

國立臺灣大學電機資訊學院資訊工程學系



碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

基於 3D 殘差網路使用資料填補和骨骼誤差

應用於虛擬實境之手部姿勢估計

Hand Pose Estimation based on 3D Residual Network with
Data Padding and Skeletal Loss for Virtual Reality Applications

丁柏文

Pai-Wen Ting

指導教授：傅立成 博士

Advisor: Li-Chen Fu, Ph.D.

中華民國 107 年 7 月

July, 2018

中文摘要



手部姿勢估計之技術在現今電腦視覺的領域中是一門熱門的研究項目，其目的在於從一張具有手部的影像中去計算出手的節點在空間中的位置並藉此建立手部姿勢。在近年，由於虛擬實境、擴增實境和混合實境等科技的發展逐漸成熟，如何使人在虛擬世界中的感覺能夠更加真實的相關技術也如火如荼地展開，然而對於手部姿勢估計來說仍然有許多困難的地方有待突破。舉例來說，手指與手掌之間互相遮蔽、手勢多樣性的變化等問題，都會造成計算上的困難。

本論文之目的即開發出一套能夠從深度影像中擷取資訊，並且精確的於 3D 座標中計算出手部節點的座標以及手勢之系統，以提供使用者能夠與虛擬實境中的世界自然互動的介面。在本文中，我們提出一個利用大量數據資料來訓練一個深度網路的手部姿勢估計模型。我們首先把手部的深度平面影像轉換成以立方空間表示，並使用資料填充的方式來擴增資料量。接著把處理好的資料拿來訓練卷積神經網路，並加入骨架穩固層來控制手部的物理限制。經由前處理和穩固層的運作，可以有效提升卷積神經網路訓練手部姿勢模型的效能。另外，本系統亦可以在配備有單個圖形處理器之電腦上即時運作。

實驗中，我們將比較在不同的條件下所訓練出來的手部姿勢估計模型之效能，以證明本論文所提出之改進方法能確實在訓練卷積神經網路時提供更好的效果。另外，我們也會在真實世界中實測提出之系統，以證明在極其困難之環境下，仍然可以維持優秀的效果。我們期望所提出之系統可以在虛擬實境或是擴增實境中提供使用者一個更加自然的體驗。

關鍵字: 手部姿勢估計、虛擬實境、卷積神經網路

ABSTRACT



Nowadays, technology of the hand pose estimation is a popular research topic in the area of computer vision. The goal is to estimate the coordinates of the hand joints in the 3D space from an image which contains hand. In recent years, because of the development of virtual reality (VR), augmented reality (AR) and mixed reality (MR), the technology of how to make people feel real in the virtual world has been developed for many years. However, there are still many difficulties in hand pose estimation. For example, the problems of self-occlusion and hand pose variations will cause the difficulties of estimation.

The purpose of this thesis is to develop a system which can extract information from depth image and estimate hand joint coordinates and pose in 3D space accurately, and can provide a natural interface between user and virtual world. In this thesis, we propose a deep network to train a hand pose estimation model by huge data. We first transform the depth image of hand to voxelized grid and use data padding to fill the data. Then, we train the convolutional neural network with preprocessed data and add a skeletal loss layer to control the shape of hand. With the preprocessing and skeletal loss layer, the performance of model can be improved significantly. Moreover, the system can run in real time with a single GPU.

In the experiments, we will compare the performance of models which are trained under different conditions to prove that our proposed method can improve the performance. In addition, we will test the system in real world to show that the system can work well even under the environment which is complex. We expect that the proposed system can provide a more natural way for users to interact with the virtual world.

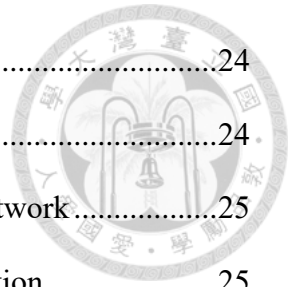
Keywords: hand pose estimation, virtual reality, convolutional neural network

TABLE OF CONTENTS



口試委員會審定書	#
中文摘要	I
ABSTRACT	II
TABLE OF CONTENTS.....	III
LIST OF FIGURES	V
LIST OF TABLES	IX
Chapter 1	1
1.1 Motivation.....	2
1.2 Related Work	5
1.2.1 Hand Pose Estimation	6
1.2.2 3D Convolutional Neural Network	8
1.3 Contribution.....	9
1.4 Thesis Organization	10
Chapter 2 Preliminaries	12
2.1 Convolutional Neural Network.....	12
2.1.1 Basic Concept.....	12
2.1.2 3D Convolutional Neural Network	18
2.2 3D Representation	18
Chapter 3 Hand Pose Estimation with Data Padding.....	21
3.1 Problem Formulation.....	21
3.2 System Overview	22
3.3 Hand Detection	23

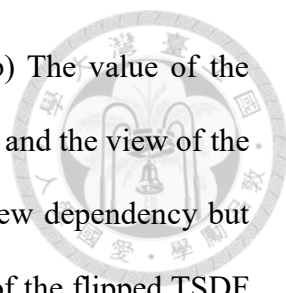
3.3.1	Center-Of-Mass	24
3.3.2	Reference Point Refining	24
3.4	Hand Pose Estimation with 3D Convolutional Neural Network.....	25
3.4.1	2D Depth Image to 3D Voxelized Grid Transformation	25
3.4.2	3D Data Padding	26
3.4.3	Architecture of the Convolutional Neural Network.....	28
3.4.4	Loss Function	30
3.4.5	Data Augmentation.....	34
3.4.6	Implementation Parameter and Detail.....	35
Chapter 4	Experimental Results.....	36
4.1	Settings of Environment	36
4.2	Datasets.....	37
4.2.1	ICVL Hand Posture Dataset	37
4.2.2	NYU Hand Pose Dataset	38
4.2.3	Big Hand 2017 Dataset	39
4.3	Experimental Results	40
4.3.1	ICVL Hand Posture Dataset	42
4.3.2	NYU Hand Pose Dataset	44
4.3.3	Real World Testing	49
Chapter 5	Conclusion	55
	REFERENCE	57



LIST OF FIGURES



- Figure 1-1 The device we have used in our experiment, which is a HMD (HTC VIVE) attached with a depth sensor (Intel RealSense F200). We take the depth image by the depth sensor and reconstruct the hand model in the VR world.3
- Figure 2-1. This figure is a typical CNN structure, which is composed by 3 convolutional layer and 2 fully connected layer. Convolutional layer will extract the features, and the first layer of fully connected layer will interact the features and the last layer of fully connected will output the predicted result.....13
- Figure 2-2. This figure shows the basic concept of the convolutional layer. The orange rectangle is a kernel (filter), and the green boxes are the corresponding features extracted by the kernels. The number of the green boxes is decided by the number of kernel. Assume there are 4 orange rectangle (kernel), then each of them will be convolved with entire input data along its height and width, and finally the 4 green boxes (feature) will be extracted and be stacked together.15
- Figure 2-3. This figure shows the basic concept of max pooling. This max pooling operates a max pooling filter with kernel size 2 and stride 2, which can downsample the input to half of the size.16
- Figure 2-4. This figure shows the function of ReLU in a visual way.....17
- Figure 2-5. This figure shows the basic concept of the 3D convolution. The idea of the 3D convolution is similar with 2D convolution. You can image that all the operations in the 2D convolution be extended to 3D simply. This figure operates a 3D convolution by a filter with size 2 and stride 1.....18
- Figure 2-6. This figure in [27] illustrates different kinds of implementation for TSDF



function. (a) The 3D model of an object or a scene. (b) The value of the projective TSDF is computed with respect to the position and the view of the camera. (c) The value of the accurate TSDF has less view dependency but needed to spend lots of time to calculate. (d) The value of the flipped TSDF is contrary to the accurate TSDF.....20

Figure 3-1. The definition of the 16 joints in this thesis. Each finger has 3 joints on it and 1 joint for the palm.22

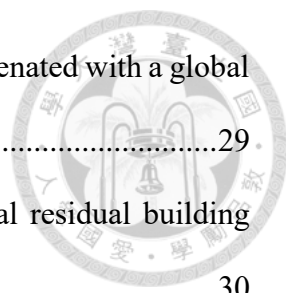
Figure 3-2. The system overview of our proposed work. The 2D depth image will first go through the method of center-of-mass which crops the bounding box of hand roughly. Then we will take the image of the hand as the input and estimate an accurate bounding box by a simple CNN. Next, we transform the 2D image to the 3D data and apply data padding. Finally, we use a 3D CNN which takes 3D data as input and estimate the 3D coordinates of the joints as the final result.23

Figure 3-3. The process of center-of-mass method. We will first eliminate the object which is too near or too far from the camera by applying thresholding. In the next step, we will calculate the center of the mass remained in the image as the reference point and crop the bounding box depend on it.....24

Figure 3-4. The simple and shallow CNN which is used to refine the position of the reference point. This CNN takes the cropped depth image of hand as input and output the offset from the original reference point to the refined reference point.25

Figure 3-5. Illustration of the data padding. (a) shows the 3D data before padding. (b) shows the 3D data after padding.....28

Figure 3-6. Architecture of the proposed 3D convolutional neural network. The network



is composed of eight residual blocks and finally be concatenated with a global pooling and a fully connected layer.29

Figure 3-7. Two kinds of residual function for CNN. Left: a normal residual building block. Right: a “bottleneck” residual building block.....30

Figure 3-8. The illustration of the finger-length-ratio loss function. This figure shows how we calculate the length of each finger. For example, the length of the thumb is estimated as the length of the red line.....32

Figure 3-9. The illustration of the bone-length-ratio loss function. This figure shows how we separate each finger into three parts, which are the three lines in the figure.33

Figure 4-1. Some examples in the ICVL Hand Posture Dataset. This dataset provides many challenging cases.38

Figure 4-2. Some examples in the NYU Hand Pose Dataset. This dataset contains many depth images which are broken and distorted.....39

Figure 4-3. Some examples in the Big Hand 2017 dataset. This dataset contains lots of depth images with different subjects.40

Figure 4-4. The successful estimation fraction of a full hand under the threshold compared with other works on ICVL Hand Posture Dataset.43

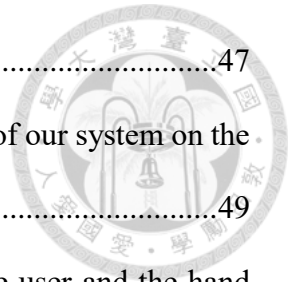
Figure 4-5. The average Euclidean Error of each joint compared with other works on ICVL Hand Posture dataset.43

Figure 4-6. Some examples of the input, ground truth and prediction of our system on the ICVL Hand Posture Dataset44

Figure 4-7. The successful estimation fraction of a full hand under the threshold compared with other works on NYU Hand Pose Dataset47

Figure 4-8. The average Euclidean Error of each joint compared with other works on

NYU Hand Pose dataset.	47
Figure 4-9. Some examples of the input, ground truth and prediction of our system on the NYU Hand Pose Dataset	49
Figure 4-10. Some examples of the real-world testing. (a) shows the user and the hand pose. (b) shows the image after depth threshold. (c) shows the bounding box of the hand and the estimated hand pose.	51
Figure 4-11. Head-Mounted Device. This figure shows the Head-Mounted Device used in the testing, which is HTC VIVE.....	52
Figure 4-12. Depth sensor. This figure shows the depth sensor used in the testing, which is Intel Realsense.	52
Figure 4-13. The user and the scene in the virtual world. The top figure shows the user with the HMD and depth sensor. The bottom figure shows the virtual world and the hand projected in it.....	53
Figure 4-14. Hand model. This figure shows the illustration after attaching a hand model to the joints.	54



LIST OF TABLES

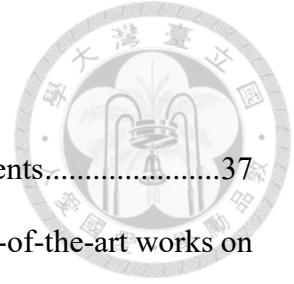


Table 4-1. The equipment specification of our computer for experiments,.....	37
Table 4-2. The quantitative result of proposed method and other state-of-the-art works on ICVL Hand Posture Dataset.	44
Table 4-3. The quantitative result of proposed method and other state-of-the-art works on NYU Hand Pose Dataset.	48
Table 4-4. The quantitative result of proposed method with different weights on Hand- Skeleton loss on NYU Hand Pose Dataset.	48
Table 4-5. The quantitative result of proposed method and other state-of-the-art works on NYU Hand Pose Dataset.	48

Chapter 1



Introduction

Hand pose estimation is a very challenging topic in the area of the computer vision, which aims to find a way to calculate the hand pose and coordinates of the joints in the 3D space. The reason why this technique getting more and more important is that it can provide a natural way for human to interact with the computer or machine without additional devices such like handles or controllers. The developing of the techniques like virtual reality (VR), augmented reality (AR) and mixed reality (MR) is becoming mature now, and the users' experiences of those applications are heavily dependent on how naturally the users can interact with the computer. Thus a system which can provide such support is necessary, and the technique of hand pose estimation undoubtedly plays an important role in it.

In this thesis, we will propose a novel system for hand pose estimation which can crop a hand from a depth image and estimate the coordinates of the joints in the 3D space accurately. Our goal is to integrate the system with the virtual reality devices like Head-Mounted display (HMD), and provide a bridge between users and the virtual world. Thus, we will also reconstruct the hand with our estimated joint coordinates and a hand model. In this chapter, we will introduce the background of this area, formulate the problems we would like to approach and briefly explain the challenges and difficulties we will face in the problems. In addition, we will also show a brief survey of the related work and the organization of the main thesis.

1.1 Motivation

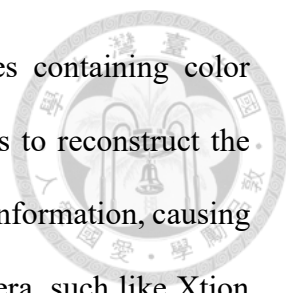


Nowadays, hand pose estimation has become one of the most popular research area in the field of the computer vision and it has also been applied to many applications of the contemporary technology such as virtual reality (VR), augmented reality (AR) and mixed reality (MR). With the invention of the novel technology listed above, Human-Computer Interaction (HCI) has become an attractive topic all over the world gradually. In addition, with the development of virtual and augmented reality, several kind of powerful devices such as HTC VIVE, Samsung Gear VR and PlayStation VR are becoming hot selling in the VR/AR/MR market. However, for the devices mentioned above, the way to interact with the virtual world heavily depends on the handles, which restricts many kinds of operation due to the limitation of number of buttons on the handles. In order to interact with the virtual world with the VR/AR/MR devices, the most natural way is to communicate or control by user's hand. For example, we can attach magnet sensors on user's hand or integrate a depth camera on the front of the Head-Mounting Devices so that we can reconstruct the hand in the virtual world by mapping the estimated joint coordinates on the virtual hand model. However, for the Head-Mounting Devices mentioned above, they do not have equip such accurate hand pose estimation system, which means that the interaction between users and computers provided by the devices can not satisfy the users' demands for better experiences. Consequently, as an efficient interactive way between the users and the computers, the connection between hand pose estimation and virtual reality technology can realize the most natural way to build a bridge for human and computer. Figure 1-1 shows the environment of our system.



Figure 1-1 The device we have used in our experiment, which is a HMD (HTC VIVE) attached with a depth sensor (Intel RealSense F200). We take the depth image by the depth sensor and reconstruct the hand model in the VR world.

As the requirement of the natural way of interaction and the development of the hardware for human-computer interaction, lots of researchers have engaged in the research of hand pose estimation in recent years. Among the methods to approach hand pose estimation including estimation with magnet sensor or other physical way which needs to attach the sensor on our body, the method related to computer vision can be regarded as the most popular and challenging one due to the reason that we do not have to attach anything on our hands. For the past years, in the time when only the RGB



cameras are available, the researchers can only obtain the images containing color information. For the problems of hand pose estimation which needs to reconstruct the model in the 3D space, RGB images do not have any spatial or depth information, causing the difficulties to solve. Fortunately, the invention of the depth camera, such like Xtion PRO, Kinect and Realsense, gave a big chance for the researchers to figure out other methods. The depth images taken from the depth camera contain the information of distance between the object and the camera. The values of pixels of depth images represent the distance from the depth camera to the object, which will be small when close and large when far, normally. With the spatial information given by the depth image, researcher can improve the performance of the method such like hand pose estimation, human pose estimation, or action recognition that related to 3D space. Since then, hand joints estimation in the 3D space has progressed significantly and can perform more accurate results.

However, even there are many researchers have engaged in this topic and paid lots of efforts with the developed high-tech hardware, there are still many difficulties and problems for the task of hand pose estimation. Before we go through the challenging problems of the hand pose estimation, we will first introduce some essences of a hand. Naturally, a human hand normally has five fingers with different lengths, which have their own two or three joints, respectively. For different fingers, the angles which they can bend and the rotations range of each joints are also different, too. Here comes the first challenging problem. A human hand can perform a variety of poses with different shapes, which means that a human hand is far from being a rigid body, and this makes the reconstruction of a hand model extremely difficult. Moreover, we can change our hand pose very quickly in a very short time period, that is, we can transform the pose from opening the hand to clenching the hand very rapidly. Thus, the fact of the high variation

makes the traditional methods of hand tracking which need the result of previous frame or time information difficult to perform well.

On the other hand, occlusion is also a very common issue in the area of computer vision, which causes the problem that we have to estimate the result with the input that has loss information. For the condition that a certain part of an object is occluded by another certain part, we call it self-occlusion. For the task of hand pose estimation, self-occlusion is also a common issue. Due to the high variation of the hand pose, each finger of a hand is easily to block each other from the view of the camera when capturing the image. Because our final goal is to apply the hand pose estimation on the applications of virtual reality (VR) and augmented reality (AR), we tend to attach the depth camera at the position which is near to the user's eye; that means, the camera's view is egocentric (*i.e.*, first person's view). Under the condition mentioned above, it is easy to find that the fingers are occluded by the palm frequently since the palm is often the closest part of a hand from the camera.

Although there are many challenging problems of hand pose estimation to be overcome, the research result can improve the experiences of the applications to the virtual reality. So, we aim to propose a hand pose estimation system which can perform well and contribute to the task of hand pose estimation.

1.2 Related Work

Because of the new developments of the hardware such as depth camera and many kinds of sensors, the technology of the human-computer interaction and virtual reality have been popular in recent years. So, many researchers engage in the research of computer vision and related issues such like hand pose estimation, action recognition and object tracking, aim to improve the performance of their applications. In this section, we

will introduce the researches that are relevant to this thesis. The introduction contains hand pose estimation and 3D convolutional neural network.



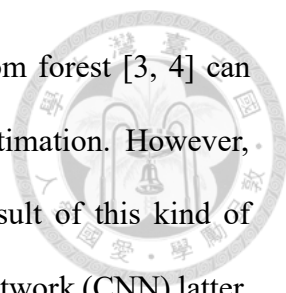
1.2.1 Hand Pose Estimation

Hand pose estimation is an issue which has been researched for many years in the area of computer vision. Until now, there are many researchers have figured out lots of methods and try to solve this problem. However, it seems that there has been no method which can solve it perfectly, so the research of hand pose estimation is still improving for the performance every year.

There have been many methods and algorithms proposed so far. Those methods can be categorized into different aspects. Regard to the input data, there are RGB image and depth image. Regard to the estimation approach, there are generative methods, discriminative methods, and hybrid methods. In this section, we will only introduce the methods which take the depth image as input.

Generative methods will pre-define a hand model and fit it into a depth image of hand by minimizing a hand-crafted cost function. Optimization algorithms such as Particle Swarm Optimization (PSO) [1], Iterative Closest Point (ICP) [2] and their combination are commonly used to optimize the estimation result. These methods' accuracy and performance heavily rely on the hand model and cost function, that is, they have to define user-specific models or other physical constraints under some special considerations. In addition, since these methods usually take temporal information to adjust the results and to better the performance, the error will be accumulated if initialization or previous estimation are inaccurate.

Discriminative methods learn to map the input depth image to direct hand pose joints' coordinates in 3D space or heat maps of each joint which can infer the coordinates from



the large quantity of training data. The methods which apply random forest [3, 4] can provide fast and good performance for the case of hand pose estimation. However, because the hand-crafted features limited the prediction, so the result of this kind of methods had been outperformed easily by the convolutional neural network (CNN) latter. Tompson et al. [5] first propose to apply CNN to predict the 2D heat-maps of each joints of hand which represent the probability of the hand joint's coordinates from a depth image in real time. Inspired by [5], Ge *et al.* [6] transform the depth image to multi-view representation from the view of x , y and z axis and train these three depth maps with three CNNs respectively to extract more useful information and finally fuse them to predict the heat maps of each joints. Ge *et al.* [7] extend their previous work and transform the 2D depth image to a 3D representation form by using Directional Projective Truncated Signed Distance Function (D-TSDF) which can balance the performance and the speed compared with TSDF and Projective TSDF, and utilize 3D convolutional network to estimate the 3D coordinates of hand joints directly. Guo *et al.* [8] propose a region ensemble network to extract the features from hand's different regions of a depth image and fuse them in the final part to estimate an accurate hand pose by a variety of information. Chen *et al.* [9] transform the 3D hand joint coordinates to a spherical part model which uses angles to represent the position of the joint as a loss function to add the physical constraint into the training stage of CNN. Wu *et al.* [10] propose a skeleton-difference layer which allows CNN to learn the shape and the physical constraints of the hand. Oberweger *et al.* [11] train a feedback loop CNN to adjust and improve the estimated result by iteratively correct the error of the hand pose. Oberweger *et al.* [12] design a bottleneck in the fully connected layer of the CNN to enforcing a prior of the 3D hand pose and refine their predicted result by training CNNs which take a small region of each predicted joint to adjust their positions. Oberweger *et al.* [13] improve their previous

work [12] by applying a new network which replaces the original one, doing training data augmentation and training a shallow CNN to refine the position of the reference point which is the center of the hand.

Hybrid approaches combine the discriminative approach and generative approach to estimate the hand pose. The algorithms for hand tracking is more likely to utilize hybrid approaches because they often have the requirement for maintaining the smooth result, which needs the previous estimated result and temporal information to support. Wan *et al.* [14] use two generative networks: a generative adversarial network (GAN) and a variational auto encoder (VAE) for hand modeling and train a discriminator to estimate the posterior of the shared latent space.

1.2.2 3D Convolutional Neural Network

Compared with the typical 2D convolutional neural network, 3D CNN takes the data with 3 channels as input, which often contains the information of depth or time. Because of the additional information attached by the data, 3D CNN has been successfully utilized in the issues of hand tracking, action recognition or dynamic recognition tasks, which need the additional information as the third dimension. Qi *et al.* [15] perform high accuracy on the task of object classification with low resolution input data by 3D CNN. Song and Xiao [16] take RGB-D images as input to do 3D object detection by 3D CNN. Wu *et al.* [17] model the 3D shapes by extracting the 3D features with convolutional deep belief network. Inspired by all these works listed above, we think that 3D CNN can also be applied to estimate the 3D hand pose with the 2D image which contains depth information.

1.3 Contribution



This thesis propose a system for hand pose estimation from cropping the hand in a depth image to estimate the hand pose via convolutional neural network. The major contributions are listed below:

- I. We propose a novel strategy to improve the input data which will ease the learning and estimation of convolutional neural network:

We use depth image as the input of the system, which is better to specify the subject; however, depth image is easily to be broken. For convolutional neural network, the quality of the training data significantly reflects the accuracy. Thus, we design a method which transforms the 2D depth image to 3D voxel and pad the broken part in the 3D space to improve the performance.

- II. We design a new constraint which will steady the estimation result and thus improve the accuracy:

The main architecture of our system is convolutional neural network, and we design a training process for hand pose estimation to improve the performance with additional physical constraints. For the data-driven data, it is able to take physical constraints, which are skeleton and joint angle for hand pose estimation into consideration. Thus, we restrict the ratio of length of fingers and the range of the joint angle for the purpose of steadying the skeleton of the hand.

- III. We design a hand pose estimation system which can be applied to virtual reality:

For the virtual reality (VR) and augmented reality (AR) applications, hand is the most natural way for users to interact with the virtual world, and thus a robust and efficient hand pose estimation system is needed. We design a hand pose estimation system which can run in real-time with one GPU and high

accuracy, and can provide a better experience to the user.



1.4 Thesis Organization

In this chapter, we state the background of hand pose estimation, the challenges of this research and the problems we would like to approach. In addition, we also go through the history of this area and introduce the related work of this research. The remaining chapters of this thesis are listed below:

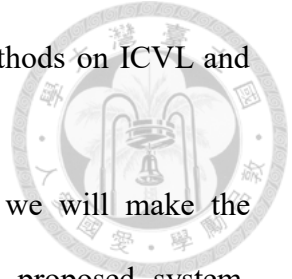
Chapter 2 – Preliminaries. In this chapter, we will introduce the tools and the state-of-the-art methods we apply in our work. We use convolutional neural network (CNN) to build our architecture for hand pose estimation, and we will also explain the basic concepts of it. In addition, we will mention how we preprocess our data including virtual device, which is Head-Mounted display (HMD).

Chapter 3 – Hand pose estimation with data padding. In this chapter, we will divide our system into three parts and explain them respectively. In the first part, we will explain how we extract the bounding box of the hand from a depth image by thresholding and a simple convolutional neural network. For the second part, we will describe the preprocessing method that how to transform the input data from the 2D depth image to 3D voxelized grid, and how to fill the voxel to pad the data. Finally, we will mention the details of our training process including network architecture, parameters of the training and loss function. Moreover, we will also explain how we add the physical constraints in the loss function, which maintain the performance of the hand skeleton.

Chapter 4 – Experimental Results. In this chapter, we will first introduce the two challenging public datasets, the ICVL dataset [18] and NYU dataset [19], which are used to evaluate the performance of our proposed method. Next, we will introduce the Big Hand 2017 dataset [20], which is used to improve our system for VR application. Then,

we will compare the results of our method with state-of-the-art methods on ICVL and NYU datasets.

Chapter 5 – Conclusion and Future Work. In this chapter, we will make the conclusion of our contributions and experimental results of the proposed system. Moreover, we will provide some further issues of the proposed system and the future work of this research.



Chapter 2 Preliminaries



In this chapter, we will briefly introduce the preliminary knowledge which we have applied or used to set up, implement and work on our system of this thesis. In addition, we will also explain the basic concept of the tools and the theory we employ. For the very first part, we will have a brief introduction of a deep learning framework called convolutional neural network (CNN), including 2D and 3D one, which is employed for hand pose estimation part of our system. Second, we will introduce the history and related knowledge of 3D representation of data, which is used to transform the 2D depth image into 3D voxelized grid as the input of CNN.

2.1 Convolutional Neural Network

In this thesis, our system is widely based on convolutional neural network since our main function, which is hand pose estimation, is built on the structure of it. In this section, we will first introduce the basic concept of convolutional neural network fundamental in different research area. For the next part, we will briefly introduce the basic idea of 3D convolutional neural network and the history of usage of it.

2.1.1 Basic Concept

Convolutional neural network (CNN) is a method of machine learning which is composed of a series of neural networks doing the convolution operation as shown in Figure 2-1, so it is also called deep learning method commonly. The perspective of CNN is inspired by the way how the creatures learn things with a virtual conception, and extract the useful features or obtain the knowledge of the patterns. That is, given an image, even RGB or depth, leaning the relation between each pixel's value and find out the connectivity patterns.

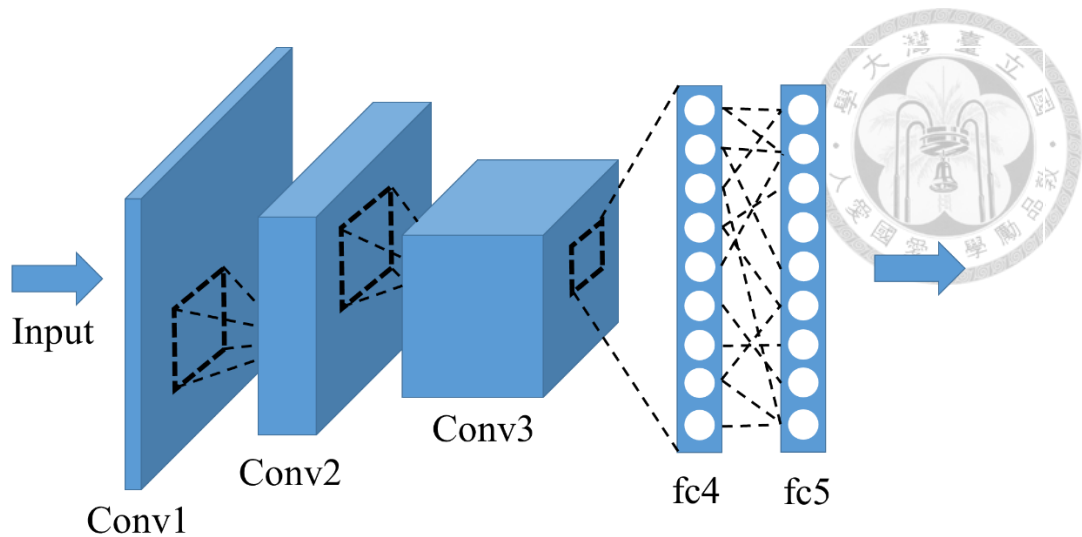


Figure 2-1. This figure is a typical CNN structure, which is composed by 3 convolutional layer and 2 fully connected layer. Convolutional layer will extract the features, and the first layer of fully connected layer will interact the features and the last layer of fully connected will output the predicted result

Thus, with the great ability to extract the features and learn the pattern from an image, CNN has successfully performed many good results for lots of issues in the area of the computer vision, including image classification, image recognition, object tracking, video analysis and action recognition.

Typically, a CNN structure will consists of several layers which operate different calculation to process the parameter from input to output. Except the first layer which receive the input and the last layer output the result, most of the layers in the CNN will receive a feature map from the previous layer, process the feature map by the defined function of the layer with the parameter and finally generate a new feature map which will be passed to the next layer. All these layers will be concatenated in many ways, even in parallel, sequentially or cycle to build the frameworks. Although the CNN structure will be built differently due to different problems, the function implemented in it will still be somewhat similar. For example, most of the CNNs will contain convolutional layer,

pooling layer, activation function layer, fully connected layer, and so on. For the different purpose or usage, depends on the problem or the balance between accuracy and speed, lots of kinds of CNN are introduced by the researchers. For example, Alex Net [21], VGG [22], Google Net [23], ResNet [24] are widely known structure in this area. In the next part, we will introduce some commonly used function of CNN layer:

Convolutional Layer: It is obvious that the convolutional layer must be an important role in this architecture since it has the name “convolutional” neural network. The operation of convolutional layer is to convolute the input feature maps and out them to the next layer. In the process of the convolution, the users will first define the size, weight, total number of kernel and other parameters of kernels which can extract different features depends on the requirement of the users. After the convolution, the value obtained by the operation will be viewed as a neuron. Depends on the parameter to the kernel, a convolutional layer can extract the low-level features such as the edges or the contours of an object, and can even find high-level features such as semantic context of a full image with the processed features. Figure 2-2 shows the basic concept of convolutional layer.

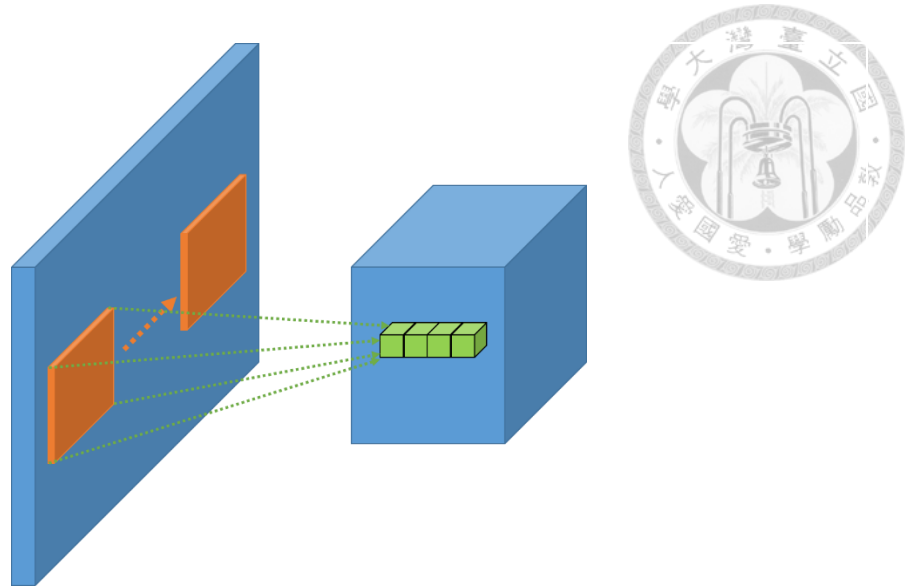
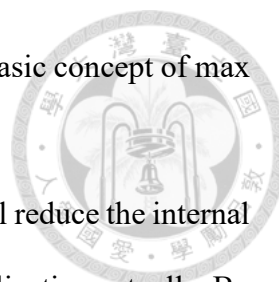


Figure 2-2. This figure shows the basic concept of the convolutional layer. The orange rectangle is a kernel (filter), and the green boxes are the corresponding features extracted by the kernels. The number of the green boxes is decided by the number of kernel. Assume there are 4 orange rectangle (kernel), then each of them will be convolved with entire input data along its height and width, and finally the 4 green boxes (feature) will be extracted and be stacked together.

Pooling Layer: Pooling layer can be viewed as a kind of convolutional layer with specific function which is designed to down-sample the feature maps in most of the CNN architecture design. In this part, we will briefly introduce two kinds of commonly used pool layer, which are Max-pooling layer and Average-pooling layer. For these two kinds of pooling payer, they are similar to the convolutional filtering but with a special rule to assign the value with the kernel. Max-pooling layer will output the largest value in the region (i.e. kernel) by setting the grid which match the largest value to be 1 and others to be 0. In addition, to down-sample the feature maps, the stride is set to be larger than 1 and is set to be the size of the kernel normally. Average-pooling layer will average the value in the region by setting each grid of the kernel to be $1 / (\text{kernel size} * \text{kernel size})$. By applying the pooling layer, the users can reduce the size of a feature map which speeds up the process while pick out the most important information in a region at the same time,



that is, removing the noise in the feature map. Figure 2-3 shows the basic concept of max pooling.

Batch Normalization Layer: The batch normalization layer will reduce the internal covariate shift in the neural network which means doing batch normalization actually. By doing so, the users can set higher learning rate while training the network, and the total time which is need to train the network will also be reduced at the same time.

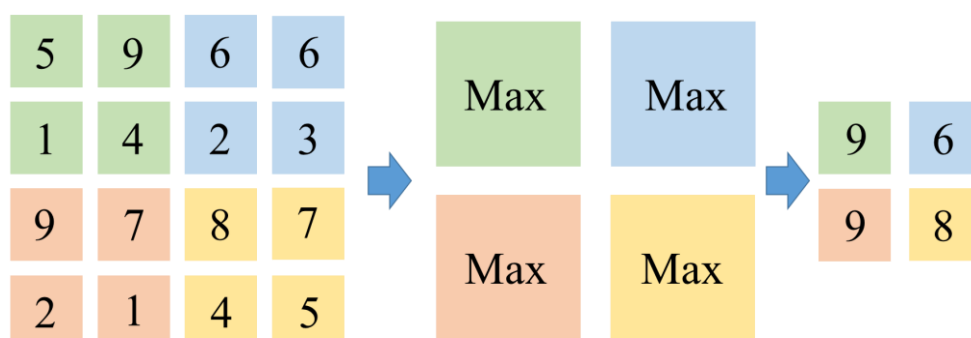


Figure 2-3. This figure shows the basic concept of max pooling. This max pooling operates a max pooling filter with kernel size 2 and stride 2, which can downsample the input to half of the size.

Activation Function Layer: The goal of adding the activation function in the neural network is to add the factor of non-linear. There are many kinds of activation function which have been applied, and one of the most commonly used activation functions is Rectified Linear Unit (ReLU). The function of the ReLU can be derived as bellow:

$$f(x) = \max(x, 0) \tag{2-1}$$

ReLU function will set the negative values as 0 and only activate the non-negative values on a feature map, which achieve the goal to add the factor of non-linear. Figure 2-4 shows the ReLU function.

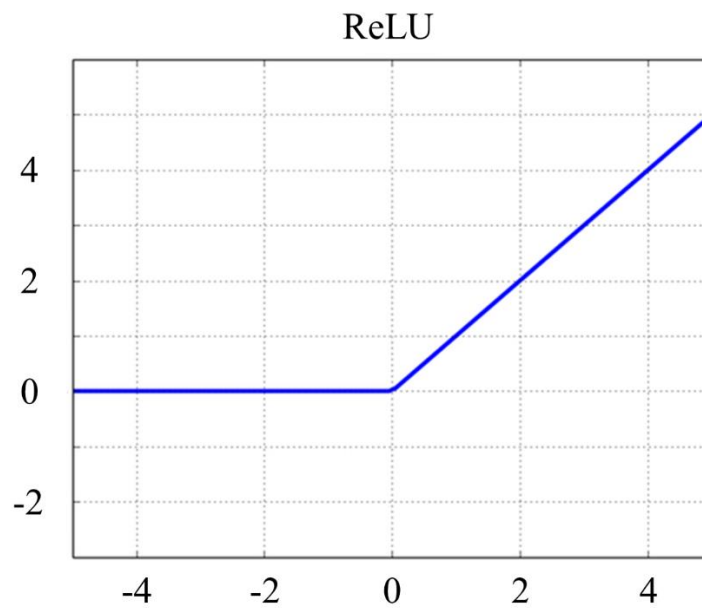
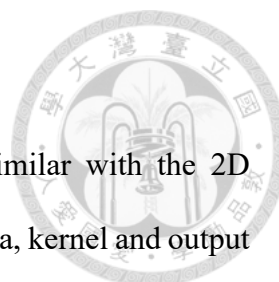


Figure 2-4. This figure shows the function of ReLU in a visual way.

Fully Connected Layer: The fully connected layer of the convolutional neural network will connect every neuron of an input feature map to every neuron of the output feature map, just like a normal neural network do. Thus, the relation between each neuron will be taken into consideration. Because of the property of the fully connected layer mentioned above, CNN will lose the spatial information when applying it, so the users often place the fully connected layer at the last part of the CNN.

Loss function Layer: The loss function will define the different between the ground truth and the predicted result, which is used to evaluate the penalty to train the CNN model. While in the training stage of the CNN, the users will evaluate the loss and update the parameters according to the loss value by back propagation. The definition of the loss function can be modulated depend on the problem or the effect that the users would like to implement. Nowadays, the commonly used loss function are Euclidean loss, soft max and cross entropy.



2.1.2 3D Convolutional Neural Network

The basic idea of the 3D convolutional neural network is similar with the 2D convolutional neural network. The different point is that, the input data, kernel and output is increased to 3 dimension, while the fully connected layer still remains the same. 3D CNN has been successfully applied to many kinds of issues in the area of computer vision such like 3D object recognition or 2D image with time information. Moreover, because the input data of the 3D CNN must be the type of 3 dimension, so the users must apply an appropriate method to transform the data to 3D one if the original data is 2D. Figure2-5 shows the basic concept of 3D convolution.

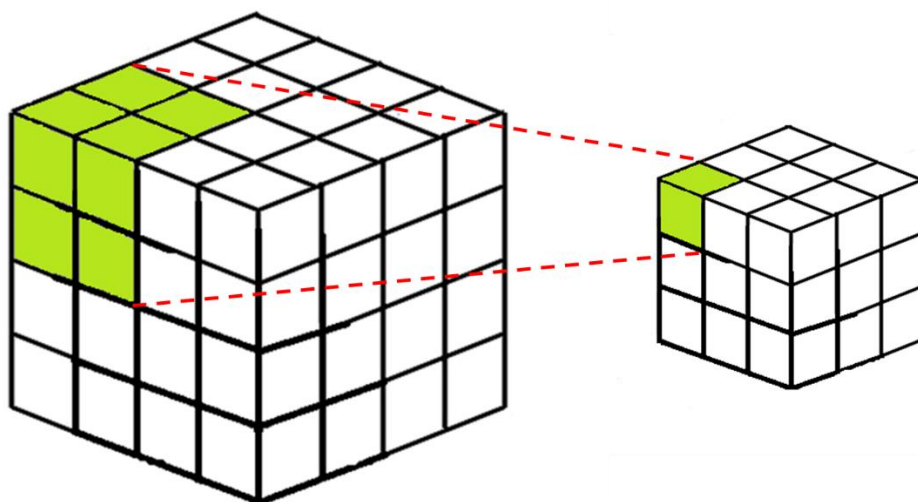
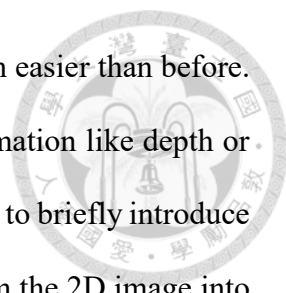


Figure 2-5. This figure shows the basic concept of the 3D convolution. The idea of the 3D convolution is similar with 2D convolution. You can image that all the operations in the 2D convolution be extended to 3D simply. This figure operates a 3D convolution by a filter with size 2 and stride 1.

2.2 3D Representation

With the development of the modern sensor such like depth camera and infrared ray



sensor, the data which contains 3D information can be obtained much easier than before. So even with the 2D image, if the pixel value contains the 3D information like depth or distance, it also can be transformed to 3D data, too. Here we are going to briefly introduce some 3D data representations and some methods which can transform the 2D image into 3D one. First, we will introduce Truncated Signed Distance Function (TSDF) which is applied in KinectFusion [25]. Now we assume that there is a 3D object in a 3D space which is stored in voxel representation, and the coordinates of the occupied voxels by the object are known. First, for the values of the occupied voxels will be set as 0, and for the other voxels which are not occupied by the objects, we will set their values as the minimum distance to the object, that is, for those voxels which are nearer to the object, their value will be smaller. In addition, depends on the position of the camera, the values of the voxels are positive if they are in front of the object, which means that the voxels can be seen from the view of the camera, and for those voxels which can't be seen from the camera view (i.e. occluded by the object) will be set negative. Finally, the values will be set a threshold and normalized, so the final values will be in the range $[-1, 1]$, and for those voxels whose distance to the object is longer than 1 will be set as 1 or -1, depends on the position. Here, for the name of Truncated Signed Distance Function, Truncated means the threshold for the maximum value for the voxels, Signed means the voxels are in front of the object or not, Distance means the distances between the voxels and the object. Then, we will next introduce projective TSDF. The idea of projective TSDF is similar with the TSDF. The only different part between this two function is that the distance from the voxel to the object is only found on the line or sight from the camera view. Finally, because the computing of the accurate TSDF is very time consuming while projective TSDF will loss too much information, so a compromise method which called projective Directional TSDF (D-TSDF) is proposed in [16] which use a 3D vector

representing three directions' distances which replace Euclidean distance. Figure 2-6 which is refer in [26] illustrates different kinds of implementation for TSDF function.

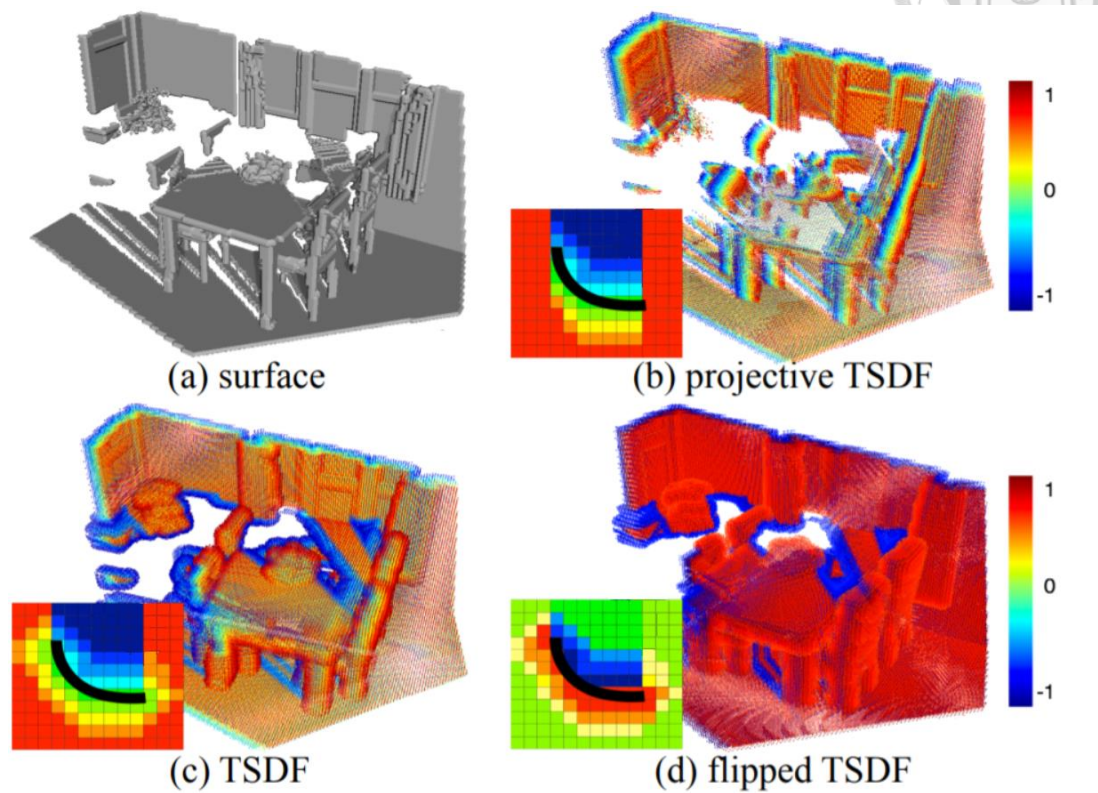


Figure 2-6. This figure in [27] illustrates different kinds of implementation for TSDF function. (a) The 3D model of an object or a scene. (b) The value of the projective TSDF is computed with respect to the position and the view of the camera. (c) The value of the accurate TSDF has less view dependency but needed to spend lots of time to calculate. (d) The value of the flipped TSDF is contrary to the accurate TSDF.

Chapter 3 Hand Pose Estimation with Data Padding



In this thesis, we will propose an accurate hand pose estimation system for the applications of virtual reality (VR), augmented reality (AR), mixed reality (MR) and human-computer interaction (HCI). Our final goal is to combine the proposed system and the head-mounted display (HMD) to provide a natural and accurate interaction under a real world condition.

In this chapter, we will describe the details of our proposed hand pose estimation system. This chapter contains three sections. In the first section, we will go through the proposed system briefly and simply describe the input and output of each part. For the second section, we will describe in detail how we extract the hand from a depth image. For the third section, we will detail the system of hand pose estimation, including input data preprocessing, network architecture and training process details.

3.1 Problem Formulation

The main problem we would like to solve in this thesis is given a 2D depth image which only contains hand, we want to estimate the 3D coordinates of each joint and palm of the hand. For 2D depth image, the input size will be $96 * 96$ pixels with one channel, and the values of the pixels will range from 0 to 255. As a result, the output should be a vector with size $3 * \text{total-joint-number}$, representing x , y and z coordinate of joints of the hand. In this thesis, we define our joint number as 16 as shown in Figure 3-1. There are several difficult challenges to achieve the goal. First of all is the high variations of the poses. This is because for a human hand, each finger has two or three joints on it, and

each joint can do the rotation, respectively. The second challenge is severe self-occlusions. Because the fingers may occlude each other under many conditions, and this will lead to the difficulty of estimating the 3D positions of joints since the information is blinded. The final one is broken image. Although we can obtain high quality depth image with the development of the depth sensor, there are still broken parts occurring in the image, and this also make the estimation be hard.

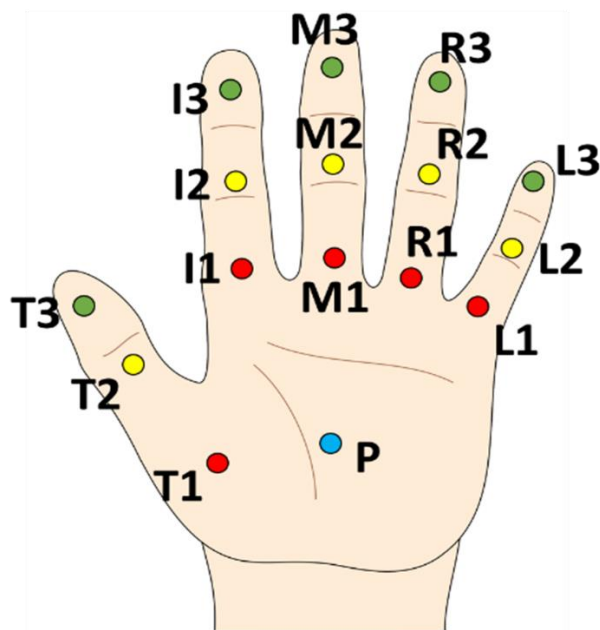


Figure 3-1. The definition of the 16 joints in this thesis. Each finger has 3 joints on it and 1 joint for the palm.

3.2 System Overview

Figure 3-2 shows our system overview. At the first step of the system, we will obtain the depth image and then crop the hand by estimating the bounding box with center-of-mass method. To bound the hand more tightly and accurately, we apply a simple and shallow CNN introduced in [13] to refine the center of the hand. After cropping the hand from the depth image with tight bounding box, we will do data preprocessing including

transformation from 2D depth image to 3D voxelized grid, data padding and data augmentation, which will be detailed in later section. Finally, we take 3D voxelized grid as the input of the 3D convolutional neural network and output the 3D coordinates of the joints of the hand.

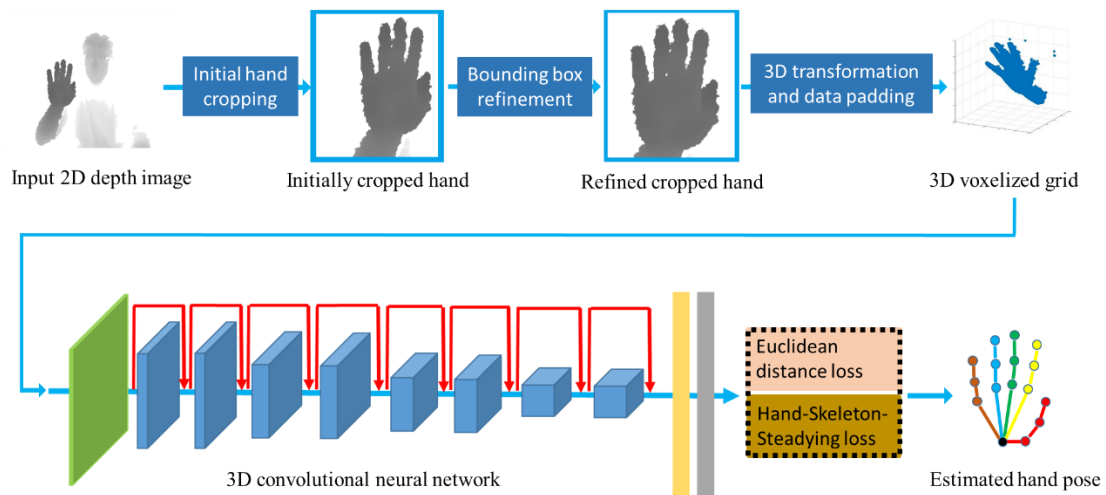
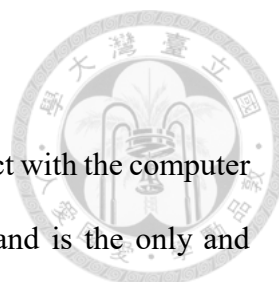


Figure 3-2. The system overview of our proposed work. The 2D depth image will first go through the method of center-of-mass which crops the bounding box of hand roughly. Then we will take the image of the hand as the input and estimate an accurate bounding box by a simple CNN. Next, we transform the 2D image to the 3D data and apply data padding. Finally, we use a 3D CNN which takes 3D data as input and estimate the 3D coordinates of the joints as the final result.

3.3 Hand Detection

To get the 3D voxelized grid of the hand, which is the input of the CNN, cropping the bounding box of the hand tightly from the depth image captured by the depth camera first is necessary. The quality of the cropping image will significantly affect the difficulty of learning and performance of CNN, so we will describe the details about how we obtain a better bounding box as follows.



3.3.1 Center-Of-Mass

The premise of our proposed system is to provide a way to interact with the computer as an application of virtual reality, so we assume that the user's hand is the only and closest object from the depth camera. Thus, we apply a simple depth threshold to the depth image to exclude other object like human body which are too far or too close to the camera. After thresholding, we choose the reference point which is the center of the hand by center-of-mass method, that is, we average the x and y coordinates of the remaining pixels which are not excluded to be the center as shown in Figure 3-3.

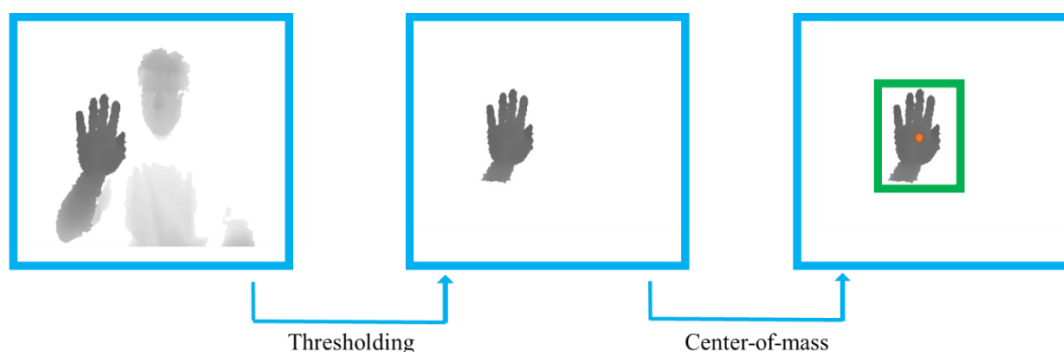


Figure 3-3. The process of center-of-mass method. We will first eliminate the object which is too near or too far from the camera by applying thresholding. In the next step, we will calculate the center of the mass remained in the image as the reference point and crop the bounding box depend on it.

3.3.2 Reference Point Refining

Although we have estimated the reference point by the center-of-mass method, there remains some errors due to the noise. The pixel values of depth image taken by the depth camera will represent the distance from camera to the object in most of the time. However, even with the modern technology, the quality of the depth camera can't perform perfectly, which may cause random pixels on the depth image reflecting error values or even broken areas and will influence the accuracy of the reference point. Hence, we apply the method

as [27] to design a simple CNN to refine the coordinate of the reference point calculation. The CNN takes the depth image cropped by the method as mentioned above as input and output an offset from the original reference point to the ground truth point which is a pair of x and y values. The structure is illustrated in Figure 3-4. Finally, we can obtain the accurate reference point by adding the offset to the original reference point and re-crop the bounding box of the hand.

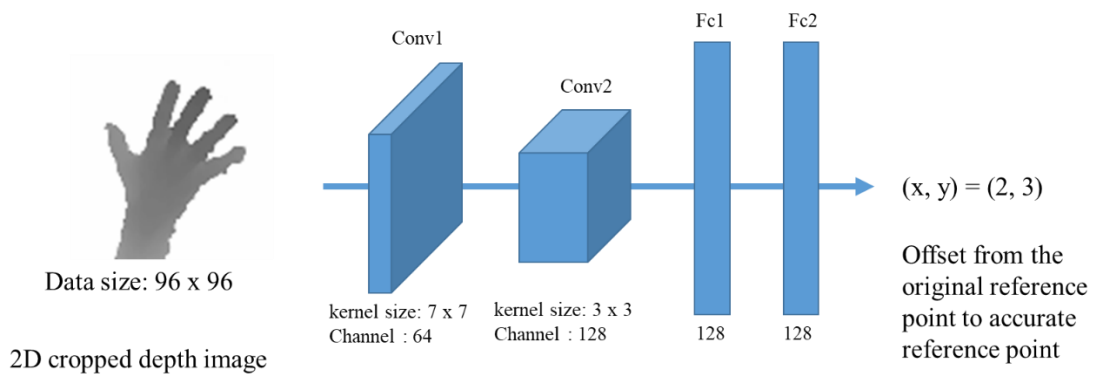


Figure 3-4. The simple and shallow CNN which is used to refine the position of the reference point. This CNN takes the cropped depth image of hand as input and output the offset from the original reference point to the refined reference point.

3.4 Hand Pose Estimation with 3D Convolutional Neural Network

In this section, we will describe the details of input data preprocessing including transformation of 2D depth image to 3D voxelized grid and data padding, 3D convolutional neural network architecture, loss function design, training parameters and implementation details.

3.4.1 2D Depth Image to 3D Voxelized Grid Transformation

To fit the input size of our proposed system, we should first convert the 2D depth

image into 3D voxelized grid representation. First of all, we resize our 2D depth image to the size of 96 * 96 pixels and normalize the values of pixels which are the distances from the depth camera to every point on the hand from 0 to 95 so that we can project the hand surface to 3D voxelized grid. After depth normalization, we create a cubic with size of 96 * 96 * 96 and project the 2D depth image into it. For example, here we denote the 2D depth image as D_2 and 3D voxelized grid as D_3 . Then, the values of voxels of D_3 will be listed as follows:

$$D_3[x][y][z] = \begin{cases} 1, & \text{if } D_2[x][y] = z \\ 0, & \text{else} \end{cases} \quad (3-1)$$

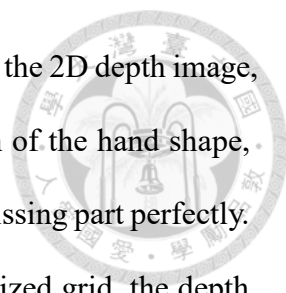
That is, voxel value will be set as 1 if the voxel is occupied by a depth point and be set as 0 if is not.

3.4.2 3D Data Padding

Nowadays, as mentioned above, because of the rise of the high quality, commercial depth camera, we can capture the depth image with accurate depth values. However, there are still many broken parts that occur in the depth image, especially on the margin of the object. This problem will cause a very large place of missing values when converting the 2D depth image into 3D voxelized grid, which leads to the difficulty for CNN to learn parameter and to do estimation. Hence, to overcome this problem, we propose a method to pad the voxels which are around the occupied values with a cube. We denote the original voxelized grid as D_3 and the padded voxelized grid as D_3^s . And the padding function is showed as follows:

$$D_3^s[x][y][z] = \begin{cases} 1, & \text{if exist } D_3[x+i][y+i][z+i] = 1 \\ 0, & \text{else} \end{cases} \quad (3-2)$$

where i ranges from -2 to 2, which is used to pad the shape of cube. By padding the data, we can improve the performance of the results, and the advantages of doing so are listed below:

- 
- I. When there is a broken part or missing value which occur in the 2D depth image, it is very difficult to be recovered because of the variation of the hand shape, that is, the method such as interpolation can't recover the missing part perfectly. However, if we convert the 2D depth image into 3D voxelized grid, the depth points are represented in a spatial way, and thus if we pad the voxels surrounding the occupied voxels, it doesn't distort the shape of the hand excessively while padding the missing part at the same time.
 - II. For the 3D voxelized grid which is projected from the 2D depth image, the maximum number of the occupied voxels is limited by the total pixel number of 2D depth image, which is $96 * 96$ pixels in our proposed system. Less information contained in the 3D voxelized grid may lead to difficulty for CNN to learn and influence the final result. But with the padded data, CNN can learn easily, converge faster and perform better with more information.
 - III. As mentioned above, the occupied voxels in the 3D data are very rare and only represent the surface of the hand. By padding the surrounding pixels with a shape of cube, the augmented 3D data may look more like a hand instead of a piece of paper as shown in Figure 3-5.

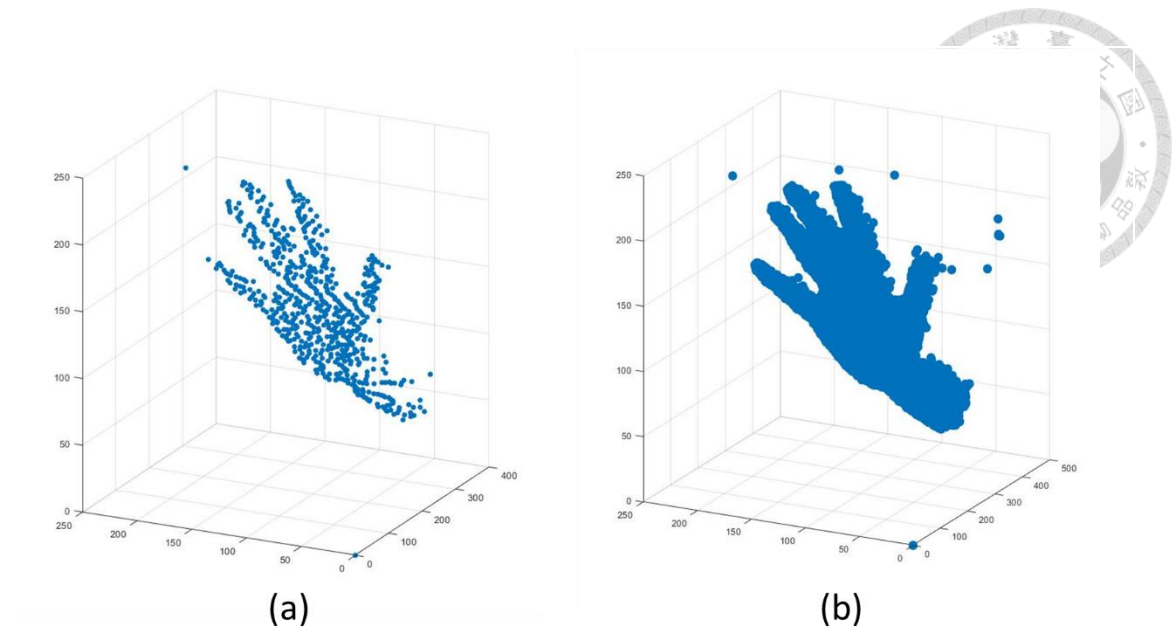


Figure 3-5. Illustration of the data padding. (a) shows the 3D data before padding. (b) shows the 3D data after padding.

3.4.3 Architecture of the Convolutional Neural Network

The proposed 3D convolutional neural network takes 3D voxelized grid with size $96 * 96 * 96$ as input and output a vector with size $3 * J$ elements, which stand for the x, y and z coordinates of the entire J joints of a hand. Our method architecture which is called HSSNet is proposed as shown in Figure 3-6 which is based on the previous work [24]. The 3D voxelized grid will first be input into a convolutional layer, whose kernel size is $3 * 3 * 3$ with stride 2, and then the input data will be downsampled to half of the original size. Then, we concatenate the convolutional layer with an activation function (*i.e.*, ReLU) and a max pooling layer with kernel size 2. After the first convolutional block, the extracted feature map will be input into 4 residual blocks as shown in Figure 3-7. The kernel size of residual block is $3 * 3 * 3$ with stride 1. Finally, we concatenate the extracted feature map from the 3D residual convolutional layer with a global pooling, and finally we take the last fully connected layer as the size of our output, which is $3 * J$.

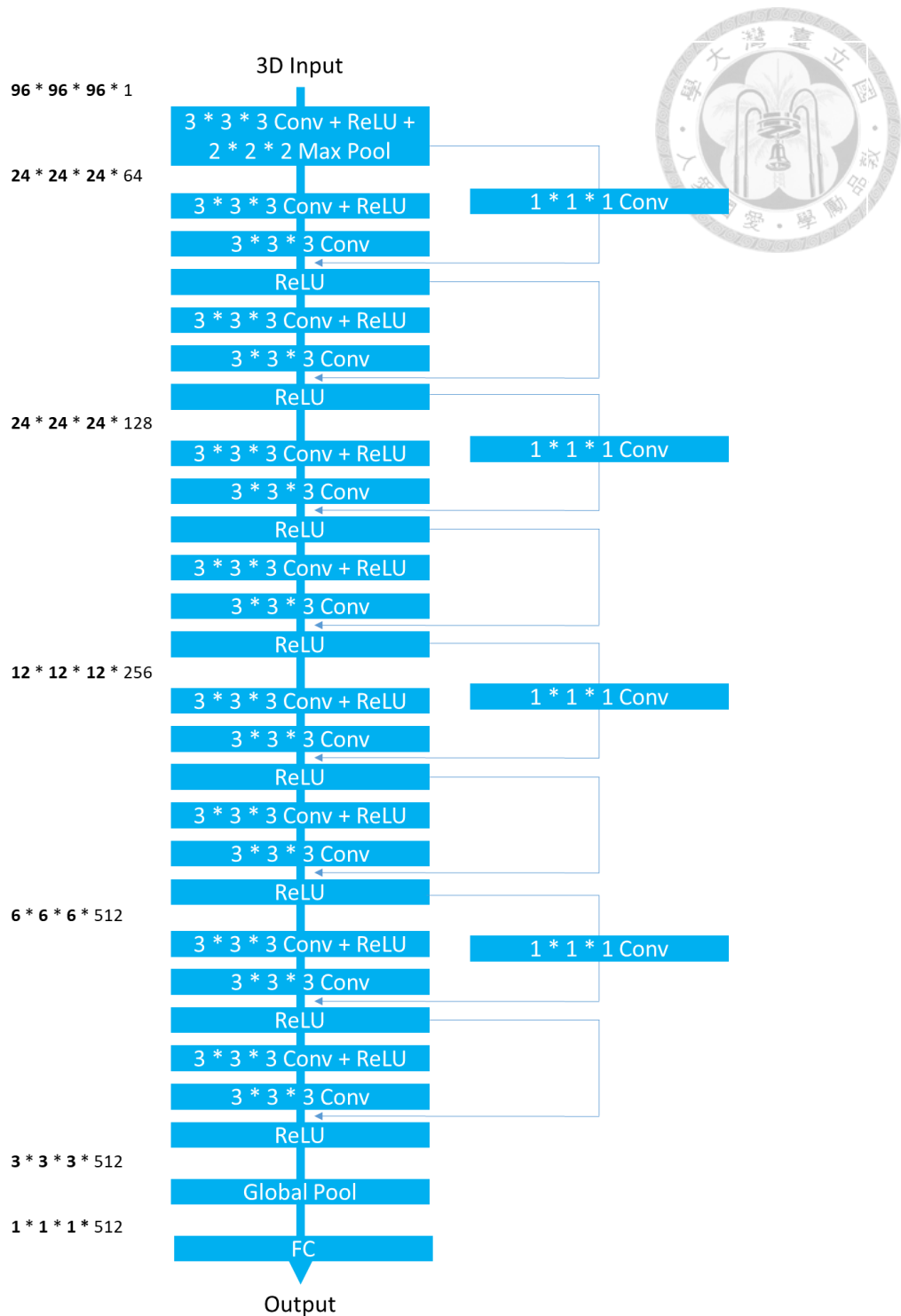


Figure 3-6. Architecture of the proposed 3D convolutional neural network. The network is composed of eight residual blocks and finally be concatenated with a global pooling and a fully connected layer.

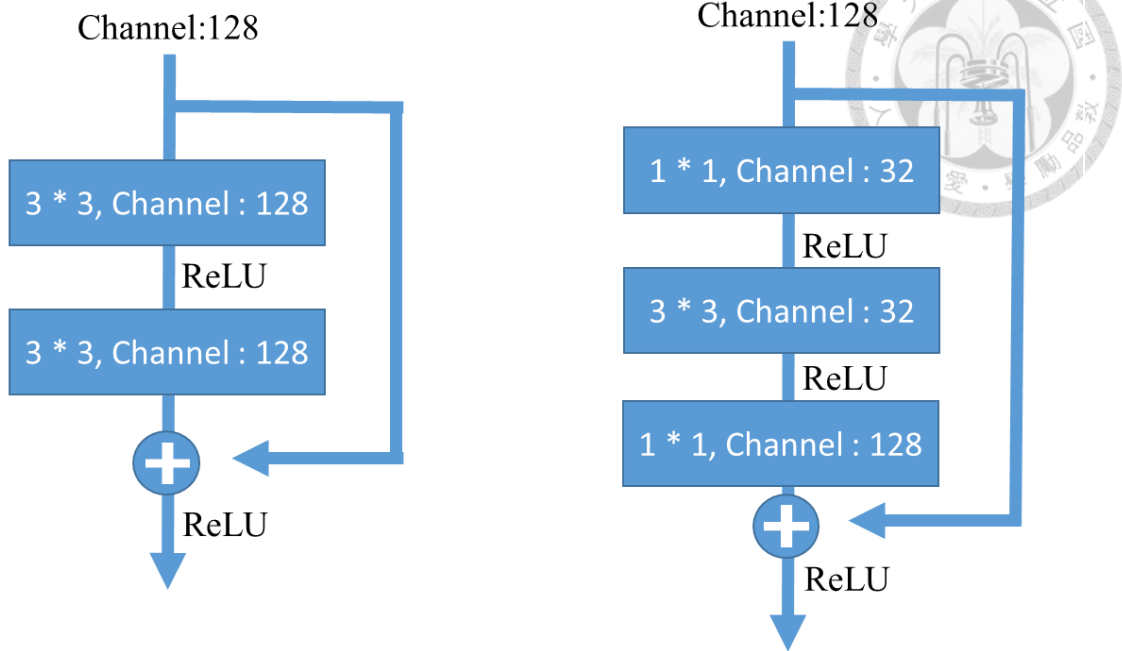


Figure 3-7. Two kinds of residual function for CNN. Left: a normal residual building block. Right: a “bottleneck” residual building block.

3.4.4 Loss Function

In this section, we will detail the design of our loss function to train our convolutional neural network. At the end of the proposed CNN, which is the last layer, the output will be generated as a $3 * J$ vector, which represents a vector of 3D coordinate, *i.e.*, x , y and z of the entire J joints of the hand. Once we have the predicted joint coordinates, we can evaluate the difference between the predicted one and the ground truth. Here, we use the Euclidean distance as the first loss function which is defined as below:

$$L_e(H_j) = \sum_{j=1}^J \|H_j(j_x, j_y, j_z) - H_j^*(j_x, j_y, j_z)\|^2 \quad (3-3)$$

where H_j and H_j^* are the predicted result and the ground truth of the j th joint of the hand, where j_x , j_y and j_z , is the x , y and z coordinate of j th joint. And, J is the total

number of joints of the hand. The Euclidean loss will directly regress the joint position coordinates, and it is also a basic loss function for training of CNN model. So, we apply this Euclidean loss as our main loss function to let CNN learn the spatial information between input depth image and output estimated result. However, for the case of hand pose estimation, Euclidean loss only takes the depth image information which is the depth values from the camera to the hand into consideration and doesn't consider any physical constraints of the hand. Therefore, in order to add the physical constraints in the training stage, we design some novel loss functions to control the skeleton of the hand.

Generally, for most of the public datasets, material models or applications which is about hand and hand pose, the reference points are often at the joints of the hand. Therefore, the link connecting a joint and another joint can be viewed as a bone, and all the joints and the bones together will form the structure of the hand. So, this structure will be like a tree with the palm as the root, and all the edges in such tree refer to bones. In our proposed Hand-Skeleton loss function, we will evaluate the loss, concerning the ratio of each finger out of all fingers in the structure. The Hand-Skeleton loss function combines two different terms, which are finger-length-ratio loss and bone-length-ratio loss, and we will explain them, respectively.

For the finger-length-ratio loss, the goal of this function is to steady the ratio of the length between each finger. By applying this loss function, we hope that the length of each finger of the predicted result can be in a reasonable range, that is, none of the fingers will be too long or too short compared with other fingers. So the loss function is defined as follows:

$$L_f = \sum_{i=1}^5 \|P_i - P_i^*\| \quad (3-4)$$

where

$$P_i = \frac{PL_i}{PL_{total}} \quad (3-5)$$

$$PL_{total} = \sum_{i=1}^5 PL_i \quad (3-6)$$



Here, i points to the i th finger of total five fingers in our function, and P_i and P_i^* are the predicted ratio and the ground truth ratio of each finger to the total length of five fingers, where PL_i means the length of one finger, which is calculated by the output 3D coordinates. Figure 3-8 illustrates the ideas behind these.

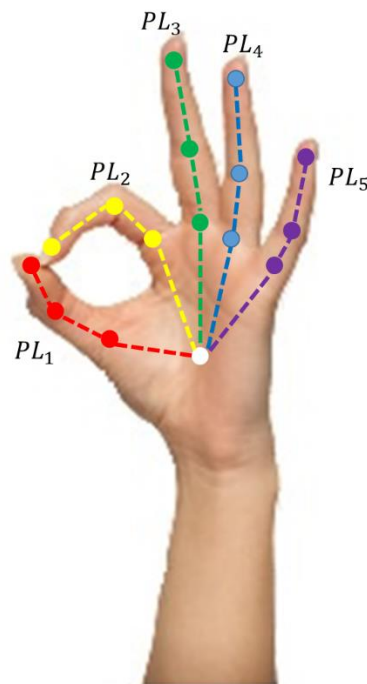


Figure 3-8. The illustration of the finger-length-ratio loss function. This figure shows how we calculate the length of each finger. For example, the length of the thumb is estimated as the length of the red line.

For bone-length-ratio loss, the goal of this function is to steady the length of each bone of a single finger. What we actually want to do is to control the ratio over three virtual bone of each finger defined by the joints. By doing so, the lengths of bones of each



finger will also be predicted in a reasonable range so that the predicted hand looks more real in the 3D space. The loss function is designed as follows:

$$L_b = \sum_{i=1}^5 \sum_{k=1}^3 \|B_{i,k} - B_{i,k}^*\| \quad (3-7)$$

where

$$B_{i,k} = \left(\frac{\|BL_{i,k}\|}{\|BL_{i,1}\| + \|BL_{i,2}\| + \|BL_{i,3}\|} \right) \quad (3-8)$$

Here, i points to the i th finger and the total number of fingers is set 5 in our function, and k is total number of the bones of a single finger which is set 3. Note that $B_{i,k}$ and $B_{i,k}^*$ are the predicted and the ground truth ratio of the length of one bone to the total length of three bones of the i th finger. $BL_{i,k}$ is the length of one bone of one finger. Figure 3-9 illustrates the ideas behind these.

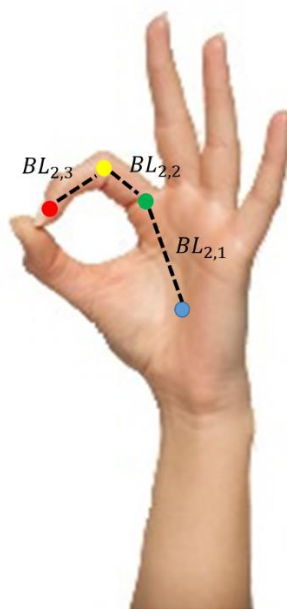


Figure 3-9. The illustration of the bone-length-ratio loss function. This figure shows how we separate each finger into three parts, which are the three lines in the figure.

Finally, the total loss for the training procedure of the convolutional neural network

will be evaluated as follows:

$$L_t = L_e + \omega_{fb}(L_f + L_b) \quad (3-9)$$

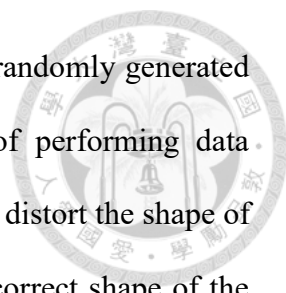
where L_e , L_f and L_b are Euclidean loss, finger-length-ratio loss, and bone-length-ratio loss, respectively. Moreover, ω_e , ω_{fb} are the weights of the three loss functions. Here, we will adjust the weight, which is ω_{fb} , to fit the value to get the best performance.

3.4.5 Data Augmentation

For the training part of the convolutional neural network, the total quantity of the data is very important and significantly impact the final estimation result, especially for the case of hand pose estimation, which has large variations in hand shapes, sizes, poses and orientations. In order to prevent the condition of overfitting and improve the performance of the proposed 3D CNN model with different kinds of hand, we decide to adopt 3D data augmentation, including translation, stretching and rotation on the training data. For the 3D data in the 3D space, we can do additional operation such as rotation with respect to the x- or y-axis which can't be performed in 2D depth image. We will first translate the occupied voxels along x-, y- and z-axis of the camera coordinate system by three translation factors t_x , t_y , and t_z . Then, we will stretch the occupied voxels along the x-, y- and z-axis by stretching factors s_x , s_y and s_z . Finally, we will rotate the occupied voxels around x, y and z axis with rotation factors r_x , r_y and r_z . For example, for an occupied voxel p , after the translation, stretching and rotation, the result will be p^t as follows:

$$p^t = R \cdot S \cdot (p + T) \quad (3-10)$$

where T is a 3×1 matrix which contains three value t_x , t_y and t_z , S is a 3×3 diagonal matrix whose values from left-top to right-bottom are s_x , s_y and s_z , R are three 3×3 rotation matrices which related to x-, y- and z-axis. In our proposed system,

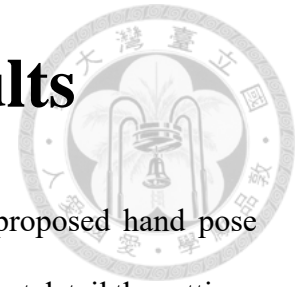


the translation, stretching, and rotation of the data augmentation is randomly generated and perform on the input data. In addition, although the goal of performing data augmentation is increasing the quantity of the data, we don't hope to distort the shape of the hand extremely which will affect the training part to learn the correct shape of the hand, so we don't distort the data in a very large range. The translation of x-, y- and z-axis is in the range of $[-5, 5]$ pixels. The stretching factors of x, y and z axis is in the range of $[0.8, 1.2]$. The rotation of the x, y and z is in the range of $[-15^\circ, 15^\circ]$. For the training time, we train our 3D convolutional neural network with both original training data and distorted training data at the same time.

3.4.6 Implementation Parameter and Detail

The input size of our proposed network is $96 * 96 * 96$ and output size is $3 * J$ which is x, y and z coordinates of J joints of hand where J is set 16 in our thesis. We train our proposed 3D neural network end-to-end with the dropout ratio 0.5 and learning rate $1 * 10^{-4}$ and batch size 5. Finally, we implement our system with python and tensorflow and train on a single GPU which is NVIDIA GTX 1080ti.

Chapter 4 Experimental Results



In this chapter, we will conduct several experiments of our proposed hand pose estimation system on public datasets under different conditions. We first detail the settings of the environment we use, and then introduce the two datasets, of which are used to evaluate our system and another dataset which is used to train the CNN to improve the performance in the real world. Next, we show the experimental results on two datasets and compare the performance with that of other state-of-the-art works. We will also evaluate the performance of the hand pose estimation with and without our proposed methods to validate the improvement of them. Finally, we will give some discussions and show some results in real-world testing.

4.1 Settings of Environment

We use a personal computer which is equipped with an Intel i5-8400 central processing unit (CPU) and 16 GB random access memory (RAM) to conduct various experiments of our proposed hand pose estimation system. The graphic processing unit (GPU) we attach onto our computer is NVIDIA GeForce GTX 1080ti. The total system is built on Ubuntu 16.04, 64-bit operating system. Table 4-1 shows the details of the equipment specification of this computer.

We apply Google tensorflow [27] as our deep-learning tool box to implement our convolutional neural network (CNN) architecture with Python as programming language. Google tensorflow [27] provides an interface to build CNN and supports parameter-updating mechanisms. In addition, it also provides many implemented layers which we have introduced in Sec. 2, including the 3D convolutional layer, and thus we can use these directly to build our work easily.

Table 4-1. The equipment specification of our computer for experiments

Equipment	Specification
Central Processing Unit (CPU)	Intel Core i5-8400 @ 2.80GHz * 6
Random Access Memory (RAM)	16.0 BG
Graphic Processing Unit (GPU)	NVIDIA GeForce GTX 1080ti
Operating System (OS)	Ubuntu 16.04
System Bit Type	64 bit

4.2 Datasets

In this section, we will detail two public datasets which are used to evaluate our system in our experiments and another dataset which is used to improve the performance for real-world testing. The three public datasets that have been used here are ICVL Hand Posture Dataset [18], NYU Hand Pose Dataset [19] and Big Hand 2017 dataset [20]. The first two challenging datasets are widely used to evaluate the performance of the hand pose estimation. So, we choose these two datasets to evaluate our proposed hand pose estimation system and compare our results with those of the other state-of-the-art works.

4.2.1 ICVL Hand Posture Dataset

ICVL Hand Posture Dataset is a public and challenging dataset for hand pose estimation which is released by Imperial Computer Vision & Learning Lab (ICVL) and is used in our experiments. The depth images of hands are captured by Intel's Creative Interactive Gesture Camera [28] from the third person's view. The total number of the depth images are about 24,000 and are divided into one group with about 22,000 depth images for the training set and another with the remaining 2000 depth images, which are two respective consecutive sequences, for the testing set. In addition, because the depth images of the training set are augmented by rotation, which rotate the images from -180

degrees to 180 degrees with the interval of 22.5 degrees, so there are about 330,000 depth images in total actually. For the annotation in the dataset, there are totally 16 joints annotated on the hand, which are 3 joints for each finger and 1 joint for the palm. Each joint is annotated with the 3D coordinates in the image space, which are u , v and d . For the depth images capture by the present depth camera, broken or missing are often occur. However, in this dataset, most of the depth images are high quality and include many difficult hand pose with serious occlusion. With the high quality and the challenging cases in the dataset, ICVL dataset is used in most of the paper related to hand pose estimation and thus we also use it to evaluate our system. Figure 4-1 shows some examples of the dataset.



Figure 4-1. Some examples in the ICVL Hand Posture Dataset. This dataset provides many challenging cases.

4.2.2 NYU Hand Pose Dataset

NYU Hand Pose Dataset is a public and challenging dataset for hand pose estimation which is released by New York University. They use Microsoft Kinect [29] to capture the depth images of hands from the three different viewpoints of third person's view with both RGB images and depth images simultaneously. The total number of the depth images is about 80,000, which are 72757 training data and 8252 testing data respectively. In the training data, there is only one subject while in the testing data, there are two in total. For

the annotation in the dataset, there are total 36 joints with the 3D coordinates for each joint. Although NYU hand pose dataset provides a large quantity images for hand pose estimation, which is a significant contribution for the training for deep learning model, compared with ICVL hand posture dataset, the quality of the depth images are not very high. For most of the depth images in the dataset, there are many broken or missing part occur especially at the margin of the hand, and the contour of the hand shape is not sharp enough. In addition, there also are some noises surrounding the hand. However, even the quality of the depth images is not good enough, it becomes the challenge condition for the research which would like to solve the problem with the broken and noise of the depth image. NYU hand pose dataset is also used in many related work, so we also choose this dataset to evaluate the performance and compare with other works. Figure 4-2 shows some examples of the dataset.

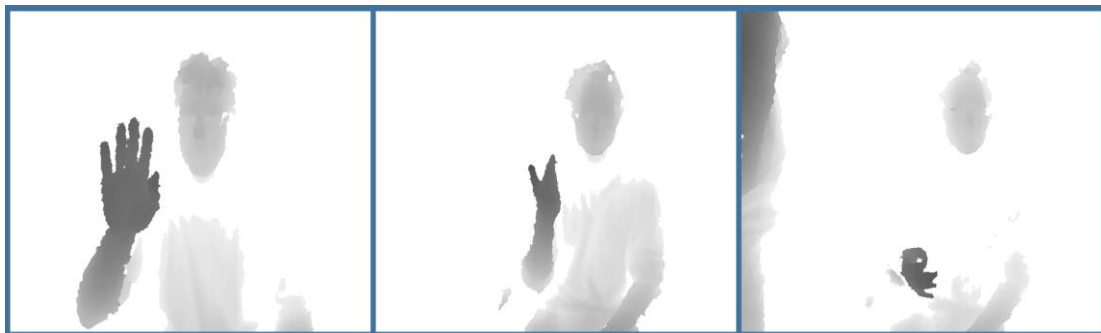


Figure 4-2. Some examples in the NYU Hand Pose Dataset. This dataset contains many depth images which are broken and distorted.

4.2.3 Big Hand 2017 Dataset

Big Hand 2017 dataset is a large-scale and challenging hand pose estimate which is collected by a novel capture method. The total number of the depth image is about 957K for training and 295K for testing which include third person's view and first person's view. For the annotation in the dataset, there are total 21 joints with the 3D coordinates

for each joint. Because this dataset is collected by the tracking system with the 6D magnetic sensors, which can inverse kinematics and automatically obtain the position of the joints, so the annotation of position of each joint in the 3D space is much more accurate than other datasets. In addition, because the annotation doesn't need to be labeled by human frame by frame, so the collection process can be very fast and thus produce lots of data in a short time. Although we don't evaluate our system performance on this dataset, we have trained our system with this dataset to improve the performance of the application of virtual reality.



Figure 4-3. Some examples in the Big Hand 2017 dataset. This dataset contains lots of depth images with different subjects.

4.3 Experimental Results

In this section, we will show the experimental results of our proposed hand pose estimation system on two public datasets, which are ICVL Hand Posture Dataset and NYU Hand Pose Dataset. In addition, we will also compare the results with the other related state-of-the-art works [3, 10, 13, 30, 31] and make some discussions of different aspect. In the last part, we will compare the performance of the baseline and the improved system under different conditions and make some discussions of them.

There are two kinds of performance evaluation methods which are commonly used for the task of hand pose estimation to test the accuracy of estimated positions of hand joints. The evaluation methods are introduced as below:

- I. **Average Euclidean distance error:** The evaluation of this method is to compute the Euclidean distance between the position of the estimated joints and the position of the ground truth joints which are defined as the same joint type of the hand in the 3D space. We will finally average the error of each joint from all the frames in the testing data of each dataset to see the overall results. The error is calculated as below:

$$Distance\ Error = \frac{\sum_{j=1}^J \|H_j - H_j^*\|}{J} \quad (4-1)$$

where J is the total number of joints. H_j is the predicted joint positions and H_j^* is the ground truth joint positions.

- II. **The fraction of success under threshold:** This evaluation is to compute the fraction of the successfully estimated hand pose frame under different threshold. A successfully estimated case means that the Euclidean distance error of every joint of a hand pose is under the threshold. The fraction is calculated as below:

$$Fraction = \frac{\sum_{n=1}^N \alpha(\max_j (\|H_j - H_j^*\|) \leq T)}{N} \quad (4-2)$$

where N is the total number of testing frames. T is the threshold. $\alpha(\text{condition})$ is an indicate function which equals to 1 if condition is true and 0 if condition is false.

For the first evaluation method, it reflects the average error of each joint and thus can represent the performance of a hand pose estimation system in a simple and easily understandable way. For the second evaluation method, it is a very strict criterion because the predicted hand pose must be perfect, and all the estimated joints' Euclidean distance error have to be smaller than the threshold, or it will be considered as a fail case. This evaluation will be plot through different thresholds and become a curve. So more robust

the hand pose estimation system is, the higher the curve is. In this thesis, we will apply there two evaluation methods in our experiments.



4.3.1 ICVL Hand Posture Dataset

In this dataset, we will compare the result of our proposed system with other state-of-the-art methods. We show the evaluation of average Euclidean distance error of each joint and the mean error on the bar chart in Figure 4-5. The result shows that our proposed method can outperform other state-of-art methods in most of the joints, which can prove that our overall prediction is more accurate in this dataset. We also show the robust ability of our system estimation in Figure 4-4, which shows the comparison of the successful fraction under different threshold evaluation, and this evaluation shows that our methods have a significant improvement compared with other works, especially when the threshold of the error is very small (*i.e.*, from 5 to 15). Moreover, this fact can also prove the powerful effect of our data preprocessing and Hand-Skeleton loss layer which add the additional physical constraints to the training of our CNN model. In the table 4-2 which shows the result of the mean Euclidean distance error. As the results show, our method can achieve 6.67mm mean distance error on testing set, which outperforms the performance of other state-of-the-art works about 1.4mm. Although our performance is slightly better than [31] with 0.12mm, [31] has to do the iterations many times to refine the result, while our work is able to predict the hand pose precisely only going through the 3D network for one time. This fact indicates that the robustness of our proposed network. Finally, Figure. 4-6 shows some examples of the results of our proposed method on the ICVL Hand Posture Dataset, including inputs, ground truths and predicted results. We can also find that our predicted results are very similar to the ground truth, which indicates the robustness of our hand pose estimation system.

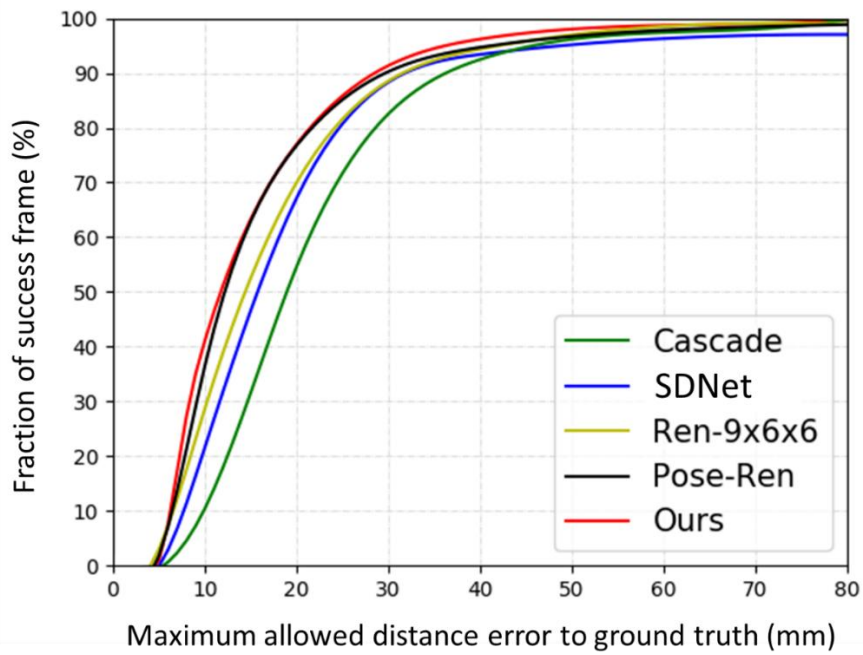
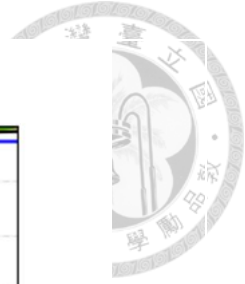


Figure 4-4. The successful estimation fraction of a full hand under the threshold compared with other works on ICVL Hand Posture Dataset.

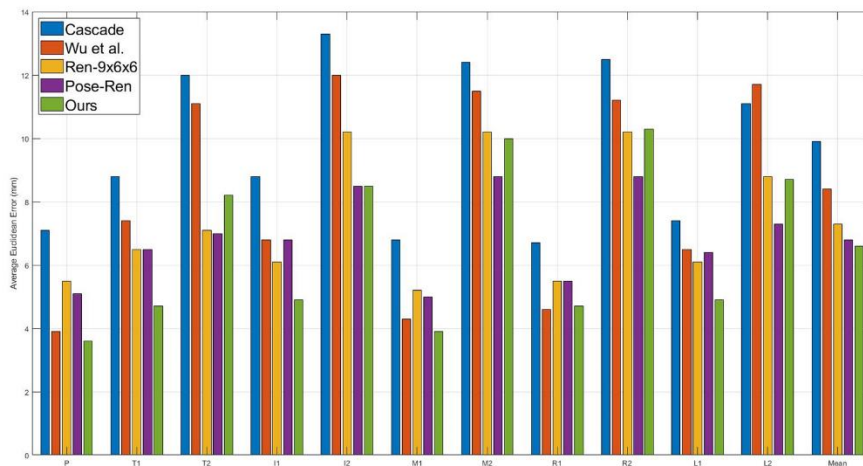


Figure 4-5. The average Euclidean Error of each joint compared with other works on ICVL Hand Posture dataset.

Table 4-2. The quantitative result of proposed method and other state-of-the-art works on ICVL Hand Posture Dataset.

Method	Average Euclidean Distance Error (mm)
Cascade	9.9
Wu et al.(SDNet)	8.45
REN-9x6x6	7.31
Pose-REN	6.79
Ours(HSSNet)	6.67

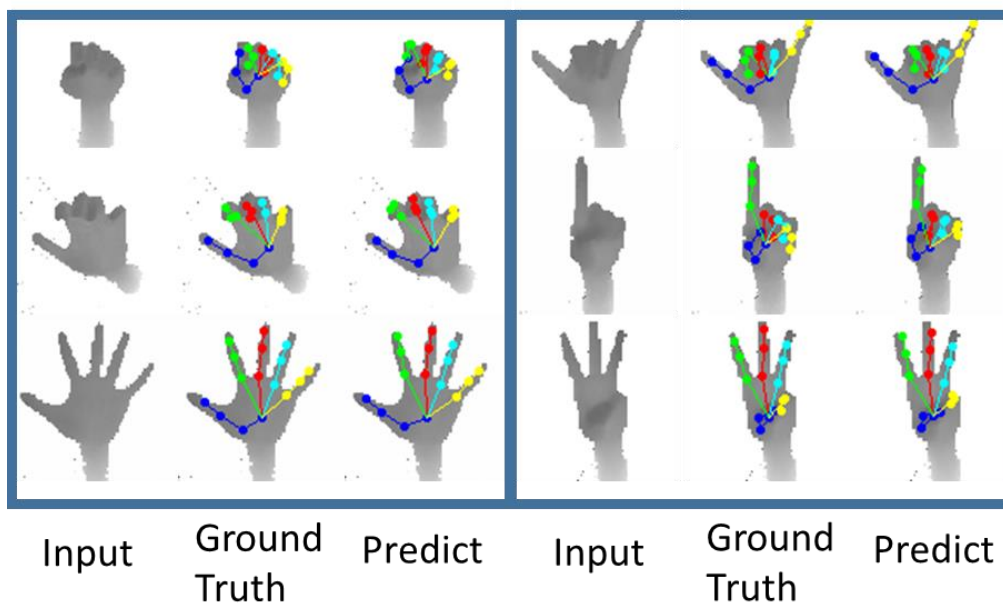
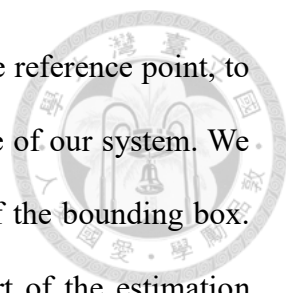


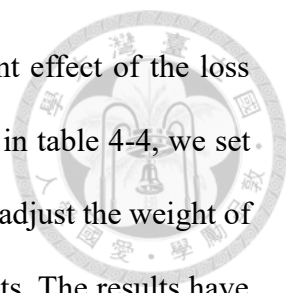
Figure 4-6. Some examples of the input, ground truth and prediction of our system on the ICVL Hand Posture Dataset

4.3.2 NYU Hand Pose Dataset

In this dataset, we will also compare the result of our method with other state-of-the-art methods as well as the ICVL Hand Posture Dataset. In addition, we will also compare our result with our baseline, which is the model trained without additional data preprocessing and additional knowledge, and prove the improvement of the performance. Moreover, we will show the result of our hand detector, which is composed of depth



threshold and a simple shallow CNN which refine the position of the reference point, to show the performance of the cropping hand from a raw depth image of our system. We use Intersection of Union (IoU) as the evaluation of the accuracy of the bounding box. Literally, IoU will compute the intersection part and the union part of the estimation bounding box and the ground truth bounding box. If the IoU value is larger than 0.5, then we consider it as a success case, and we have achieved the precision as 0.97. In addition, because we don't like to distort the image when we resize the cropped image from the bounding box, so we will estimated the bounding box as a square. By doing so, when we resize the image to the size which fits the input of CNN, the shape of the hand will not be distorted too extremely. We also show the experimental results on NYU Hand Pose Dataset compared with other state-of-the-art works in Figure 4-7, Figure 4-8 and Table 4-3. As the Figures and the table show, the performance of our method can outperform other works. Our proposed hand pose estimation achieve 11.66mm mean Euclidean distance error, while the mean error of other state-of-the-art works is 12.24mm. That is, our performance is better than other works about 0.58mm. For the fraction of success under threshold, we also outperform other state-of-the-art works, especially at the threshold from 20 to 40mm. As the results show, we can easily notice that the performance of NYU Hand Post Dataset is worse than the performance of ICVL Hand Posture Dataset. This is caused by the low quality of the broken depth image, even we propose the method which can recover the missing part of the depth image, it is still very difficult to perform as well as the ICVL Hand Posture Dataset which has better image quality. However, with the proposed improvement, we can still perform better results compared with other works. Figure. 4-9 shows some examples of the results of our proposed method on the NYU Hand Post Dataset, including inputs, ground truths and predicted results. In the table 4-4, we have compared the experimental result of the effect of the Hand-Skeleton loss function



in a quantitative way, and this table also demonstrates the significant effect of the loss function with the physical constraints in the training part. As shown in table 4-4, we set the weight of the Euclidean error loss function to be 1, and gradually adjust the weight of the Hand-Skeleton loss function to be different value to see the results. The results have shown that when the weight of the Hand-Skeleton loss function increases, the performance becomes better. However, when the weight becomes larger than a threshold, the performance will starts to become worse. This fact indicates that although the Hand-Skeleton loss function can help to maintain the skeleton of the hand, it is only an additional knowledge, so it can't be used individually or dominate the training process. But overall this function can indeed improve the performance in the training part. Because the final goal of our proposed system is to apply the hand pose estimation to the applications of the virtual reality, so making the skeleton of the hand as stable as possible is very important. Sometimes, a little bit error of the position of a single joint may be tolerable. However, if the skeleton of the hand is distorted, it will be very strange and troublesome and effect the user experience dramatically, especially when we attach the estimation result on a hand model. In table 4-5, we have compared the experimental result of the effect of the data padding in a quantitative way. As the results show, when we pad the data with the voxels from the size $3 \times 3 \times 3$ to $5 \times 5 \times 5$, the performance is improved. This fact indicates that the padding exactly recover the missing part of the broken image. However, when the size of the padding voxel comes to $7 \times 7 \times 7$, the performance will starts to become worse. And this fact indicates that if we pad the data with inappropriate size, the padding voxels will blur the data and decrease the performance. Figure 4-9 shows some examples of NYU dataset.

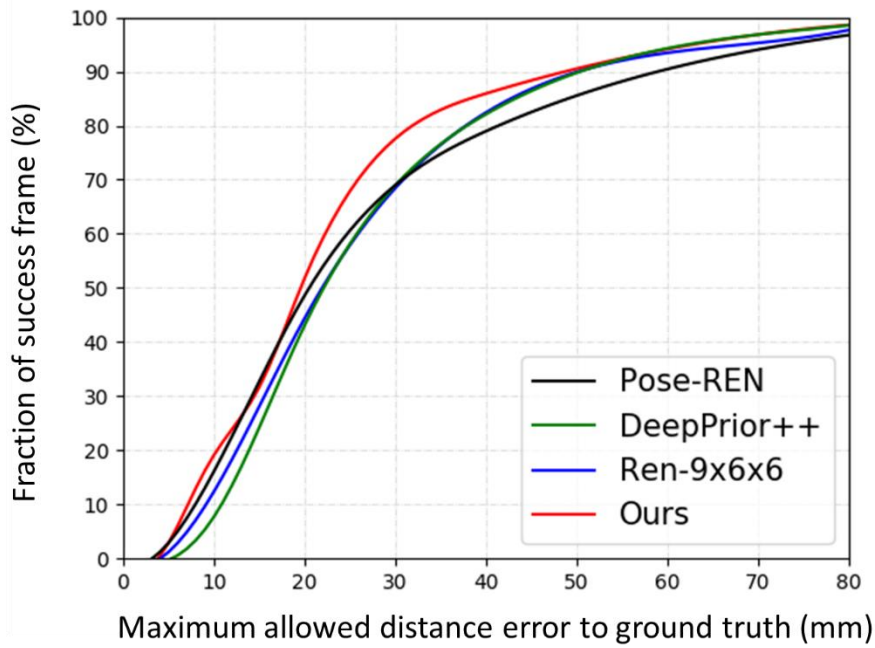


Figure 4-7. The successful estimation fraction of a full hand under the threshold compared with other works on NYU Hand Pose Dataset

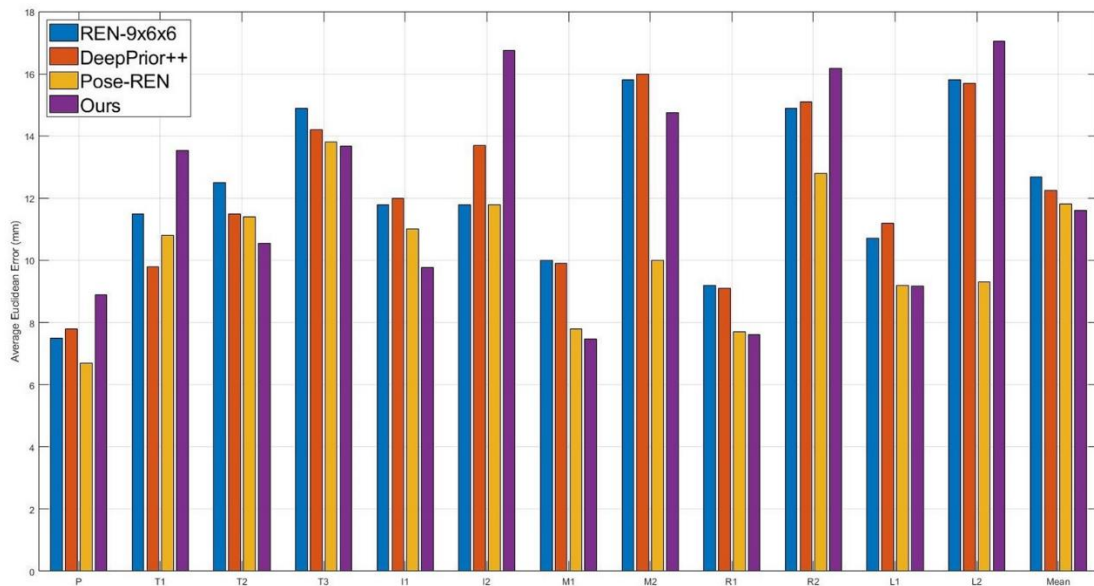


Figure 4-8. The average Euclidean Error of each joint compared with other works on NYU Hand Pose dataset.



Table 4-3. The quantitative result of proposed method and other state-of-the-art works on NYU Hand Pose Dataset.

Method	Average Euclidean Distance Error (mm)
REN-9x6x6	12.69
DeepPrior++	12.24
Pose-REN	11.81
Ours(HSSNet)	11.66

Table 4-4. The quantitative result of proposed method with different weights on Hand-Skeleton loss on NYU Hand Pose Dataset.

Network Settings (weight)	Average Euclidean Distance Error (mm)
Baseline (0.0)	13.22
HSSNet (0.3)	12.06
HSSNet (0.4)	11.95
HSSNet (0.5)	11.66

Table 4-5. The quantitative result of proposed method and other state-of-the-art works on NYU Hand Pose Dataset.

Network Design(voxel)	Average Euclidean Distance Error (mm)
Baseline (without padding)	11.87
HSSNet (3x3x3)	11.79
HSSNet (5x5x5)	11.66
HSSNet (7x7x7)	11.91

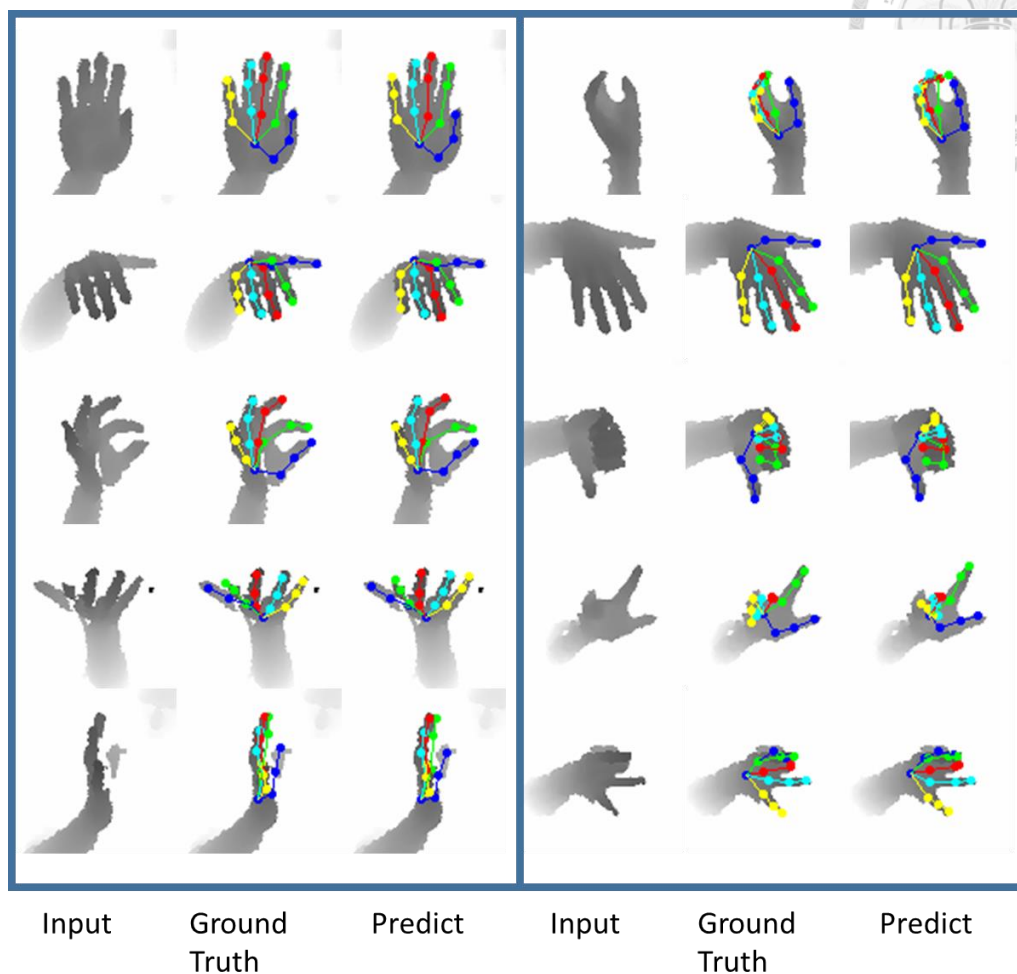
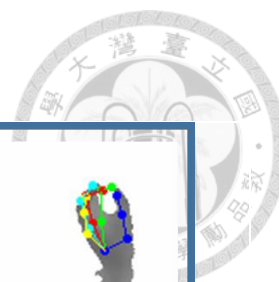


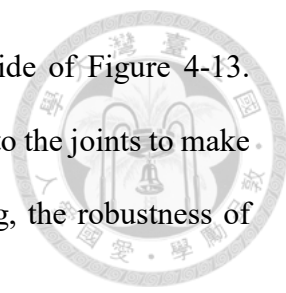
Figure 4-9. Some examples of the input, ground truth and prediction of our system on the NYU Hand Pose Dataset

4.3.3 Real World Testing

In this section, we are going to test our hand pose estimation system in the real world under the environment of both Linux and Windows, and the deep learning toolkit is Tensorflow. The depth camera and the Head-Mounted display (HMD) which we use in this testing is Intel RealSense F200 and HTC VIVE, respectively. RealSense is a kind of portable camera which can take RGB and depth image simultaneously with high quality. We attach the depth camera in front of Head-Mounted display so that we can obtain the

depth image of hand from the first person view. Figure 4-12 shows the picture of the Intel Realsense. HTV VIVE is a kind of Head-Mounted display which includes a helmet, two helmet detectors and two handles. Because the goal of our work is to provide a natural way to interact with the computer or virtual world, so we do not use the handles in our testing. VIVE can provide a good experience for the users by constructing a wonderful virtual world. Figure 4-11 shows the picture of the HTC VIVE.

For the testing under Linux environment, because the setting and usage of Head-mounted display is difficult, we only test our system with the depth camera. Figure 4-10 shows the predicted results of our system, which contains the real-world environment including the user with hand pose, depth image after the processing of threshold and the bounding box and estimated hand pose. The results show that even with some difficult poses, our system can estimate the hand pose correctly. Moreover, we can also crop the hand tightly with bounding box, which is the red rectangle, even with different hand pose or different distance from the depth camera and hand. For the testing under Windows environment, we will use the Head-Mounted device. We will then detail how we implement the hand pose estimation in the virtual world. Figure 4-13 shows how we combine our hand pose estimation system with the depth sensor and Head-Mounted Device (HMD) and provide a natural way to interact with the objects in the virtual world. As Figure 4-13 shows, we attach the depth sensor, which is Realsense, to the front of the HMD, which is HTC VIVE. Then, we obtain the depth image containing hand by the depth sensor and input into our hand pose estimation system and estimate the joint positions in the 3D space and project them into the virtual world. Once we have a virtual hand in the virtual world, we can do many kinds of pose like grabbing, pushing or pressing, and interact with the computer in a natural way by having interaction with the object in the virtual world. For example, we can press a bottom, pushing a box or pull a rope. We



can also play chess in the virtual world as shown in the bottom side of Figure 4-13. Moreover, as shown in Figure 4-14, we can also attach a hand model to the joints to make the hand more real. By showing the results in the real-world testing, the robustness of performance of our proposed hand pose estimation can be proved.

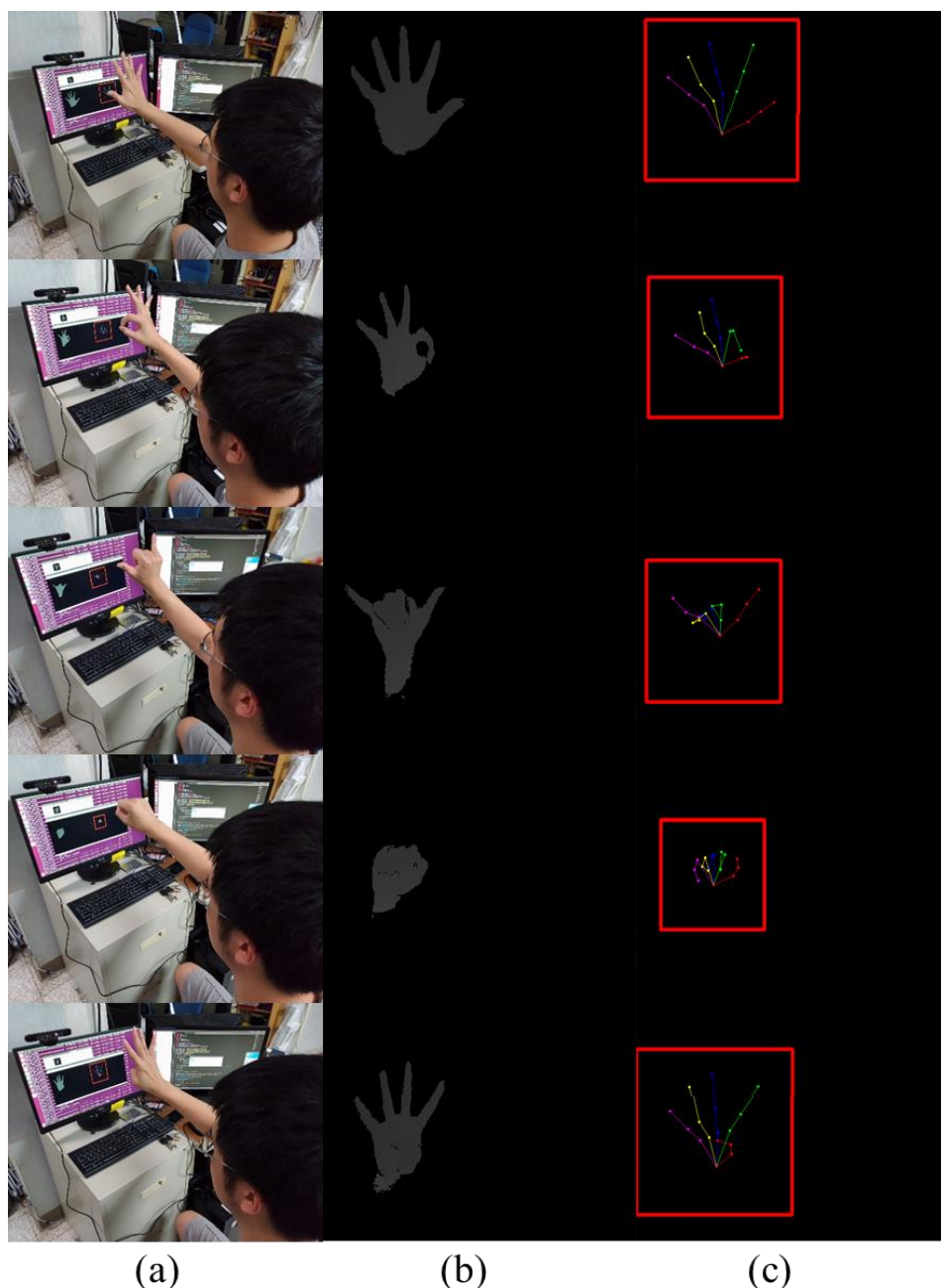


Figure 4-10. Some examples of the real-world testing. (a) shows the user and the hand pose. (b) shows the image after depth threshold. (c) shows the bounding box of the hand and the estimated hand pose.



Figure 4-11. Head-Mounted Device. This figure shows the Head-Mounted Device used in the testing, which is HTC VIVE.



Figure 4-12. Depth sensor. This figure shows the depth sensor used in the testing, which is Intel Realsense.

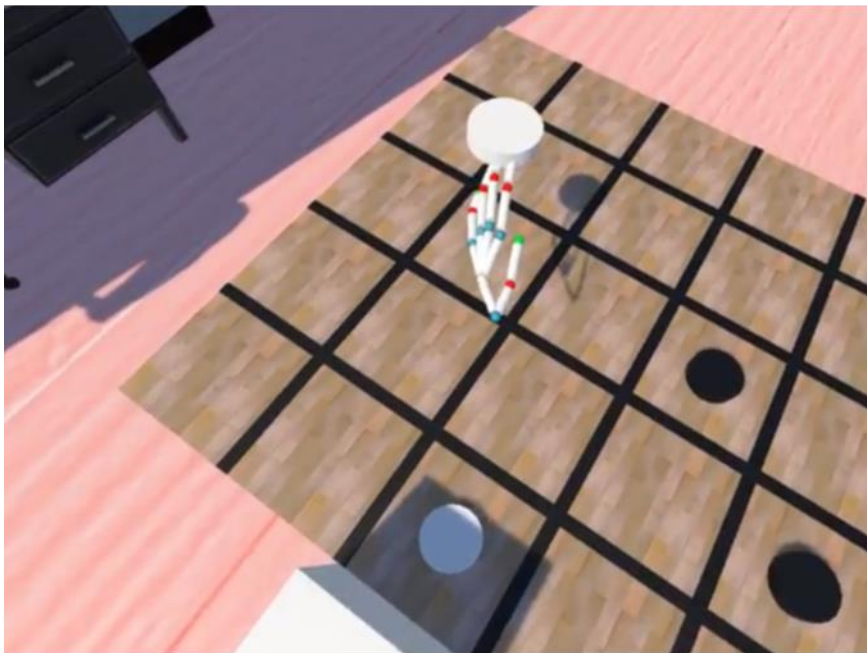


Figure 4-13. The user and the scene in the virtual world. The top figure shows the user with the HMD and depth sensor. The bottom figure shows the virtual world and the hand projected in it.



Figure 4-14. Hand model. This figure shows the illustration after attaching a hand model to the joints.

Chapter 5 Conclusion



We propose a real-time, accurate 3D hand pose estimation system including hand detection and 3D convolutional neural network to predict human hand pose from a depth image. In addition, even with a depth image which misses some parts, it can be well estimated by our proposed system. The experimental results which are evaluated on several public and challenging datasets show that our system works well under most of the conditions.

To overcome the problem of bad quality of depth image such like image broken or missing and perform well and steady, we propose a deep-learning method to train a 3D convolutional neural network (CNN) model and take some physical constraints into consideration. First, we transform the 2D depth image to 3D voxelized grid and propose a method to pad the data, which fills the missing part of the data and ease the learning process of CNN. Second, we add two additional loss function into CNN to improve the performance and steady the estimation result, which are finger-length-ratio layer and bone-length-ratio layer. For the finger-length-ratio layer, we constrain the length of each finger by taking the ratio of five fingers into consideration, and this can prevent the situation that the estimation of one of the finger is too long or too short. For bone-length-ratio layer, we constrain the length of the bone of each finger, which is defined by the joints, and this can maintain the steady of the estimation of the skeleton. With the additional knowledge mentioned above, our system is able to predict the hand pose more accurately.

Overall, our proposed system contains a hand detection part and a hand pose estimation part. In addition, we compare our method with other state-of-the-art methods on two public datasets and shows better performance. Therefore, we expect that our

proposed system can help to provide a real-time and accurate way for the interaction between human and computer, especially on the applications of the virtual reality (VR), augmented reality (AR) and mixed reality (MR) in our future life.

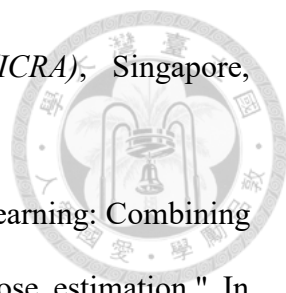


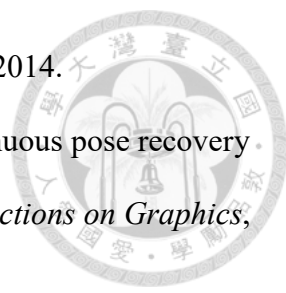
REFERENCE




- [1] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3D tracking of hand articulations using Kinect," in *British Machine Vision Conference*, 2011.
- [2] A. Tagliasacchi, M. Schroeder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, "Robust articulated-icp for realtime hand tracking," In *Computer Graphics Forum*, 34(5), 2015
- [3] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression, " In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [4] C. Wan, A. Yao, and L. Van Gool, "Direction matters: hand pose estimation from local surface normal," In *European Conference on Computer Vision*, 2016.
- [5] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," in *ACM Transactions on Graphics*, 33(5):169, 2014.
- [6] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [7] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "3d convolutional neural networks for efficient and robust hand pose estimation from single depth image," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [8] H. Guo, G. Wang, X. Chen, C. Zhang, F. Qiao, and H. Yand, "Region ensemble network: Improving convolutional network for hand pose estimation," in *IEEE International Conference on Image Processing*, 2017.
- [9] Chen, T.-Y., P.-W. Ting, M.-Y. Wu, and L.-C. Fu, "Learning a Deep Network with Spherical Part Model for 3D Hand Pose Estimation," presented at the *IEEE*

International Conference on Robotics and Automation (ICRA), Singapore, Singapore, 2017.

- 
- [10] M.-Y. Wu, Y.-H. Tang, P.-W. Ting, and L.-C. Fu, "Hand Pose Learning: Combining Deep Learning and Hierarchical Refinement for 3D Hand pose estimation," In *British Machine Vision Conference*, 2017
- [11] M. Oberweger, P. Wohlhart, and V. Lepetit, "Training a feedback loop for hand pose estimation," In *IEEE International Conference on Computer Vision*, pages 3316–3324, 2015.
- [12] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands deep in deep learning for hand pose estimation," In *Computer Vision Winter Workshop*, pages 21–30, 2015.
- [13] M. Oberweger and V. Lepetit, "Deepprior++: Improving fast and accurate 3d hand pose estimation," In *IEEE International Conference on Computer Vision*, Oct 2017.
- [14] C. Wan, T. Probst, L. Van Gool, and A. Yao, "Crossingnets: Combining gans and vaes with a shared latent space for hand pose estimation," In *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.
- [15] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [16] S. Song and J. Xiao, "Deep Sliding Shapes for amodal 3D object detection in RGB-D images," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [17] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D shapenets: A deep representation for volumetric shapes," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [18] D. Tang, H. Jin Chang, A. Tejani, and T.-K. Kim, " Latent regression forest: Structured estimation of 3d articulated hand posture," In *IEEE Conference on*

- 
- Computer Vision and Pattern Recognition*, pages 3786 – 3793, 2014.
- [19] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Realtime continuous pose recovery of human hands using convolutional networks," In *ACM Transactions on Graphics*, 33(5):169, 2014.
- [20] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim, "Bighand2.2m benchmark: Hand pose dataset and state of the art analysis," In *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.
- [21] Krizhevsky, A., I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.
- [22] Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [23] Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1-9.
- [24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [25] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," In *ISMAR*, 2011.
- [26] S. Song, F. Yu, A. Zeng, A. Chang, M. Savva, and T. Funkhouser, "Semantic Scene Completion from a Single Depth Image," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [27] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig



Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

- [28] S. Melax, L. Keselman, and S. Orsten, "Dynamics based 3d skeletal hand tracking," In Proceedings of *Graphics Interface*, 2013, pages 63–70. Canadian Information Processing Society, 2013.
- [29] Zhang, Z., "Microsoft kinect sensor and its effect," *IEEE multimedia*, vol. 19, pp. 4-10, 2012.
- [30] H. Guo, G. Wang, X. Chen, and C. Zhang, "Towards good practices for deep 3d hand pose estimation," *arXiv preprint arXiv:1707.07248*, 2017.
- [31] X. Chen, G. Wang, H. Guo, and C. Zhang, "Pose guided structured region ensemble network for cascaded hand pose estimation," *arXiv preprint arXiv:1708.03416*, 2017.