Training for hand pose estimation with augmented dataset

handed in
FORSCHUNGSPRAXIS

Yongchen Lu

born on the 26.01.1993
living in:
Schrofelhofstr. 12
81369 Munich
Tel.: 04917645957319

Chair of
AUTOMATIC CONTROL ENGINEERING
Technical University of Munich

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Dirk Wollherr

Supervisor:	Shile Li
Start:	01.12.2016
Delivery:	27.04.2017

TECHNISCHE UNIVERSITÄT MÜNCHEN
LEHRSTUHL FÜR STEUERUNGS- UND REGELUNGSTECHNIK
UNIV.-PROF. DR.-ING./UNIV. TOKIO MARTIN BUSS

01. Nov. 2016

F O R S C H U N G S P R A X I S
for
Yongchen Lu
Student ID 03669350, Degree EI

Training for hand pose estimation with augmented dataset

Problem description:

Hand pose estimation plays an important role in some human-robot interaction tasks, such as gesture recognition and learning grasping capability by human demonstration. Since emergence of consumer-level depth sensing device, a lot of depth image based hand pose estimation methods appeared.
For learning based methods, a large dataset for training is required to obtain good performance [1, 4]. Existing hand pose datasets usually only contain clean hand images, where the hand is not in contact with other objects [3, 2]. This constrains the trained models to be applied on hand-object interaction scenarios. To overcome this issue, a hand pose dataset that contains a lot of occlusion cases, is needed, because occlusion often appears during hand-object interaction.
In this Forschungspraxis, the student will add artificial occlusion to existing dataset [3], in order to simulate hand-object interaction cases. The student will then use the augmented dataset to train a neural network following [1], and test the performance in real hand-object interaction scenario.

Work schedule:

- Literature review on hand pose estimation.
- Create augmented dataset.
- Training using augmented dataset.
- Test performance on real captured data.

Bibliography:

[1] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015.
[2] Danhang Tang, Hyung Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3d hand poses. *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.
[3] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin.
[4] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. Model-based deep hand pose estimation.

Supervisor:	M. Sc. Shile Li
Start:	01.12.2016
Delivery:	27.04.2017

(D. Lee)
Professor

Abstract

Hand pose estimation plays an important role in industrial area, especially in some human-robot interaction tasks, such as gesture recognition for robots. With the development of depth sensing device, there are a lot of hand pose estimation methods based on depth image. For learning based methods, a good performance of the system is based on a large dataset. There are some hand pose dataset online such as NYU dataset. Those datasets usually only contain clean hand images, which means the hand has no contact with other objects. But in real application the interaction scenarios are very common. To overcome this problem and let the system become more robust, an augmented hand dataset which contains a lot of occlusion cases is needed for training of the system. We use Convolutional Neural Networks to predict the joint locations of a hand based on augmented hand dataset. Firstly artificial occlusion is added to existing dataset to simulate occlusion cases. Then the augmented hand dataset is used to train the neural network and finally the performance of the system in real hand-object occlusion scenario is evaluated.

Contents

Chapter 1

Introduction

Human hand pose estimation plays an important part in various applications in human-computer interaction. Scientists and researchers have studied such topic in computer vision for decades and made great breakthrough recently due to the implementation depth cameras. The problem is challenging due to the highly articulated structure, significant self-occlusion and viewpoint changes [XZ16].
In most cases the robots need to recognize human hand pose then they will execute the related tasks for human. They would make use of the camera to capture the image and analyse the image to estimate the hand pose due to the captured image. As a result, it is required to design a system for robot that the input is image and the output is joint information of the hand which can represent the hand pose. Additionally, the hand occlusion cases and the prediction accurancy need to be taken into account.
In our work the system is a learning based convolutional neural network. There are some datasets online like NYU dataset for training of the system. But those datasets only contain clean hand and no occlusion cases. Therefore a new dataset with occlusion cases is required. We created augmented dataset by adding some generated occlusions on the existing dataset to simulate the real occlusion cases. It is assumed the system trained by the augmented dataset can handle occlusion cases in in real scenario.
More specifically, our works are as follows:

- Generate the augmented dataset based on the existing dataset. In the generated dataset the hand is occluded randomly. The form of the occlusion is randomy chosen from circle and square. The size of the occlusion is also random. For better comparasion of the result, the size of the occlusion is limited to 1%, 4%, 9% and 16% of the size of the hand part. The augmented dataset is generated by preprocessing the existing NYU hand pose dataset.

- Generate the neural netwrok for training. The network is trained end-to-end via standard back-propagation. Additionally two special layers, “bottleneck” layer and a PCA layer, are added to the network . ”Bottleneck” layer generates

fewer neurons than the last layer and PCA layer generates the same number of output as the label input to calculate the loss. This mechanism let the system learn the geometry of the hand automatically and predict the hand pose more accurate.

- Implement the trained neural network into real hand pose estimation. Using the depth camera to capture the real image, cropping the bounding box of the hand and inputing the preprocessed image into the neural network system to realize the hand pose estimation. In implementation the hand is occluded by real object, on the other word, the hand will grasp something and the intersection between hand is preprocessed (to be set as same as the background in implementation).

1.1 Related Work

In Computer Vision the hand pose estimation is an classical problem, but it is now very popular probabily because of the development of hardware, like depth sensors and cameras. A good overview of earlier work is given in Coates's paper [AC11]. Here we will discuss only more recent work, existing methods can be roughly categorized as two complementary paradigms, model based (generative) or learning based (discriminative).
Model base method depends highly on the hand geomentry. Firstly they would define an energy function to quantify the discrepancy between the synthesized and observed images, and optimize the function to obtain the hand pose. After the dedicated optimization, the final obtained pose could be highly accurate. Oikonomidis and Qian use a 3D hand model and Particle Swarm Optimization to handle the large number of parameters to estimate [IO11] [CQ14]. Melax also considers dynamics simulation of the 3D model [SM13]. All these works require careful initialization in order to guarantee convergence and therefore rely on tracking based on the last frames pose or separate initialization methods. Such tracking-based methods have difficulty handling drastic changes between two frames, which are common as the hand tends to move fast.
Learning based methods use machine learning methods. Given image appearance and hand pose information, such system could find the relationship between them after learning. For example, Huber and Ionescu's methods rely on multi-layered Random Forests for the prediction [Hub64] [CI14]. The former uses invariant depth features, and the latter uses clustering in hand configuration space and pixel-wise labelling. However, both do not predict the actual 3D pose but only classify given poses based on a dictionary. Nowdays with the popularity of deep learning the convolution neural network is widely used in solving such problem. Evaluating the regression function is usually much more efficient than model based optimization.
All of those learning based methods only use clean hand image, which means the hand is presented without any occlusion cases before the camera. But in real appli-

cation the hand will interact with other objects. Under such condition the system should be robust and predict the hand pose correctly. In order to train the system we need to generate the augmented dataset at first. In this dataset the original hand will be randomly occluded.

Chapter 2

Technical Approach

2.1 Hand Model

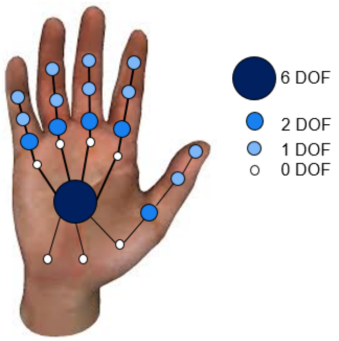


Figure 2.1: Illustration of hand model.

Human hand can be represented by the joints and the connections between them. As illustrated in Figure 2.1, the hand pose is 26 degrees of freedom (DOF), defined on 23 joints. There are 3 DOF for global palm position, 3 DOF for global palm orientation. The remaining DOF are rotation angles on joints. Each joint contains 3 axises u, v, d which represent the position and depth information in depth image. But in our work we do not use all of the joints to evaluate the precision of the hand pose estimation, for example in the Oberweger's experiment only 14 joints of the hand are used [XZ16]. So the same 14 joints in the dataset are selected for the simplicity of the dataset creation and better comparasion with previous results.

2.2 Creation of Augmented Dataset

To simplify the regression task, at first it is necessary to estimate a coarse 3D bounding box containing the hand using a simple method, by assuming the hand is

the closest object to the camera: a fixed-size cube centered on the center of mass of this object is extracted from the depth map, and is resized to a 128 x 128 patch of depth values normalized to (-1, 1). The aim is that the only interested part in depth image is the hand, the other objects like environment and human body shall be eliminated. With effects of light sensors some points depth are not available or are deeper than the back face of the cube. Those depths are assigned the value of 1. This normalization is important for the training of the system.
After selecting the hand part in depth image, some occlusions are added on the existing hand part. Because the final interested part is still the hand, the occlusion part will be set to the same value as the points for which the depth is not available. As mentioned before, the size of the occlusions, the position of occlusions and the form of the occlusions are random. In order to simplify the problem, finally we limit the form of occlusions to circle and square. The size of occlusions are limited to 1%, 4%, 9%, 16% of the hand part.
Take the 1% square occlusion for example: Because the patch size is 128 x 128 so the square occlusion size is 1.28 x 1.28. Based on the center of the patch and the random functions in the software the position of the up-left corner of the square is decided. From this up-left coner a square area with the size 1.28 x 1.28 can be selected and assigned the value of 1. For other size and the circular occlusion the procedure are similar. The Figure 2.2 shows the samples from the augmented dataset.

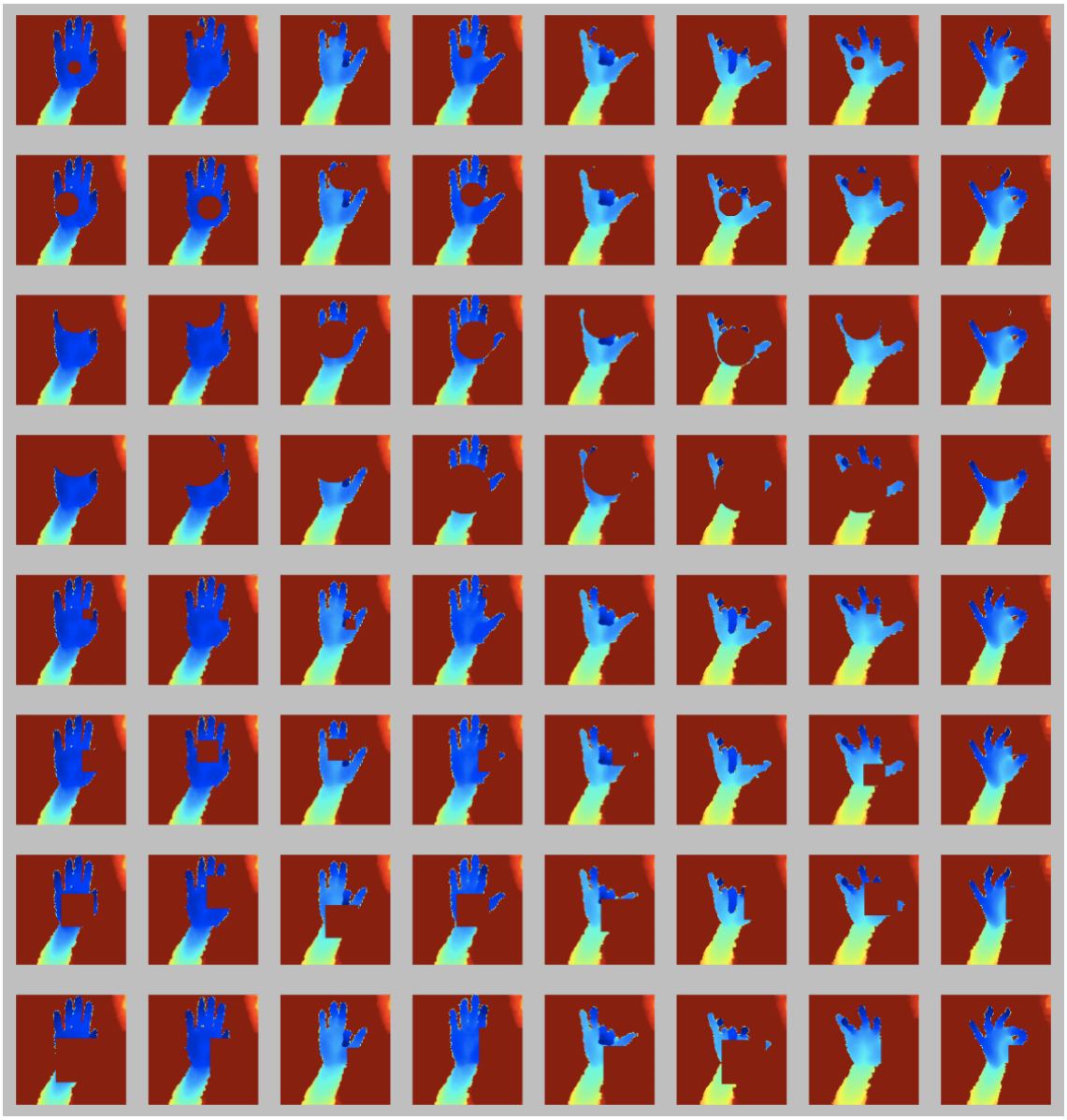


Figure 2.2: Samples of augmented dataset. The first 4 lines are circural occlusion and the last 4 lines are square occlusion. The size of occlusions are limited to 1%, 4%, 9%, 16% of the hand part respectively. The occlusions on hand are set to the same value of the points for which the depth is not available (like background).

In real implementation the reference point will be fluctuated from the ground truth and the ground truth is unknown. A suitable way is to randomize the reference point, including the position and the depth. But there is a trade off between the random range and the precision. If the random range is very wide then the final precision is very low because for certain random value the data become fewer. It is assumed that reference point is not far away from the ground truth, then the

random range is set to a reasonable small value. And with more data even wide random range is also acceptable.

2.3 Convolutional Neural Network Structures

In CNN there are several typical layers such as convolutional layer, max-pooling layer and ReLU layer.
The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of neurons, which have a small receptive field, but extend through the full depth of the input volume. During the forward process, neurons compute a dot product of their weights with the input and are followed by a non-linearity function. The Figure 2.3 shows the neuron model and the equation (2.1) explains the detail computation.

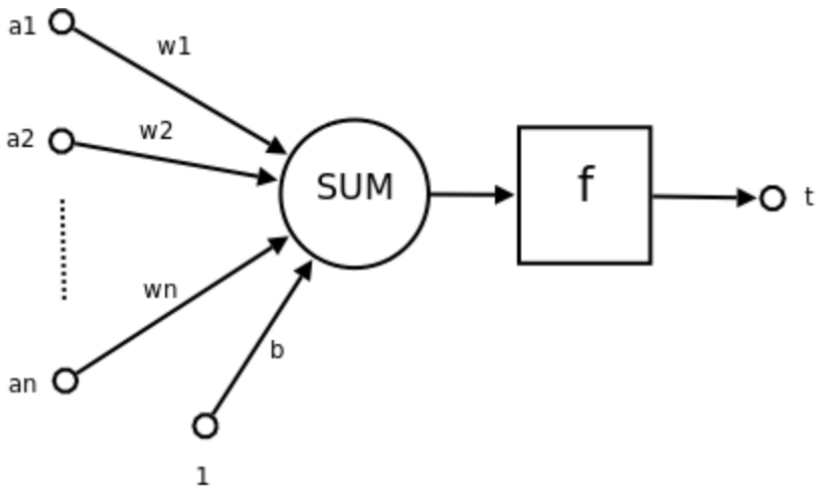


Figure 2.3: The neuron model. They compute a dot product of their weights with the input and are followed by a non-linearity function.

$$t = f(\sum_i W_i * a_i + b) \tag{2.1}$$

ReLU is the abbreviation of Rectified Linear Units. This is a layer of neurons that applies the non-saturating activation function:

$$f(x) = max(0, x) \tag{2.2}$$

Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent $f(x) = tanh(x)$ and the sigmoid function $f(x) = (1 + e^{-x})^{-1}$. Compared to other functions the usage of ReLU is preferable, because it results

in the neural network training several times faster, without making a significant difference to generalisation accuracy.
Another important layer is max-pooling layer. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. An explaination is showed in Figure 2.4.

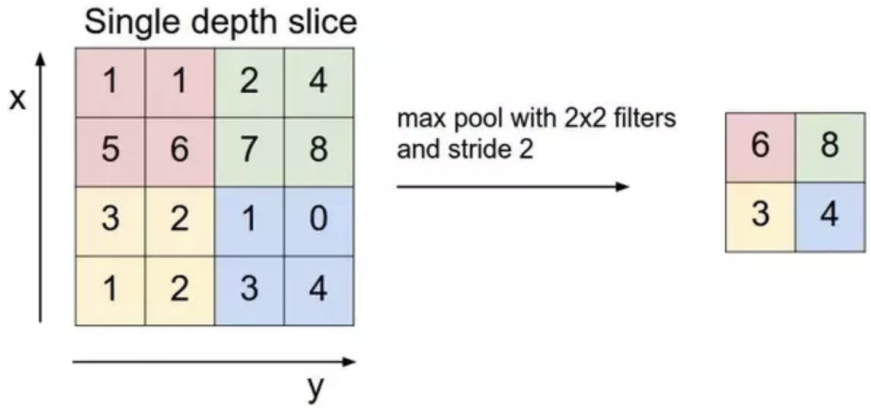


Figure 2.4: An explaination of the operation in max-pooling layer.

We first considered two standard CNN architectures. The first one is a simple shallow network, which consists of a single convolutional layer, a max-pooling layer, and a single fully-connected hidden layer. The second architecture is a deeper but still generic network, with three convolutional layers followed by max-pooling layers and two fully-connected hidden layers. All layers use Rectified Linear Unit activation functions. Finally, we chose the second approach, as shown in Figure 2.5.

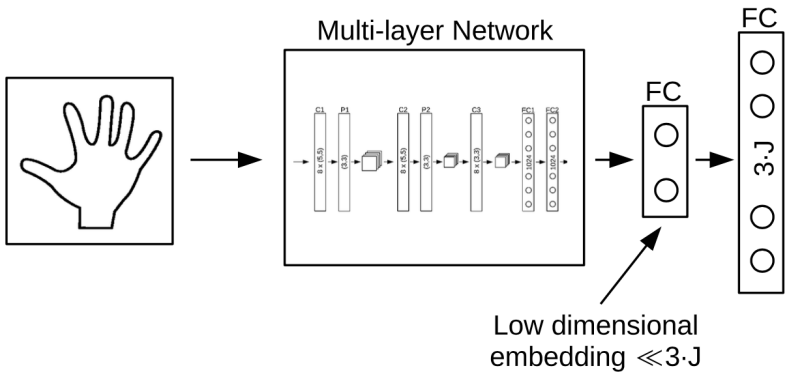


Figure 2.5: The implementation of Convolutional Neural Network with "Bottleneck" layer and PCA layer.

2.4 "Bottleneck" Layer and PCA Layer

Two special layers "Bottleneck" layer and PCA layer are added for hand geometry learning for the system. If standard neural network structure is implemented then the final predicted hand pose will be abnormal. The reason for that is the hand is restricted by physical factors. For example the thumb cannot turn to the wrist. In order to take those physical factors into account, two special layers "Bottleneck" layer and PCA layer are added.

"Bottleneck" means the output of this layer is fewer than the input. The aim is to let the system learn the physical constraints automatically in this way. Additionally, to keep the consistency of the related work in Oberweger's method, after the "Bottleneck" layer 42 neurons are generated from the last layer. But for final loss calculation the size of the prediction and the ground truth shall be the same. Apparently the 42 is smaller than the size of the ground truth size 108. So some reconstruction process is required. That is the function of PCA layer.

Principal component analysis (PCA) is a statistical transformation that convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. In other words, it is a linear transformation that chooses a new coordinate system for the data set such that greatest variance by any projection of the data set comes to lie on the first axis (then called the first principal component), the second greatest variance on the second axis, and so on. PCA can be used for reducing dimensionality by eliminating the later principal components.

Now we list all of our joints data as a data matrix, X (n x p)(the column-wise empirical mean can be calculated and substracted from X at first. Subtracting the mean is equivalent to translating the coordinate system to the location of the mean.), where each of the n rows represents a different result of observation, and each of the

p columns gives a particular data of each joint. Then we calculate the covariance matrix responding to p columns. The covariance formula is as follows:

$$\mathrm{cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - E(X))(y_i - E(Y)). \tag{2.3}$$

After that the eigenvectors and eigenvalues of the covariance matrix are calculated. Sorting the eigenvectors according to their eigenvalue and eliminating the later principal components will lead to the dimensionality reduction. Similarly the inverse PCA will cause the increase of the dimensionality.
Now the number of output is larger than the input. With the help of the inverse PCA process the same number of output as the ground truth is generated. In Matlab most of the data can be calculated directly, including empirical mean, covariance matrix and inverse covariance matrix. We save the data of the inverse covariance and empirical mean and feed those data to the initial weights of the PCA layer. Then the network is trained using back-propagation. With "Bottleneck" layer and PCA layer the system learn the geometry of the hand automatically and predict the hand pose more accurate.

2.5 Real Hand Pose Estimation

The final step is to implement the trained neural network into the real hand pose estimation. Here the depth camera is used. In practice we use the ASUS Xtion Pro to capture the color image and depth image. Because there are two cameras on the device who are responsible for color image and depth image capture separately, so the calibration is needed to be set at first.
Color image is used in preprocessing such as interesting hand part cropping and the object occlusion on hand detection and separation for depth image. Another important problem is that in implementation the ground truth of the joint position in hand is unknown. In augmented dataset creation we select the wrist point as the reference point. So it is required to find the reference point at first. In other words in real hand pose estimation the reference point and the object shall be separated automatically. For this operation using different color is a good method because some softwares (opencv) can distinguish different color and process them. In order to better realize the color separation the HSV color space is implemented. HSV are the most common cylindrical-coordinate representations of points in an RGB color model. HSV is cylindrical geometries with hue and his angular dimension, starting at the red primary at 0°, passing through the green primary at 120°and the blue primary at 240°, and then wrapping back to red at 360°. In geometry, the central vertical axis comprises the neutral, achromatic, or gray colors, ranging from black at value 0, the bottom, to white at value 1, the top. The explaination is in Figure 2.6.

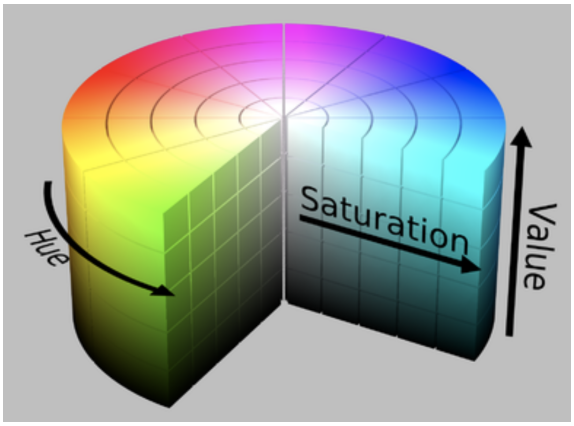


Figure 2.6: HSV cylinder.

For calculation of the H,S,V, the formula are as follows:

$$M = max(R, G, B); m = min(R, G, B); C = M - m \tag{2.4}$$

$$H' = \begin{cases} undefined & if C = 0 \\ \frac{G-B}{C} mod 6 & if M = R \\ \frac{B-R}{C} + 2 & if M = G \\ \frac{R-G}{C} + 4 & if M = B \end{cases}; H = 60 * H' \tag{2.5}$$

$$V = M \tag{2.6}$$

$$S_{HSV} = \begin{cases} 0, & if v = 0 \\ \frac{C}{V}, & otherwise \end{cases} \tag{2.7}$$

From the experience the blue and green color are the two best choices. So we implement those two colors in reference point and the object occlusion. Firstly the green parts in the color image are separated. The HSV for green is around (90,100,100), so the upper limitation value (100,360,360) and lower limitation value (40,60,60) is set. That means the points in the color image whose HSV value is in the range between upper and lower limitation will be separated. Then the biggest part of separated area is selected and the center position of this part is considered as the reference point.
Figure 2.7 (a) shows one example of captured color image by the depth camera. User wears a band and green point on the band is used to donate the reference point. The object grasped by the user is blue. The separation result is shown in Figure 2.7 (b). The black points represent the green parts in color image. Because of the noise in environment there are several green parts with small area but on the whole he green notation on the band is the biggest part. After separation the yellow circle denotes this biggest green area.

(a) The color image. (b) Processed color image.

Figure 2.7: Reference point selection. Because of the noise in environment there are several green parts with small area but on the whole he green notation on the band is the biggest part. After separation the green circle denotes this biggest green area.

With the selected reference point the interested part of the hand can be selected. But this operation is executed in the depth image because the input for the system is depth image. After the cropping operation a cube of the depth image is selected like the preprocessing operation in original dataset. One example is shown in Figure 2.8 (a). In real implementation the object is in blue and the object can be separated as well. After the preprocessing of separation one example is in Figure 2.8 (b). Now this processed image can be imported to the trained neural network to predict the different joints of the hand pose.

(a) Cropped area of hand with object occlusion.

(b) Processed cropped area of hand without object occlusion.

Figure 2.8: Cropped area of hand with and without object occlusion.

Also the depth of reference point is also critical. In dataset the depth is in hand but in depth image the depth on the position of reference point is on hand. In

order to avoid the problem caused by the error of the reference point, in dataset creation we randomize the bounding box a little. In this way the final trained neural network can accept some error caused by the fluctuation of the reference point and the system will become more robust.

When depth camera works, the environment needs to be clean, which means it contains less noise. In our experiment the background is white wall. Much noise will affect the depth information record and color separation. Also the depth camera has minimal distance for normal working, it is necessary to pay attention to these factors.

Chapter 3

Evaluation

In this section we evaluate the architecture introduced in the previous section on created augmented dataset. Then the performance on real hand pose estimation is analyzed.

3.1 Benchmarks

Firstly benchmarks and the parameters are introduced. The augmented dataset is based on the following dataset:
NYU Hand Pose Dataset: This dataset contains over 72k training and 8k test frames of RGB-D data captured using the Primesense Carmine 1.09. It is a structured light-based sensor and the depth maps have missing values mostly along the occluding boundaries as well as noisy outlines. For experiments we use only the depth data. The dataset has accurate annotations and exhibits a high variability of different poses. The training set contains samples from a single user and the test set samples from two different users. The ground truth annotations contain J = 36 joints, however Oberweger uses only J = 14 joints, and we did the same for comparison purposes [MO15].

3.2 Parameters and Optimization

The network is trained by minimizing the distance between the prediction and the expected output per joint, and a regularization term is implemented for weight decay to prevent over-fitting, where the regularization factor is 0.001. The networks are trained with back-propagation using Stochastic Gradient Descent with a batch size of 128 for 100 epochs. The learning rate is set to 0.01 and a momentum is of 0.9.

3.3 Evaluation Metrics

Two different evaluation metrics as Oberweger's method are used [MO15] :

- the average Euclidean distance between the predicted 3D joint location and the ground truth, and
- the fraction of test samples that have all predicted joints below a given maximum Euclidean distance from the ground truth, as was done in Taylor's method [JT12]. This metric is generally regarded very challenging, as a single dislocated joint deteriorates the whole hand pose.

3.4 Evaluation With Original Dataset

Before stepping into the training on the augmented dataset, the evaluation on the original dataset is performed. The result figures are as following:

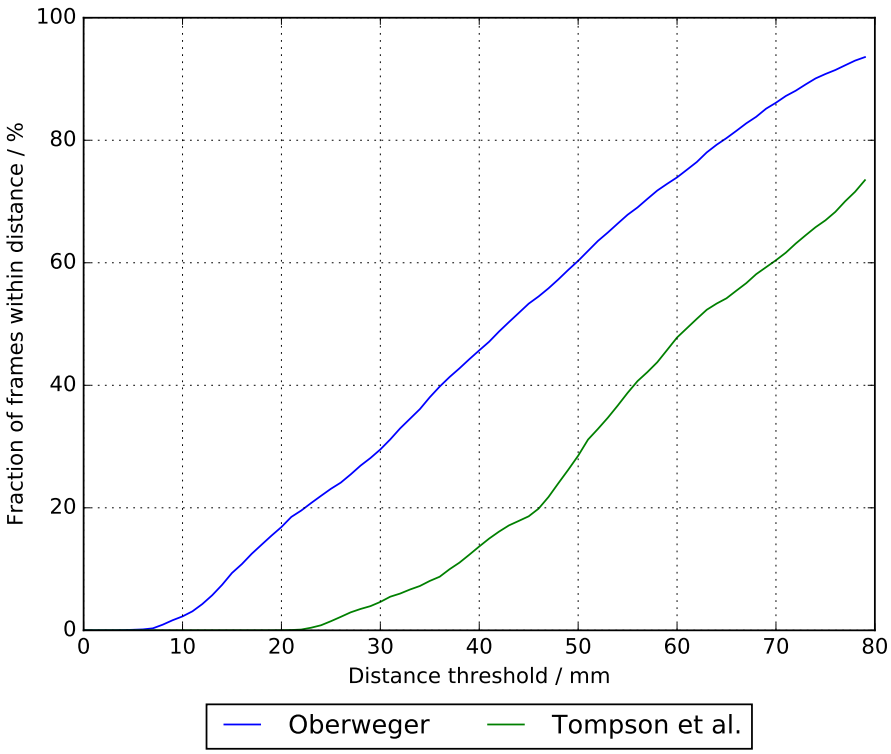


Figure 3.1: Fraction of frames where all joints are within a maximum distance. A higher area under the curve denotes more accurate results. From the plot it is clear that the Oberweger's method is much better than Tompson et al. method.

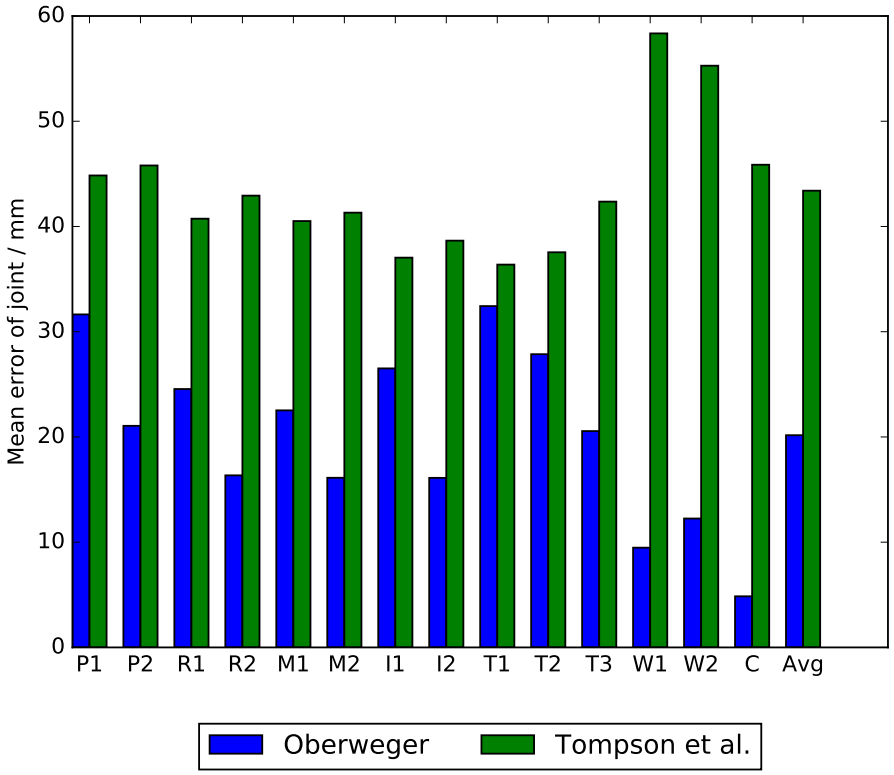


Figure 3.2: Average joint errors. For completeness and comparability we show the average joint errors, and the results are consistent. It is showed that for each joint the Oberweger's method is much preciser than Tompson et al. method.

We evaluate the fraction of frames where all joints are within a maximum distance for different approaches. A higher area under the curve denotes more accurate results. From the Figure 3.1 it is clear that the Oberweger's method is much better than Tompson et al. method.
For completeness and comparability we show the average joint errors, which are, however, not as decisive as the evaluation in Figure 3.1. Though, the results are consistent. The evaluation of the average error is more tolerant to larger errors of a single joint, which deteriorate the pose as for Figure 3.1, but are insignificant for the mean if the other joints are accurate. From the graph it is also clear that for each joint prediction in handpose the Oberweger's method is much more accurate.

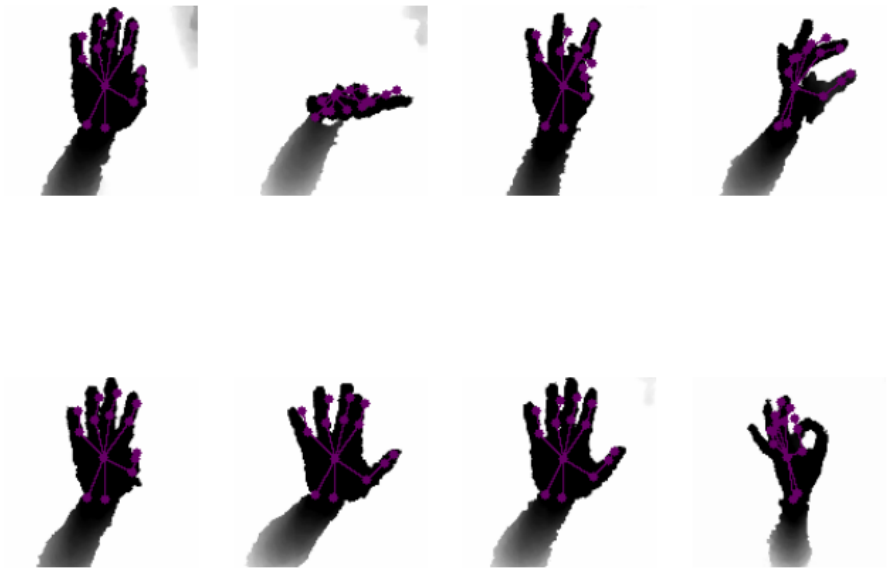


Figure 3.3: Qualitative results. We show the inferred joint locations on the depth images (in gray-scale). The ground truth is shown in gray, the predicted results are in purple. The circle represents the joint of the hand and the line is the connections between joints. From the graph one can see that the predicted joints can approximately represent the hand pose.

Figure 3.3 shows the qualitive result of the prediction based on Oberweger's method. We show the inferred joint locations on the depth images (in gray-scale). From the figure it is obvious that the predicted joints can represent the original hand pose. In most cases it is almost correct but there is still some exceptions like the thumb in eighth picture of Figure 3.3. Our experiment shows that the network in Oberweger's method is trained well. The joint loss is well behaved as the errors spread over different parts. This is important for learning an articulated structure like hand. Also the "Bottleneck" layer and the PCA layer in the network are of great importance. They help the cost function to convergen quickly which means the optimal point could be reached quickly. At the same time due to those two layers, the network avoids the risk of over-fitting because the network will learn the geometry constraints automatically. If given more training data the result will become more accurate. For the training of the augmented dataset we will use the same structure. And the trained parameters (weights and bisas in different layers)

will be saved for further training in order to save the training time.
For training it does not mean the more iterations the higher precision the system finally gains. At first the more iterations leads to higher precision but then the system goes into a state where the precision keeps stable, after that the precision continues to decrease. The stable state is a good state to stop the training because after that the decrease of the precision is caused by the overfitting. Also time saving benefits from the fewer iterations. Now 20000 training iterations are executed.
Hardware plays an important role in the training phase. Now in caffe the data can be processed on GPU. Yet with Nvidia Geforce 940M (2G) GPU the total training process last 2 hours. If larger memory GPU provided, the time consuming can be greatly decreased.

3.5 Evaluation With Augmented Dataset

As prescribed the occlusions are limited to the form of circle and square. The size of occlusions are restricted to 1%, 4%, 9%, 16% of the hand part. In order to evaluate them effectively, different forms of occlusions are compared separately and in certain form of occlusion the prediction of each size is evaluated independently.
The result figure is as Figure 3.4. The figure shows fraction of frames where all joints are within a maximum distance responding to different square occlusion and the baseline no occlusion. From the figure it is clear that the bigger of the square occlusion, the less precision the final prediction will gain. A higher area in the figure under the curve denotes more accurate results. The result of no occlusion is on the top of all lines because there is no occlusion in test dataset which means the dataset provides more information of hand pose. On the contrary 16% occlusion is the biggest occlusion which hides the most information of the hand and the system needs to predict the joint information in the occlusion part. As a result the final prediction is of less precision.

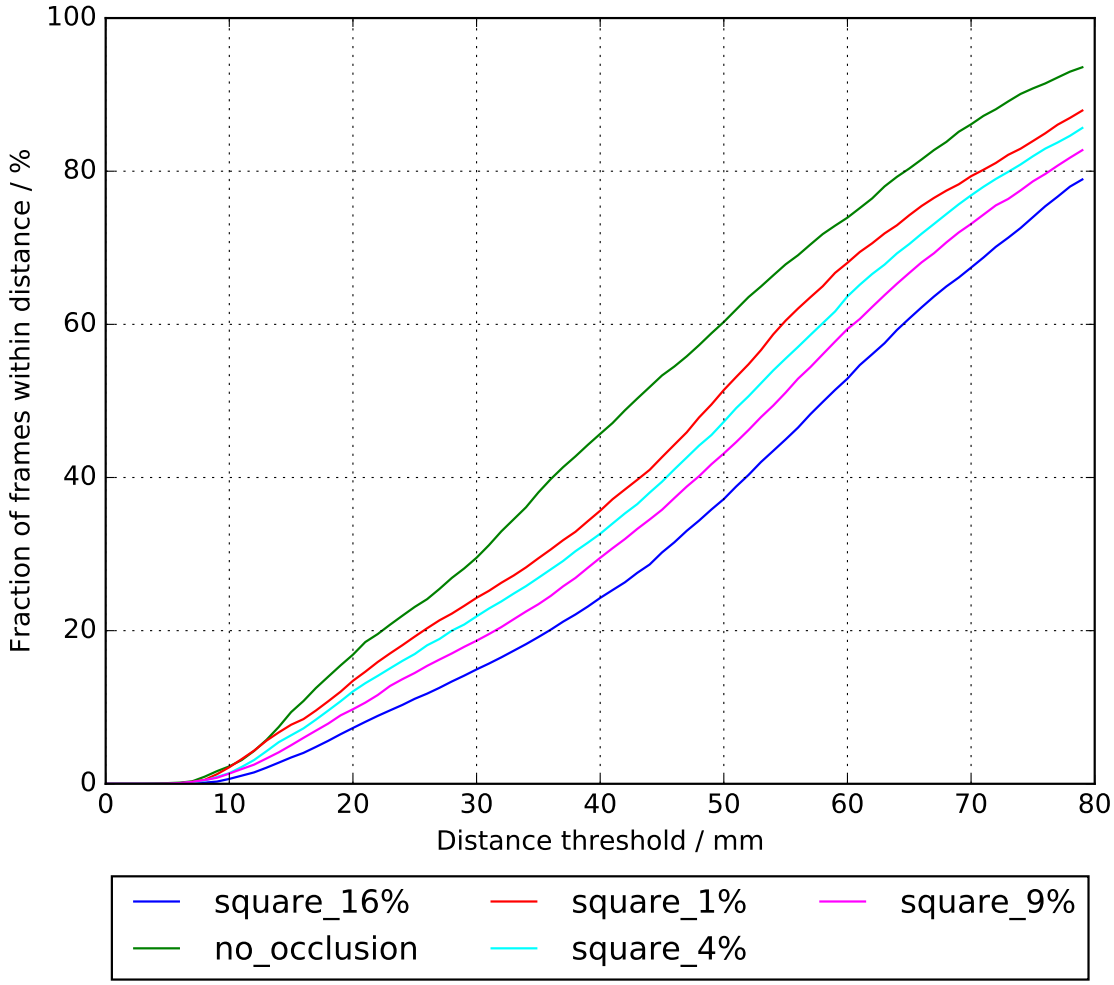


Figure 3.4: Fraction of frames where all joints are within a maximum distance in different square occlusion. A higher area under the curve denotes more accurate results.

The consistent result is showed corresponding to the average joint errors, which are in Figure 3.5. As discussed previously the evaluation of the average error is more tolerant to larger errors of a single joint, which deteriorate the pose as for Figure 3.5, but are insignificant for the mean if the other joints are accurate. From the graph it is also clear that for each joint prediction in hand pose with less occlusion is much more accurate. The same result can be seen in circular occlusion which are showed in Figure 3.6 and Figure 3.7.

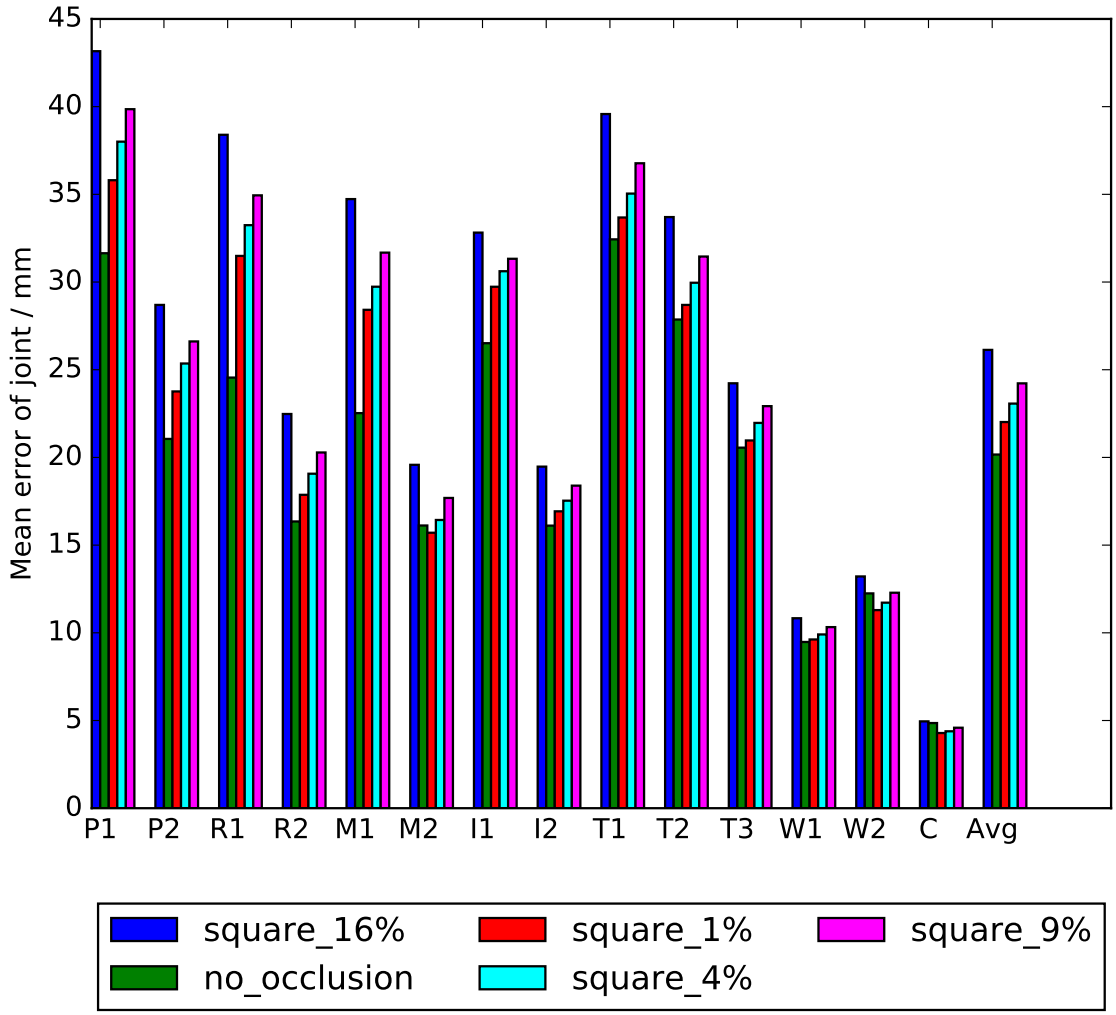


Figure 3.5: Average joint errors in different square occlusion. It is showed that for each joint with less occlusion is much preciser than that with larger occlusion.

3.6 Evaluation in Real Hand Pose Estimation

The final step is to predict the hand pose in real implementation. The depth camera captures the color image and find the reference point according the notation on the color image. The detailed process is introduced in previous chapter. Here the final prediction precision is compared among the trained neural network based on different dataset. Because for real implementation no ground truth is provided so the performance can only be compared through the consistency between original hand pose image and predicted joint position.
Finally there are 2 trained neural network. The one is based on original dataset and the other is based on augmented dataset. There are two kinds of performance: the

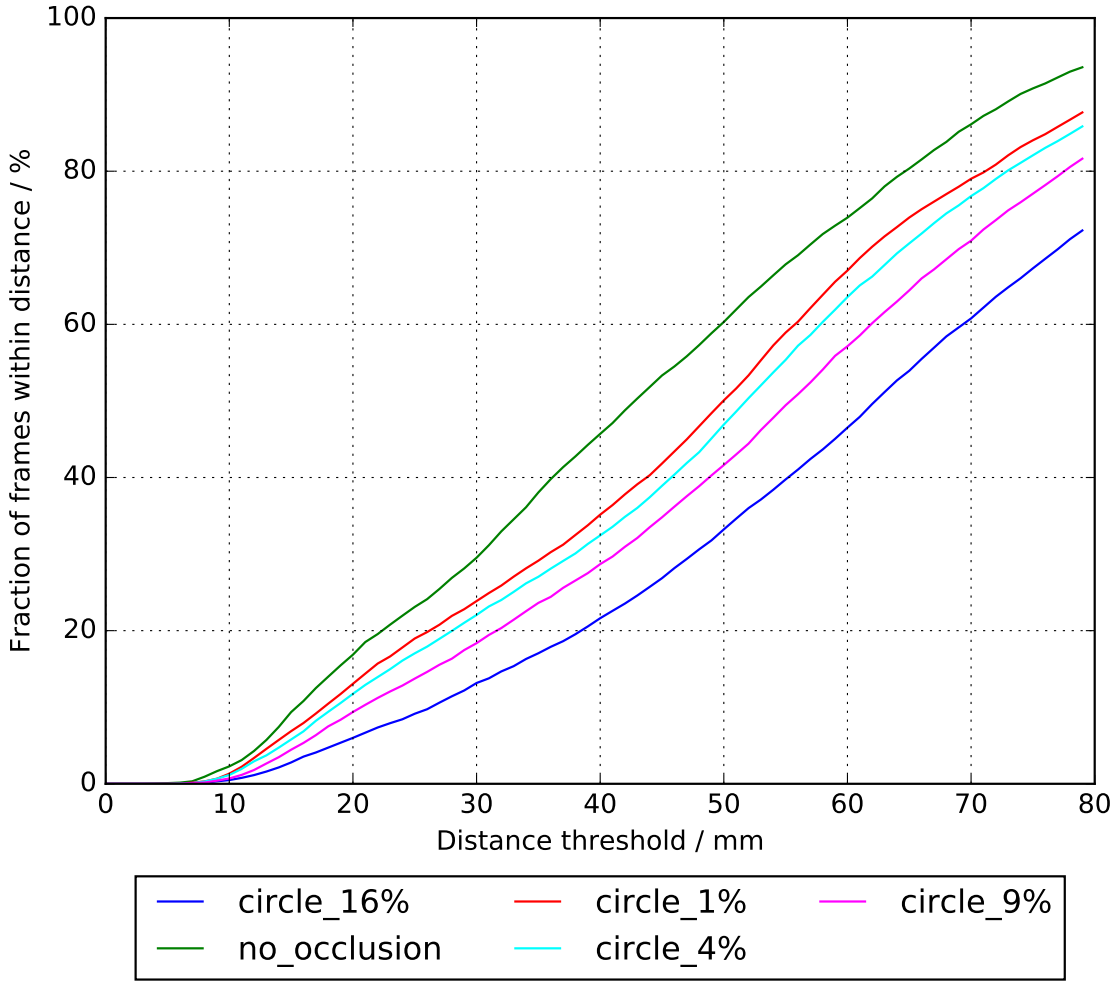


Figure 3.6: Fraction of frames where all joints are within a maximum distance in different circular occlusion. A higher area under the curve denotes more accurate results.

augmented dataset based neural network performs better than the original dataset based neural network and both of the neural network perform similar. The results are shown in Figure 3.8 and Figure 3.9.
From the Figure 3.8 it is obvious that the trained neural network predicts the joint position more precise which means the final predicted joint positions are closer to the original joint position in hand pose image. The reason for that is the augmented dataset based trained neural network has learned those occlusion cases, so the final joint pose prediction is more robust to the different occlusions on hand. But for the original dataset based trained neural network such occlusion is a new case, so the final prediction is less precise. And it turns to the default hand pose which is decided by the parameters in PCA layer.
However for some hand poses the final predictions between both trained neural

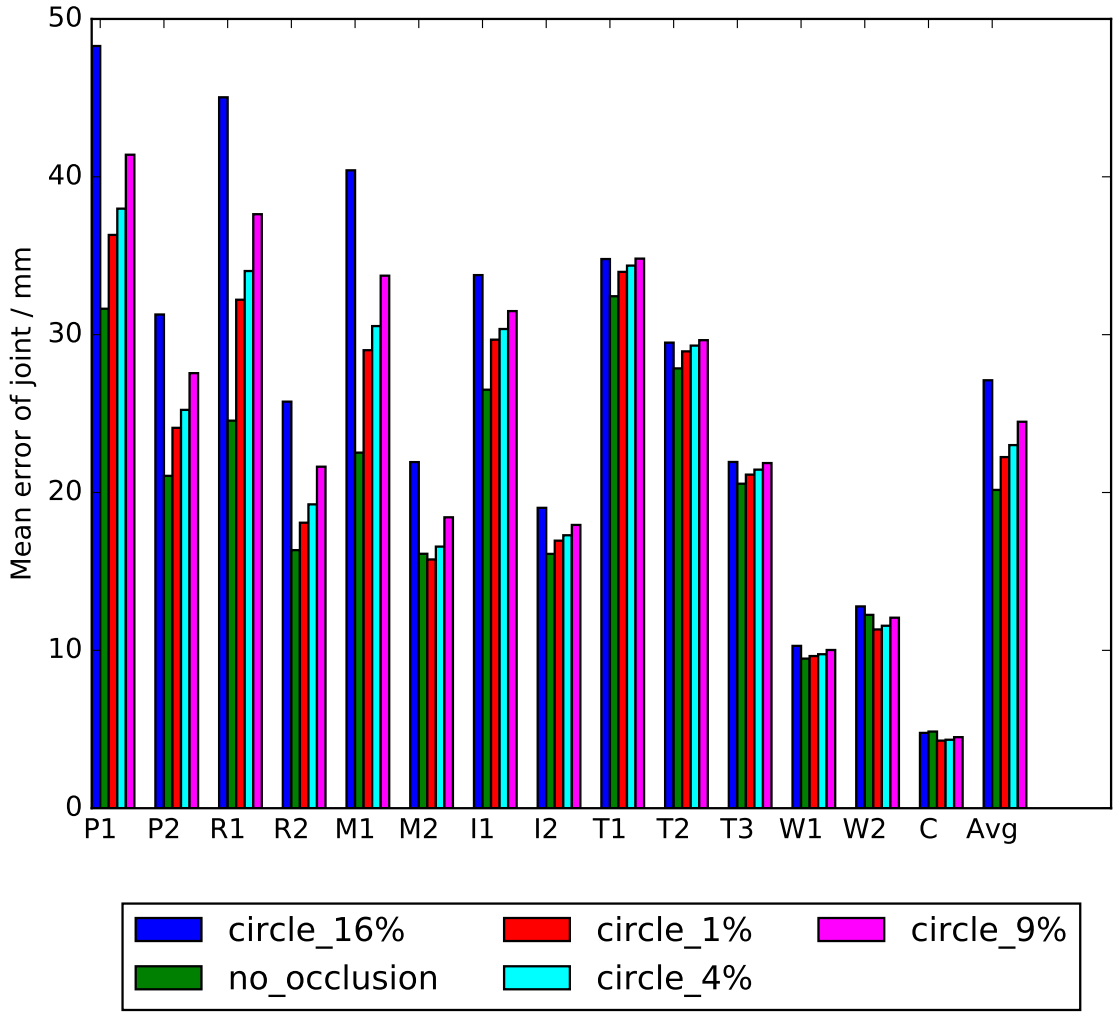


Figure 3.7: Average joint errors in different circular occlusion. It is showed that for each joint with less occlusion is much preciser than that with larger occlusion.

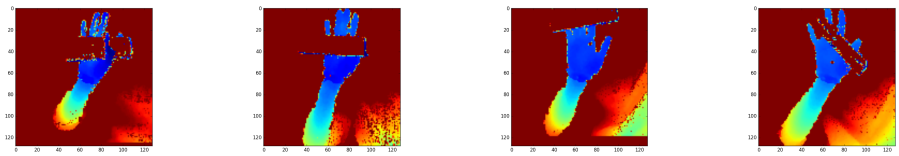
networks are tiny. One possible reason for it is such hand pose is new to both trained neural network. So both of them prefer to predict into the default hand pose and the results are similar. But in some aspects, like the predictions in thumb and little finger, the augmented dataset based trained neural network even performs better than the other.

Another reason is that in reality the users' hand are usually different from the hand in dataset. Because in NYU dataset there is only one person's hand. There are several solutions to this problem, one is to create a new dataset which contains more different persons' hand. Basically it is very difficult because preparing such dataset costs a lot of time. Another solution is to adjust current usr's hand to the hand in the dataset. If the usr's hand is smaller then enlarge it, otherwise decrease the size of the hand. In our experiment we change the size of bounding box to adjust

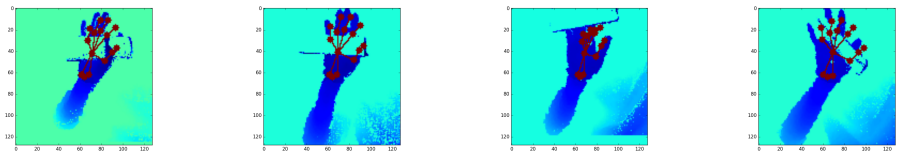
the size of the hand. To some extend it is not a good method to implement the neural network, but it is a good way to verify whether the neural network is possible or not to put into implementation. After adjusting the parameters the results show that the the neural network works but the final predictions are not one hundred percent correct. Apart from some tiny parts in hand the predictions on the whole are correct.

Thirdly, the environment affects the result as well. In real implementation the bounding box selects the hand and other parts of the user and environments at the same time. Such extra information influences the final result because in dataset there are only hand pose information. This problem can be solved in data capture when the environment is clean and the hand is not so close to the users body.

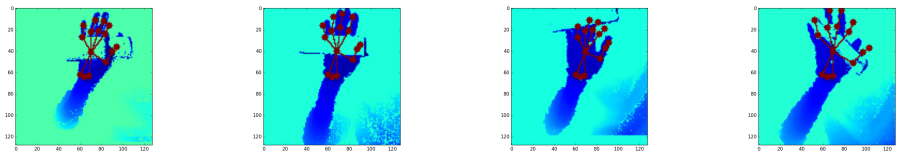
The best solution to all of those problems is to add more data into the training dataset. If all kinds of information is included in the training dataset then the final trained system is much more robust and could deal with all kinds of situations. But data preparing is the key to the learning based method and will cost the most time and further improvement can be realized in this area.



(a) occlusion case 1 (b) occlusion case 2 (c) occlusion case 3 (d) occlusion case 4

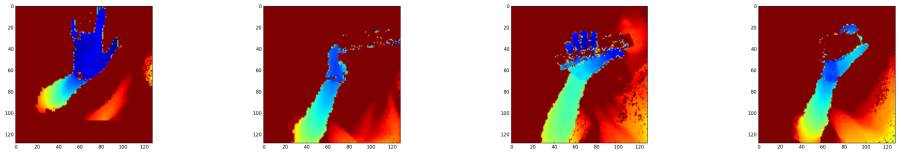


(e) original: 1 (f) original: 2 (g) original: 3 (h) original: 4

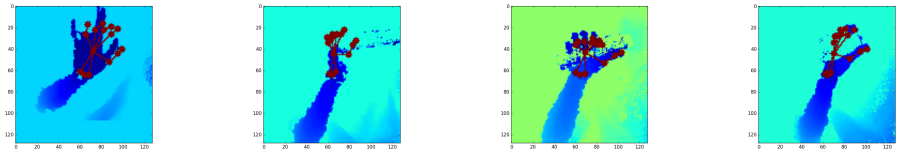


(i) augmented: 1 (j) augmented: 2 (k) augmented: 3 (l) augmented: 4

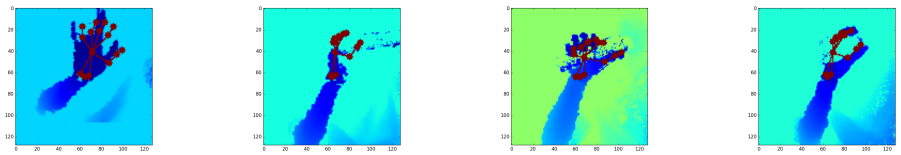
Figure 3.8: the augmented dataset based neural network performs better than the original dataset based neural network.



(a) occlusion case 1 (b) occlusion case 2 (c) occlusion case 3 (d) occlusion case 4



(e) original: 1 (f) original: 2 (g) original: 3 (h) original: 4



(i) augmented: 1 (j) augmented: 2 (k) augmented: 3 (l) augmented: 4

Figure 3.9: both of the neural network perform similar.

Chapter 4

Conclusion

We evaluated the convolutional neural network for hand pose estimation by directly regressing the joint locations. In standard convolutional neural network we added two special layer "Bottleneck" layer and PCA layer, which let the network learn the geometry constraints of hand automatically. Based on original dataset we also created the augmented dataset with occlusions. Then the convolutional neural network is trained on the augmented dataset and the performance of the trained network is evaluated as well. And the experiment results verify the augmented dataset based network performs better than the original dataset based network. Finally the network is implemented in the real hand pose estimation.
Essentially our approach is robust to the random of the hand. With more data the final system can perform better and this is the further step we need to consider, including more users hand and more different hand pose. We hope this work can inspire more works on effective hand pose estimations with discriminative methods.

Bibliography

[AC11] H. Lee. A. Coates, A. Y. Ng. Analysis of Single-Layer Networks in Unsupervised Feature Learning. *A. In Proc. of AISTATS*, 2011.

[CI14] C. Sminchisescu C. Ionescu, J. Carreira. Iterated Second-Order Label Sensitive Pooling for 3D Human Pose Estimation. *Computer Vision and Pattern Recognition*, 2014.

[CQ14] X. Sun C. Qian. Realtime and Robust Hand Tracking from Depth. *Computer Vision and Pattern Recognition*, 2014.

[Hub64] P. J. Huber. Robust Estimation of a Location Parameter. *Annals of Statistics*, 1964.

[IO11] A. A. Argyros I. Oikonomidis, N. Kyriazis. Full DOF Tracking of a Hand Interacting with an Object by Modeling Occlusions and Physical Constraints. *A. In Proc. of AISTATS*, 2011.

[Jon14] Jonathan Tompson, Murphy Stein, Yann Lecun and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (TOG)*, 33(169), 2014.

[JT12] J. Shotton J. Taylor. The Vitruvian Manifold: Inferring Dense Correspondences for One-Shot Human Pose Estimation. *Computer Vision and Pattern Recognition*, 2012.

[MO15] Vincent Lepetit Markus Oberweger, Paul Wohlhart. Hands Deep in Deep Learning for Hand Pose Estimation. *In Proceedings of 20th Computer Vision Winter Workshop*, pages 21–30, 2015.

[SM13] S. Orsten S. Melax, L. Keselman. Dynamics Based 3D Skeletal Hand Tracking. *In Proc. of Graphics Interface Conference*, 2013.

[XZ16] Qingfu Wan Xingyi Zhou. Model-based Deep Hand Pose Estimation. *Computer Vision and Pattern Recognition (cs.CV)*, 2016.