved detector

4.2 Performance Evaluation

This network is evaluated on two benchmarks for pose estimation. It won the COCO 2016 keypoints challenge. Here we analyse the running time.

4.2.1 Runtime

According to paper [12], the runtime is due to two major parts: one is the CNN processing time $O(1)$ which constant with varying number of people and another one is the multi-person parsing time $O(n^2)$, n is the number of people in the scene. OpenPose running on one NVIDIA GeForce GTX-1080 GPU can achieve 8.8fps for a video with 19 people

Chapter 5: Recognition Methodology

Human activity and gesture action analysis is a crucial yet challenging task in computer vision. Applications of human activity and gesture action recognition range from video surveillance over human computer interaction to robotics and skill evaluation. As mentioned in related work, most prior work focuses on recognizing 3D skeleton based action sequences. Since accurate depth information can be obtained only in a limited range of distance, and the agility of the unmanned aerial vehicle is very sensitive to the weight and the size of the camera, we decided to develop a framework for recognizing action from commercial RGB cameras.

In this research, three different kinds of architectures are implemented which are a Temporal Convolutional Network with residual connections (Res-TCN), Multi-Modal network with modality hallucination, and an LSTM network. This chapter gives details about these architectures.

5.1 Temporal Convolutional Network with Residual Connections

Many approaches have been proposed for the task of understanding human actions from video. Temporal Convolutional Network (TCN) [1] is one of them,

which has been proposed to solve the action segmentation problem in an RGB video. As an extension or modification of the TCN, Res-TCN [5] is a new framework to deal with human action recognition using as input skeleton sequences from RGB-D sensors. Instead of taking 3D skeletons input from RGB-D sensors, we propose to feed into the network with the raw skeleton pose detected from RGB frames. This architecture demonstrates that for most of the human action recognition tasks, depth information is redundant as it does not lead to improved performance.

5.1.1 Overview of Temporal Convolutional Neural Networks

In this section, we briefly introduce the structure of a Temporal Convolution Network (TCN) as provided in the original paper [1]. The original TCN is designed for temporal action segmentation in video and it follows a convolutional encoder-decoder design. This approach uses a single set of computational mechanisms - 1D convolutions, pooling, and channel-wise normalization - to hierarchically capture low-, intermediate- and high-level temporal information. Here, we adopt the encoder portion of the network for action recognition.

The input to an original TCN is a sequence of video features. Let $X_t \in \mathbb{R}^{F_0}$ be the input feature vector of length F_0 for time step t for $t \leq T$. Note that T is the number of the frames in a video and it may vary for each sequence. Denote the number of time steps in each layer as T_l. The true action label for each frame is $y_t \in \{1, ..., C\}$, where C is the number of classes.

In the part of the encoder, there are L convolutional layers. A set of 1D filters

is applied, which captures how the input signals evolve over the course of an action. The filters for each layer are parameterized by tensor $W^{(l)} \in \mathbb{R}^{F_l \times d \times F_{l-1}}$ and biases $b^{(l)} \in \mathbb{R}^{F_l}$, where $l \in 1, ..., L$ is the layer index and d is the filter length. Given an output X_{l-1} of the previous layer, the l-th layer output, X_l is

$$X_l = \sigma(W^{(l)} \star X_{l-1}), \tag{5.1}$$

Note here, the j-th feature vector in $W^{(l)} \star X_{l-1}$ can be represented as:

$$(W^{(l)} \star X_{l-1})_j = \sum_{i=1}^{F_{l-1}} X_{l-1,i} * W_{i,j}^{(l)} + b^{(l)}, \tag{5.2}$$

where σ is a non-linear activation function ReLU. The whole network is trained with back-propagation. In an attempt to further improve the interpretability of TCN, the residual connections architecture of [4] is adopted.

5.1.2 Architecture of TCNs with Residual Connections (Res-TCN)

In the work of [5], the original TCN is re-designed by factoring out the deeper layers into additive residual terms which yields both interpretable hidden representations and model parameters. For the purpose of 3D human activity recognition, the input to the model X_0 are the frame-wise 3D skeleton features concatenated temporally across the entire RGB-D video sequence. Let $x_t \in \mathbb{R}^{F_0}$ be the skeleton features extracted from an RGB-D video frame t. $x_t(d)$ is the d-th dimension of the feature with the interpretable meaning associated with it. Res-TCN stacks building blocks called *Residual Units* as introduced in [4] and adapts the pre-activation

scheme of [6]. Each unit in layer l performs the following computation:

$$X_l = X_{l-1} + F(W^{(l)}, X_{l-1}) \tag{5.3}$$

$$F(W_l, X_{l-1}) = W^{(l)} \star \sigma(X_{l-1}), \tag{5.4}$$

F denotes the residual unit. For the l-th layer, X_{l-1} denotes the input, W_l is the set of learnable parameters and σ is a ReLU activation function. In the first layer, Res-TCN takes raw skeleton as input and generates the activation map X_1 which is then passed to the later layers. For a Res-TCN with N residual units, the hidden representation after N residual units is:

$$X_N = X_1 + \sum_{i=2}^{N} W^{(i)} \star \sigma(X_{i-1}) \tag{5.5}$$

$$X_1 = W^{(1)} \star X_0, \tag{5.6}$$

where $W^{(1)}$ represents a set of filters applied to the raw skeleton input X_0. X_1 is the resulting activation map. The operation $\star$ can be seen as a 1D convolution over a tensor and a matrix. The details of how it works will be discussed in the next section.

For classification, average pooling over the temporal sequence is applied and a softmax layer with the number of neurons equal to the number of classes, is attached.

5.1.3 Model Parameters Analysis

Res-TCN has been shown to perform well on the large scale NTU-RGBD dataset. To better interpret the information in a 2D sequence, we adjust the original

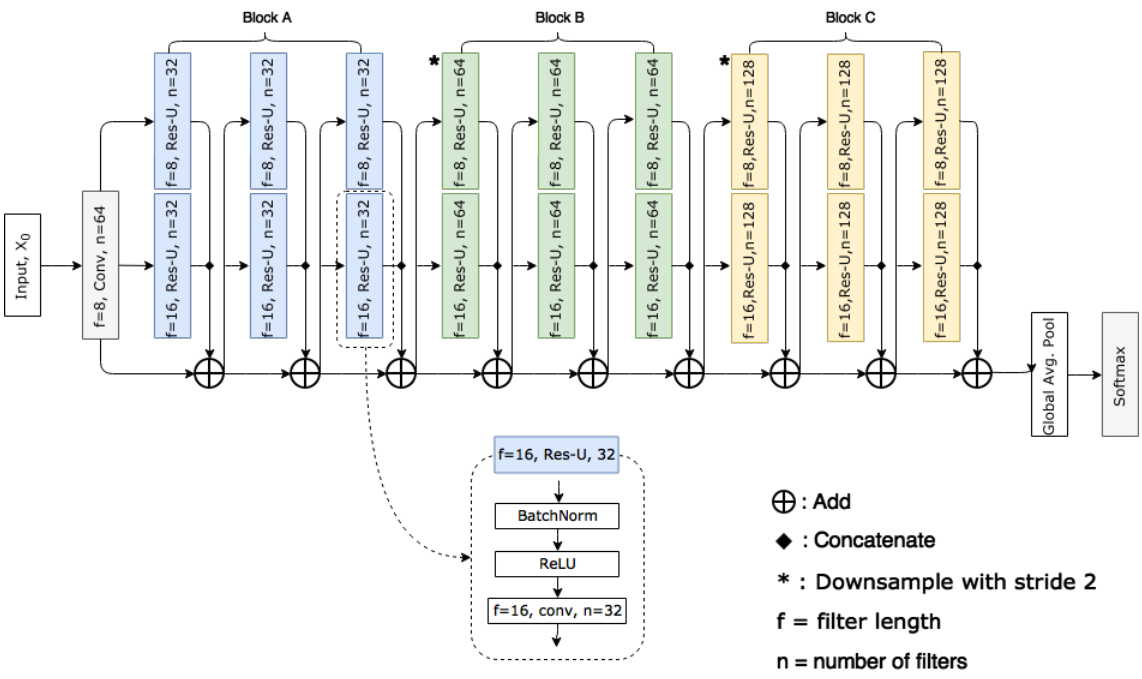

Figure 5.1: Res-TCN model architecture

Res-TCN architecture. The modified model architecture is shown in Figure 5.1. In this section, we analyze each filter and what it represents.

Consider the first convolution layer $W^{(1)} \in \mathbb{R}^{f_l \times F_0 \times F_1}$, with F_1 filters $W_j^{(1)} \in \mathbb{R}^{f_1 \times F_0}$, $j = 1, ..., F_1$. Each of the filters consists of F_0 1D filter elements $W_{j,i}^{(1)} \in \mathbb{R}^{f_1}$, $i = 1, 2, ..., F_0$ where f_1 is the length of the 1D filter elements. $W_j^{(1)}$ computes 1D convolution over $X_0 \in \mathbb{R}^{T \times F_0}$. We denote this 1D convolution operation on the input feature map as $\star$.

$$X_1 = W^{(1)} \star X_0 \tag{5.7}$$

for the j-th feature vector $X_{1,j}$ can be considered as:

$$X_{1,j} = \sum_{i=1}^{F_0} W_{j,i}^{(1)} * X_{0,i} + b_j \mathbf{1}, \;\; j = 1, 2, ..., F_1 \tag{5.8}$$

where $\mathbf{1}$ is the identity vector in $\mathbb{R}^T$.

An important property of $W^{(1)}$ is, for each feature dimension $d \in F_0$, it can learn the explainable meaning associated with it, since the d-th dimension of the skeleton feature $x_t(d)$ represents a spatial configuration of a particular joint at time t.

Let us now move our analysis to deeper layers in the model. In a Res-TCN formulation, we adopt residual learning to every stacked deeper layer. Formally, we consider the l-th layer with input X_{l-1} and output X_l. The residual unit can be defined as:

$$X_l = F(W^{(l)}, X_{l-1}) + X_{l-1} \tag{5.9}$$

This operation $F + X_{l-1}$ is performed by a shortcut connection and element-wise addition. For example, consider the two sets of residual unit in the second layer, they both take the feature maps $X_1 \in \mathbb{R}^{T\times 64}$ as input, and generate two sub-feature-maps F' and $F'' \in \mathbb{R}^{T\times 32}$. The full feature map is the concatenation, $F = [F' \quad F'']$. The second layer output can be simply computed by element-wise addition $X_2 = F + X_1$. This technique can avoid the vanishing gradient problem during optimization.

5.1.4 Experiments and Evaluation

This network architecture is evaluated on the skeleton based human activity recognition dataset of NTU [16]. And this approach is validated as a both interpretable and discriminative.

5.1.4.1 *Implementation Details*

We follow the skeleton feature extraction procedure as introduced in [16]. The original dataset contains 3 dimensinal locations of 25 major body joints in the scene. The corresponding pixels on RGB frames and the depth maps are also provided for each joint and every frame. Even though the Res-TCN architecture is designed for 3D skeleton based action recognition, our goal is to recognize actions from only RGB frames sequences. Thus, in order to evaluate the robustness and generality, we trained the network on both 3D skeleton data and 2D skeleton data.

For 3D skeleton data, we normalize the raw (X, Y, Z) values of each skeleton joint and concatenate all values to form a skeleton feature as a vector per frame. Knowing that there are at most 2 subjects in the scene and there are 25 joints per skeleton, a skeleton feature is a $2 \times 25 \times 3 = 150$ dimensional vector. Then we ran the deep learning framework on Keras [19] on top of Tensorflow [20]. We use the base learning rate of 0.01, and decrease the learning rate by a factor of 10 when the testing loss plateaus for more than 3 epochs. To avoid overfitting, we add a dropout layer of rate 0.3 after each activation layers. We use stochastic gradient descent and nesterov acceleration with a momentum of 0.9. For all convolution layers, we apply an L-1 regularizer with a weight of $1e^{-4}$. The experiments are run on two Nvidia GeForce GTX TITAN X GPUs.

For 2D skeleton data, we perform the same preprocessing steps as the 3D data. Since the dimensionality is reduced, a skeleton feature per frame is a $2 \times 25 \times 2 = 100$ dimensional vector.We set the base learning rate as 0.01 and decrease it by a factor

of 5 with 10 epochs plateaus. We use 0.5 dropout rate, SGD with 0.9 momentum, an L-1 regularizer with $1e^{-4}$ weight.

5.1.4.2 *Performance Evaluation*

The reason that the Res-TCN formulation yields explainable spatio-temporal representation is elaborated in [5]. And it is validated that the model is able to produce discriminative spatio-temporal features for 3D human action analysis. Here, we compare the performance on 2D action versus 3D action in Table 5.1.

	Cross-Subject	Cross-View
3D	76.1%	77.1%
2D	73.0%	73.5%

Table 5.1: Comparison of 2D and 3D input on NTU RGB-D dataset with Cross-Subject and Cross-View settings in accuracy (%)

Even though the results with 3D input still outperform the one with 2D input, we find that 2D input gives an unexpectedly good performance. This means depth that can provide only limit improvement in this classification settings with 60 categories.

5.1.5 Conclussion

In this section, we discussed the architecture of Res-TCN and explain the interpretability of model parameters. We compared the performance on two types

of inputs format and showed that there is room to enhance action recognition on RGB video sequences.

5.2 Multi-modal network with modality hallucination

Although Res-TCN performs excellently on 2D skeleton data, we cannot deny that 3D data does provide more useful information. Another question came to our mind: can we learn the depth information from RGB image? The intuition behind is quiet straight-forward as humans appear to infer depth from RGB images using experience. Thus, spurred by this idea, a hallucination model is proposed to improve the RGB performance by learning the depth information at the training stage.

5.2.1 Overview of the Hallucination Model

2D skeleton data and depth skeleton data offer different and often complementary information. The fusion of multimodal data in networks has been successfully demonstrated in various applications, including object recognition, object detection and segmentation. However, it is challenging to collect more than two modalities of data for the same action. Since RGB image capturing devices are very common, but depth capturing devices are much less prevalent. This means that many recognition tasks will need to perform well using only RGB images. Thus, we adopt a modality hallucination architecture for the 2D skeleton based action recognition model which incorporates depth side information at training time. The convolutional hallucination network takes the 2D skeleton data as input and is taught to

mimic the convolutional features from the 3D network. The multimodal data will be fused to classify the action labels. At testing time, the 2D skeleton data will be jointly processed through the 2D network and the hallucination network to improve the action recognition performance.

5.2.2 Network Architecture

In this section we are going to describe the deep network architecture, which uses temporal convolution network to learn a 3D representation from the 2D skeleton data.

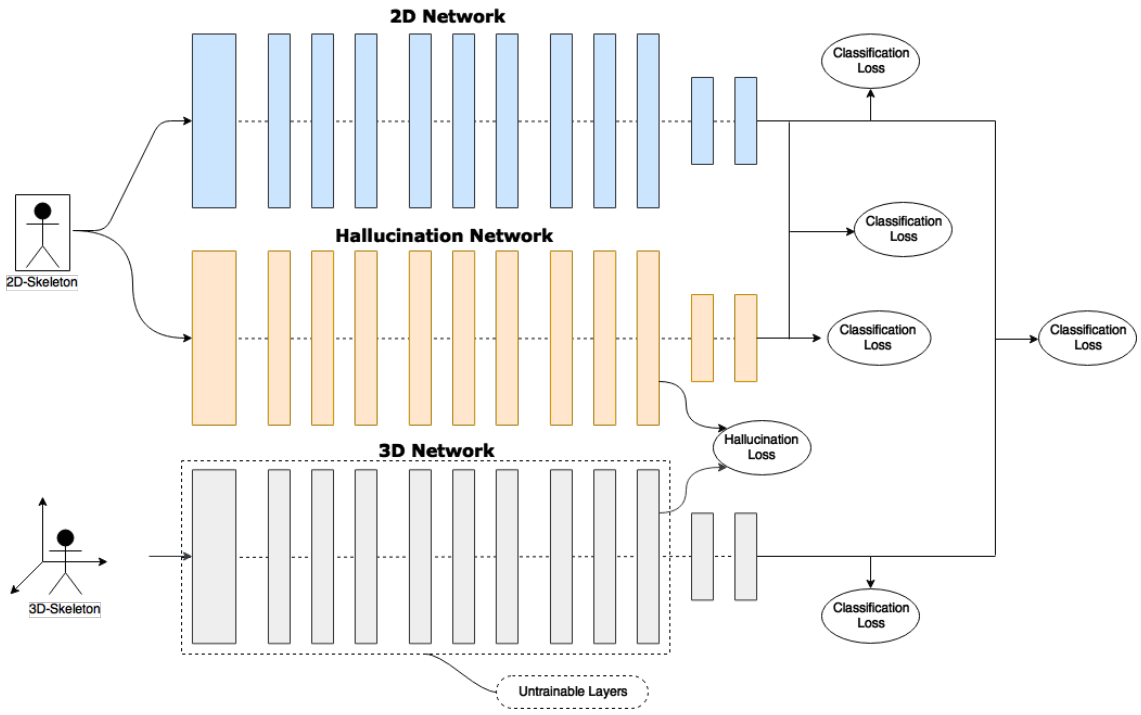

Figure 5.2: Hallucination model architecture

Figure 5.2 illustrate the training architecture for the hallucination model. The network has 3 sections: a 2D network, a hallucination network and a 3D network.

For simplicity and efficiency, we choose Res-TCN as our base network architecture to extract feature representations from 2D data or 3D data and do the classification respectively. In this architecture, the hallucination network is taught to learn the relationship between the other two networks. It mimics the 3D mid-level features during the training stage. To force the 3D module to share information with the 2D module through this hallucination network, we add a regression loss between the paired hallucination and 3D layers, in order to minimize the difference between their representations. Thus, the hallucination network is able to generate representations in the higher layers similar to the 3D network. Notice that the parameters of the hallucination network are independent of both the 2D network and the 3D network. The details about choosing the regressing loss and the mid-level feature extraction layer will be discussed in the next section.

In the testing stage, the 3D module can be removed when only 2D skeleton input is available. We feed the 2D skeleton input to both the 2D network and the hallucination network to produce two scores, then the two scores are fused and we take the softmax to get the final decision. The testing model can be found in Figure 5.3.

5.2.3 Implementation and Optimization

In this section, we introduce the implementation and optimization details for this architecture. At the training stage, we are given access to 2D and 3D input pairs and category labels for the skeleton sequences. We trained the 2D network

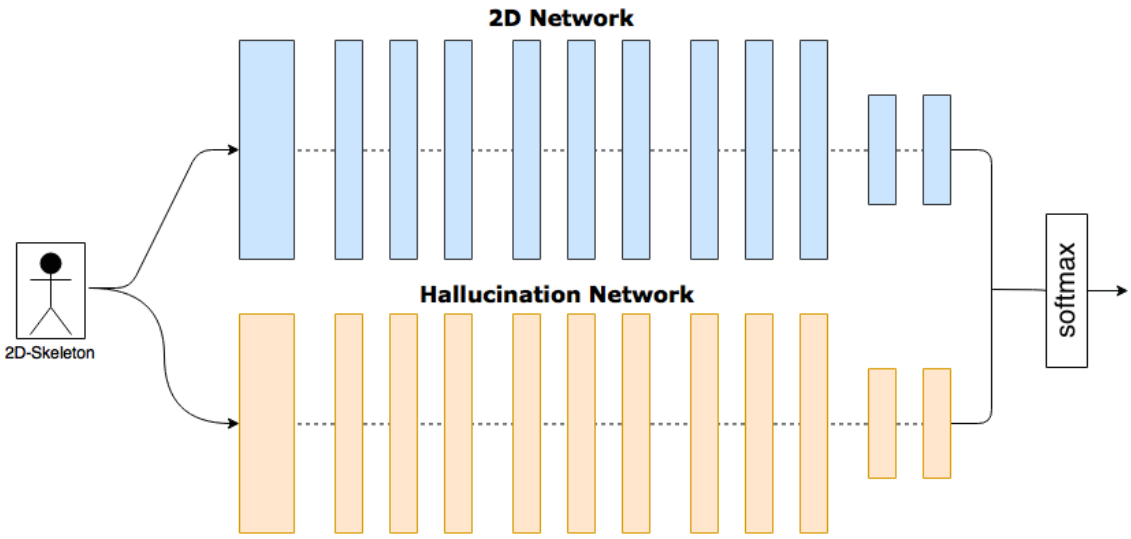

Figure 5.3: Testing Stage Hallucination Architecture

and the 3D network independently using the Res-TCN architecture with the corresponding input. Then, we added the hallucination network into it and initialized it with the parameters from the pre-trained 2D network. Next, we fine tuned the overall network by introducing hallucination loss and joint classification losses.

Hallucination Loss

To regularize the distance between the representations from the hallucination network and from the 3D network, we added a Euclidean loss between their activations. The loss is defined as follow:

$$L^{(l)}_{hallucination} = \|\sigma(A^{(l)}_{hallucination}) - \sigma(A^{(l)}_{3D})\|^2_2 \tag{5.10}$$

where $A_{hallucination}$ is the hallucination activation and A_{3D} is the 3D activation. σ is the sigmoid activation function. We can apply the hallucination loss after any layer in the network and optimize it directly. To determine which layer should be chosen to compute mid-level regression loss, we extracted outputs from layers in Block C

in the Res-TCN architecture. The highest overall performance was achieved when we added the hallucination loss before the average pooling layer.

Combined Loss Function

To further optimize the network, we applied the cross-entropy loss to the outputs of all the three networks. Combined with the hallucination loss, we trained the model with balanced multiple losses. Precisely, we applied 6 total losses, 5 softmax cross entropy losses using action category labels and one hallucination loss which matches the mid-layer activations from the hallucination branch to those from the 3D branch. The general form of the loss function is given as:

$$L_{overall} = \alpha\, L^{(l)}_{hallucination} + \beta\, L^{2D}_{softmax} + \gamma\, L^{Hal}_{softmax} + \delta L^{3D}_{softmax} + \pi\, L^{Hal2D}_{softmax} + \eta\, L^{2D3D}_{softmax} \tag{5.11}$$

where α, β, γ, δ, π, η are the loss weights with respect to the corresponding loss functions. The determination of the weights is determined empirically after some iterations. We adopted the methodology in [7], where the weight in the hallucination loss depends on the scale of the loss function. When choosing the value we consider the layer from which the hallucination loss is extracted. To avoid that the hallucination loss dominates the overall loss, we heuristically set the weight of the hallucination loss α around 5 times the contribution on the other losses. Since at testing stage, the performance is determined by the joint softmax output from the 2D network and the hallucination network, we set the softmax loss weight $\pi = \eta = 2.0$, then $\beta = \gamma = \delta = 1.0$. The weights can be determined by running a few training iterations in practice.

5.2.4 Experiments and Evaluation

Since the hallucination model requires two kinds of input in different modalities, we evaluated this architecture on the NTU RGB-D dataset as well. We took the 2D skeleton data and 3D skeleton data as the inputs to train the network. Then we used 2D data only at test time.

5.2.4.1 *Base Network*

In our experiments, we chose Res-TCN as the base network architecture for all the three modules. The 2D module and the hallucination module are initialized with 2D weights trained by ourselves. The 3D module was initialized with the pretrained 3D weights for Res-TCN.

5.2.4.2 *Training Parameters*

The overall network is optimized using the Keras deep learning framework [19] with Tensorflow backend [20]. We set the initial learning rate of 0.001. Except for the frozen layers in the 3D module, all layers can be updated with the unified learning rate. We use Stochastic Gradient Descent to optimize the objective function. We use a momentum of 0.9 and a weight decay of 0.0005. The network is trained in 100 iterations.

5.2.4.3 *Evaluation*

Using the pre-trained model of Res-TCN network for initialization, we trained our hallucination network with one euclidean loss and five cross-entropy loss. The hallucination network surpasses performance of Res-TCN with 2D input. Furthermore, in the cross-view setting, the hallucination network even outperforms the network with 3D input. Table 5.2 reports the performance of the hallucination network compared to the Res-TCNs.

	Cross-Subject	Cross-View
Res-TCN with 3D	76.1%	77.1%
Res-TCN with 2D	73.0%	73.5%
Hallucination	75.5%	77.7%

Table 5.2: Comparison of Hallucination network and the base networks on NTU RGB-D dataset in accuracy (%)

5.3 **Long Short Term Recurrent Neural Network**

In the previous sections we saw that Res-TCN with an hallucination model can perform better in RGB data. However, since they both conduct temporal convolution over a fixed-length time interval, real-time inference is not possible. Therefore, we provide another method using LSTM to solve this limitation.

5.3.1 Introduction

Even though Res-TCN provides us a way to explicitly learn readily interpretable spatio-temporal representations for action recognition, it requires a fixed time length skeleton sequence as the input in order to do the temporal convolution. This limitation put a ceiling on the real-time performance of the network. One of the most important evaluation standards in human robot interaction is the reaction time, i.e. the time between the robot being presented with an instruction and the robot initiating a response. Recurrent neural networks (RNNs) offer themselves as good tools to achieve the online recognition task.

RNNs have long been used for modeling temporal sequences. Different from the feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. Just as convolutional networks can readily scale to images with large width and height, and some convolutional networks can process images of variable size, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length. Given a sequence $\mathbf{x}$, initialized with hidden state $\mathbf{h}^{(0)}$, for each time step from $t = 1$ to $t = \tau$, the following update equations can be applied:

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \tag{5.12}$$

$$\mathbf{h}^{(t)} = tanh(\mathbf{a}^{(t)}) \tag{5.13}$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)} \tag{5.14}$$

$$\hat{\mathbf{y}}^{(t)} = softmax(\mathbf{o}^{(t)}), \tag{5.15}$$

where the parameters are the bias vectors **b** and **c** along with the weight matrixes **U**, **V** and **W**, respectively for input-to-hidden, hidden-to-output and hidden-to-hidden connections.

The mathematical challenge of learning long-term dependencies in recurrent networks is that gradients propagated over many stages tend to either vanish or explode. The next section describe approaches to overcoming this problems.

5.3.2 Overview of LSTM

In order to alleviate the gradient vanishing problem, a typical LSTM model is adopted. In this model, a unit is composed of 4 gates. A basic LSTM block is shown in Figure 5.4.

An LSTM block contains an input gate i_t, a forget gate f_t, a cell state c_t, an output gate o_t and an output response h_t. The forget gate decides what information should be thrown away from the cell state using a sigmoid layer. The input gate governs the information flow into the cell state. The output gate controls what parts of the cell state we are going to output as h_t. The cell state ensures that the gradient can pass across many time steps without vanishing or exploding due to its self connected architecture. At time t, the recursive computation of activations of units is

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.16}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.17}$$

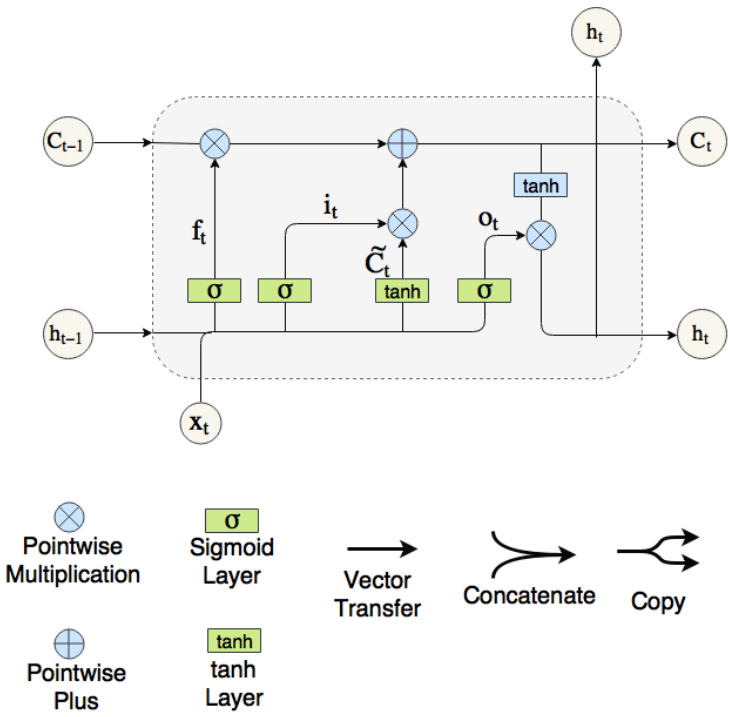

Figure 5.4: The structure of a LSTM memory cell

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{5.18}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t) \tag{5.19}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.20}$$

$$h_t = o_t * tanh(C_t) \tag{5.21}$$

where σ is the sigmoid function defined as $\sigma(x) = 1/(1 + e^{-x})$, W_α is the weight matrix, and b_α is the bias associated with $\alpha \in \{f, i, C, o\}$.

The advantages of LSTMs for modeling sequential data are twofold. First, LSTM models are straightforward to train end-to-end. Second, LSTMs are not confined to fixed length inputs or outputs. They are able to model sequential data of variable lengths.

5.3.3 Model Architecture

In this section, we describe the proposed model for action recognition. Given a sequence of skeleton data, our goal is to generate classification labels at every single time step and continuously update our prediction while sequences are processed.

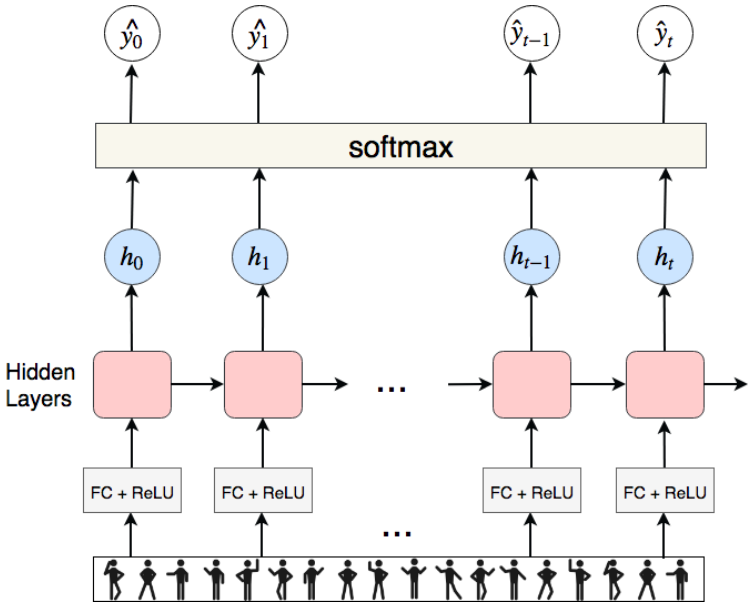

Figure 5.5: The unfolded diagram of the LSTM model

The flowchart of the action recognition model is presented in the Figure 5.5. In this model, the LSTM is trained by 2D skeleton sequences input $\mathbf{x} = [x_1, x_2, \ ... \ , x_T]$ and output labels $y \in [0, \ ... \ , N-1]$, where input duration T varies from sequence to sequence. N is the number of classes. The weights of the hidden layers are recursively updated according to equations 5.16 to 5.21. The softmax layer is used to map the output from the LSTM to a categorical distribution over N different classes:

$$P(Y = i|h_t, W_s, b) = softmax(W_s h_t + b_s), \tag{5.22}$$

The predicted label is obtained by maximum likelihood as

$$\hat{y}_t = argmax_i P(Y = i|h_t, W_s, b), \tag{5.23}$$

Since our goal is to achieve real-time recognition rather than producing label after seeing the whole sequence, at training time every single moment in a skeleton sequence will be labelled with the same as the label in the whole sequence. The network is trained by backpropagation at each frame.

5.3.4 Experiments and Evaluation

To evaluate the online performance of this network necessary to implement the interaction with drones, we use the Helicopter Marshalling Dataset. The overall network was trained from scratch with random initialization. We also tried to add deeper layers in the network, however we found that, they didn't help much. We compare the performance of LSTM with Res-TCN and the hallucination network on the Helicopter dataset. Table 5.3 reports the result of the LSTM network.

Res-TCN	Hallucination	LSTM
74.0%	76.0%	90.5%

Table 5.3: Comparison of LSTM network and the Temporal Convolutional Networks on Helicopter Marshalling dataset in accuracy (%)

On this dataset, LSTM obviously surpasses both the Res-TCN and the hallucination network. However, this network is very sensitive and task-specific. Due to simplicity, this architecture does not perform well on large-scale datasets such as

the NTU RGB-D dataset. To further improve this architecture, we believe deeper layers and more trainable weights are needed.

5.4 Comparison and Discussion

In this section we proposed three different architectures to tackle the action recognition problem. Comparison for Res-TCNs with 2D and 3D inputs convinced us that action recognition can be done without depth. Hallucination network further provide evidence for this assumption. Then to address the real-time demands in the human robot interaction application, we adopted the LSTM model to accomplish a real-time recognition framework.

Chapter 6: Conclusion

In this thesis, a pipeline for 2D skeleton based action recognition was proposed.

First, we adopted OpenPose to estimate human poses in real-time. Then we implemented an evaluated 2D action recognition using as input the human poses. We demonstrated that one of our frameworks is competitive with the 3D based method. In order to prove its robustness and generality, this approach was trained on a large scale action recognition dataset. Later on, we fine tuned the network using task-specific datasets.

Another approach used the LSTM recurrent neural network. This network makes the online recognition task possible. On our specific task, this method outperforms the CNN based framework. It can model temporal sequences very well and is not confined to fixed length inputs or outputs.

We also presented a new helicopter signal dataset with nine categories. This work can be applied for UAV control as well as for helicopter marshalling.

6.1 **Limitations and Future Work**

Currently, we can see the valuable potential of the hallucination network. Comparing the performance between Res-TCNs with 2D and 3D inputs, we notice

that Res-TCN fail to discover depth information adequately. This shortcoming put a ceiling on the performance of the hallucination network. To further overcome this, a better-behaved base architecture for 3D pose estimation need to be proposed. Rather than learning from depth information, optical flow or 3D flow could be included as side information to be learned.

Another issue is the lack of training data for natural gesture recognition. The movements recorded in the helicopter marshalling dataset are is stiff and inflexible. When people interact with robots in everyday life, they will use more natural, smoother movements. Future studies on natural human robot interaction need to be done.

Also the speed of the overall system is restricted by the performance of OpenPose. Even though OpenPose is able to accomplish real-time pose estimation, its speed of hand pose detection is linear in the number of hands in the scene. Since both detection and recognition tasks are based on learning, it is possible to design an end-to-end trainable framework which might improve both accuracy and speed.

Bibliography

[1] C. Lea, M.D. Flynn, R. Vidal, A. Reiter and G. D. Hager. "Temporal convolutional networks for action segmentation and detection". In *The IEEE conference on Computer Vision and Pattern Recognition(CVPR)*, June 2017.

[2] Fermller, C., Wang, F., Yang, Y., Zampogiannis, K., Zhang, Y., Barranco, F., and Pfeiffer, M. (2016). *Prediction of manipulation actions. International Journal of Computer Vision*, 1-17.

[3] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. and Darrell, T., 2015. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of *the IEEE conference on computer vision and pattern recognition* (pp. 2625-2634).

[4] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[5] Kim, T.S. and Reiter, A., 2017. Interpretable 3D Human Action Analysis with Temporal Convolutional Networks. arXiv preprint arXiv:1704.04516.

[6] Herath, S., Harandi, M. and Porikli, F., 2017. Going deeper into action recognition: A survey. Image and Vision Computing, 60, pp.4-21.

[7] Hoffman, J., Gupta, S. and Darrell, T., 2016. Learning with side information through modality hallucination. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 826-834).

[8] Du, Y., Wang, W. and Wang, L., 2015. Hierarchical recurrent neural network for skeleton based action recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1110-1118).

[9] Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L. and Xie, X., 2016, February. Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks. In AAAI (Vol. 2, p. 8).

[10] Li, C., Wang, P., Wang, S., Hou, Y. and Li, W., 2017, July. Skeleton-based action recognition using LSTM and CNN. In Multimedia and Expo Workshops (ICMEW), 2017 IEEE International Conference on (pp. 585-590). IEEE.

[11] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. and Darrell, T., 2015. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2625-2634).

[12] Cao, Z., Simon, T., Wei, S.E. and Sheikh, Y., 2016. Realtime multi-person 2d pose estimation using part affinity fields. arXiv preprint arXiv:1611.08050.

[13] Papandreou, G., Zhu, T., Kanazawa, N., Toshev, A., Tompson, J., Bregler, C. and Murphy, K., 2017. Towards Accurate Multi-person Pose Estimation in the Wild. arXiv preprint arXiv:1701.01779.

[14] Wei, S.E., Ramakrishna, V., Kanade, T. and Sheikh, Y., 2016. Convolutional pose machines. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4724-4732).

[15] Simon, T., Joo, H., Matthews, I. and Sheikh, Y., 2017. Hand Keypoint Detection in Single Images using Multiview Bootstrapping. arXiv preprint arXiv:1704.07809.

[16] Shahroudy, A., Liu, J., Ng, T.T. and Wang, G., 2016. NTU RGB+ D: A large scale dataset for 3D human activity analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1010-1019).

[17] Wan, J., Zhao, Y., Zhou, S., Guyon, I., Escalera, S. and Li, S.Z., 2016. Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 56-64).

[18] Kendon, A., 2004. Gesture: Visible action as utterance. Cambridge University Press.

[19] Chollet, F., 2015. Keras.

[20] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M. and Ghemawat, S., 2016. Tensor-

flow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[21] Shin, M.C., Tsap, L.V. and Goldgof, D.B., 2004. Gesture recognition using Bezier curves for visualization navigation from registered 3-D data. Pattern Recognition, 37(5), pp.1011-1024.

[22] Oka, K., Sato, Y. and Koike, H., 2002. Real-time fingertip tracking and gesture recognition. IEEE Computer graphics and Applications, 22(6), pp.64-71.

[23] Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S. and Kautz, J., 2016. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4207-4215).

[24] Van den Bergh, M. and Van Gool, L., 2011, January. Combining RGB and ToF cameras for real-time 3D hand gesture interaction. In Applications of Computer Vision (WACV), 2011 IEEE Workshop on (pp. 66-72). IEEE.

[25] Kölsch, M. and Turk, M., 2004, May. Robust Hand Detection. In FGR (pp. 614-619).

[26] Ohn-Bar, E. and Trivedi, M., 2013, June. In-vehicle hand activity recognition using integration of regions. In Intelligent Vehicles Symposium (IV), 2013 IEEE (pp. 1034-1039). IEEE.

[27] Mittal, A., Zisserman, A. and Torr, P.H., 2011, September. Hand detection using multiple proposals. In BMVC (pp. 1-11).

[28] Toshev, A. and Szegedy, C., 2014. Deeppose: Human pose estimation via deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1653-1660).

[29] Newell, A., Yang, K. and Deng, J., 2016, October. Stacked hourglass networks for human pose estimation. In European Conference on Computer Vision (pp. 483-499). Springer International Publishing.

[30] Zimmermann, C. and Brox, T., 2017. Learning to Estimate 3D Hand Pose from Single RGB Images. arXiv preprint arXiv:1705.01389.

[31] Rabiner, L.R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), pp.257-286.

[32] Yamato, J., Ohya, J. and Ishii, K., 1992, June. Recognizing human action in time-sequential images using hidden markov model. In Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on (pp. 379-385). IEEE.

[33] Samaria, F. and Young, S., 1994. HMM-based architecture for face identification. Image and vision computing, 12(8), pp.537-543.

[34] Arulampalam, M.S., Maskell, S., Gordon, N. and Clapp, T., 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on signal processing, 50(2), pp.174-188.

[35] Kwok, C., Fox, D. and Meila, M., 2003. Real-time particle filters. In Advances in neural information processing systems (pp. 1081-1088).