Hand Pose Estimation from Depth Data with Convolutional Neural Networks

Master Thesis

Autor:
Vinagre Ruiz, Manuel

Advisor:
Rincón Zamorano, Mariano

Master in Advanced Artificial Intelligence

College of Computer Engineering
The National Distance Education University (UNED)

September, 2017

Acknowledgements

Firstly, I want to thank my thesis advisor Dr. Mariano Rincón for his patience and given me their support in this thesis. I would also like to thank Dr. Alícia Casals for her unconditional advice and support in the development of this thesis. Also I want to thank Dr. Joan Aranda for his support to my ideas and his contribution with new ones. Our discussions were really interesting.
I would like to express my great gratitude to my family and my love Violette for believing in me and their support during all these years of my studies.

Abstract

The estimation of hand position and orientation, pose, is of special interest in many applications related to Human Robot Interaction, such as human activity recognition, sign language interpretation, or as a human computer interface in virtual reality systems, advanced entertainment games, gesture-driven interfaces, and in teleoperated or in autonomous robotic systems.

This project focusses on the problem of hand pose estimation using convolutional neural networks (CNN) from depth data. Recently, different CNN architectures have been proposed in order to find an efficent and reliable methodolgy to resolve the complexity that involves the variablity in apperance of a hand, with its gestures, changes of orientation, occlusions and so. The use of CNN opens new opportunities for improvements in this research by providing the capability of learning from many samples. This work pretends to advance a step further on the hand pose estimation problem. With this aim, the hand pose estimation using CNN by modifying the output data with a new representation of the hand pose is proposed. An Euclidean Matrix Distance (EDM) is proposed as a hand pose representation. This representation encodes structural information of the hand pose and captures local correlations and dependecies between some hand keypoints.

To evaluate the performance of the proposed method, different CNN architectures using EDM representation are explored and compared with the hand pose representation defined by the position of hand keypoints in the 3D Cartesian coordinate system.

Experimental results show that using EDM representation as target layer in the convolutional network increases the performance of all the proposed architectures in terms of mean square error, both in training and testing sets. As a conclusion, this work shows that EDM representations help to reduce some ambiguities of current hand pose estimation methods using a CNN, by incorporating structural information of the hand and capturing keypoint/joint correlations. This research also gives some insights to investigate in future advances in hand pose estimation using CNN models and will help to explore new strategies for this problem.

Contents

Chapter 1

Introduction

The availability of a reliable hand pose estimation is a high demand for multitude of interactive applications as virtual reality systems, advanced entertainment games, gesture-driven interfaces, and autonomous robotic systems.

Different approaches have been explored for capturing hand pose information. Wearable devices as CyberGlove, 5DT Data Glove Ultra, X-IST Data Glove and P5 Glove offer an occlusion-free hand pose estimation. They rely on exoskeleton devices endowed with sensors and covered with a glove. However they are expensive and unwieldy, constraining the flexibility of hand movements. Lighter approaches based on markers and computer vision have been implemented. They rely on the inference of an approximate pose from the identification of retro-reflective or color patterns attached to the hand, using intensity images extracted from a single or multiple cameras. Despite this approach is lighter than wearable devices, they do not offer enough usability due to the need of attaching marks to the user's hand, their cost is high, and they need multiple cameras to deal with self-occlusions.

Some attempts to estimate hand pose by considering only markerless image information from a single camera have been made. One of the major challenges in these strategies is the non-trivial task of extracting the hand pose from the background. Extensive computer vision research have been carried out over the past years to deal with this problem. Fortunately, new inexpensive camera sensors that provide intensity and depth information based on structured lighting or time-of-flight technology reduce the complexity of background-foreground separation and increase the reliability of the spatial information that can be extracted. This is one of the major reasons why hand pose estimation from intensity and/or depth data is a trending topic in the computer vision community.

Two major complementary paradigms exist in vision-based hand pose estimation: generative or discriminative. In the generative methods a hand model drives the optimization of an objective function to recover the hand pose. The computational load of the optimization process of these methods is high, resulting in low-rate estimation systems, which have to be improved using expensive computation acceleration units. On the other hand, discriminative methods perform a nonlinear multiple regression that maps the observed data to its corresponding hand pose. The evaluation of a regression function is usually much more time efficient than a model based optimization.

Recently, discriminative models based on neural networks with new architectures have emerged behind many state-of-the-art approaches, giving place to the concept of Deep Learning. This new methodology outperforms other computing methods in multiple domains as computer vision. It also reduces the need for feature engineering and it can be implemented from different architectures that can be easily adapted to new problems (transfer learning). Recently, several research works have applied convolutional neural networks (CNN) to solve hand pose estimation. In spite of that, the bound of the optimum solution that can be reached with this kind of algorithms remains unanswered and still has to be largely analyzed. One of the main reasons is the influence of the network architecture and available data on the prediction performance.

Until now, different CNN architectures have been explored to resolve hand pose estimation, as well as the processing of the input data for a better performance. This work opens new research lines as the study of the influence in the performance of modifying target representation from a hand representation space which encodes geometrical properties, which by now remains to be explored.

1.1 Motivation

Our general motivation is to contribute to the research of a reliable solution towards a low cost and markerless hand pose estimation. Concretely, our main goal is to obtain a new approach of hand pose estimation from a single depth map of the hand using Convolutional Neural Networks.

1.2 Problem statement

The problem of the hand pose estimation is formulated as a non-linear regression from a single depth image to a 3D representation of the hand shape or pose. It is important to remark that in this work no temporal information of past or future poses will be taken into account.

Given the following notation:

- Input data $x \in \mathbb{R}^{N \times M}$ is a N-by-M depth map where an element $x_{u,v}$ represents the distance from the reference of the camera to the point of the surface captured by the depth sensor in the (u, v) image position.
- Target data $y \in \mathbb{R}^{K \times 3}$ is a hand pose representation described by K keypoints of the hand. An element $y_i \in \mathbb{R}^3$ is the 3D position of the "i" keypoint referred to the center of mass of the hand $\mathcal{C}_{ref}$.
- A training dataset $\mathcal{D}_{train} = \{x_i, y_i\}_{i=1}^{\mathcal{Q}}$ is a set of $\mathcal{Q}$ correspondence pairs (x, y) used in the learning phase.
- A test dataset $\mathcal{D}_{test} = \{x_i, y_i\}_{i=1}^{\mathcal{R}}$ is a set of $\mathcal{R}$ correspondence pairs (x, y) used in the test phase.
- A CNN regression model is defined as $\mathcal{M}_L$, where L is the layer architecture of the network.
- An objective function $\mathcal{L}(\hat{y}, y)$ where $\hat{y} = \mathcal{M}_L(x)$ defines the quality of $\hat{y} \rightarrow y$ correspondence.

Then, the problem statement is to obtain an adequate CNN regression model $\mathcal{M}_L$ that minimizes $\mathcal{L}(\hat{y}, y)$. The mathematical formulation of this problem is described by the equation 1.1.

$$\underset{\mathcal{M}_L}{\operatorname{argmin}} \sum_{i=1}^{\mathcal{R}} \mathcal{L}(\mathcal{M}_L(x_i), y_i) \tag{1.1}$$

where $(x_i, y_i)_{i=1}^{\mathcal{R}} \in \mathcal{D}_{test}$.

Obviously, the exploration of the complete space to resolve this problem is unreachable. So, the best-of-fit search process depends on the experience of the designer and of previous work results. Furthermore, a set of layer parameters θ_w have to be learned and a set of hyperparameters of the learning process θ_h have to be optimized for a given model. So, the process mathematically formulated in equation 1.2 is performed by the different model proposals.

$$\underset{\theta_w,\theta_h}{\operatorname{argmin}} \sum_{i=1}^{\mathcal{Q}} \mathcal{L}(\mathcal{M}_L(x_i), y_i) \tag{1.2}$$

where $(x_i, y_i)_{i=1}^{\mathcal{Q}} \in \mathcal{D}_{train}$. Again, the exploration of all combinations in 1.2 is intractable and only a set of different hyperparameter configurations is considered.

1.3 Hypothesis

Following our motivation and related works (see chapter 3), our contribution is based on the following hypothesis:

"Representing the target space of the hand pose by a model that encodes hand key-point dependencies improves the performance of the CNN regression problem statement formulated in 1.1."

This is based on the fact that we do not need to explicitly modify our CNN network to model the underlying joint dependencies.

1.4 Objectives

Our overall objective is to justify and validate our hypothesis in section 1.3. So, our main objective is to obtain and evaluate a hand pose representation Γ that captures richer information about pairwise correlations between the position of K hand joints:

$$\Gamma : \mathbb{R}^{K\times 3} \rightarrow V^* \tag{1.3}$$

In order to reach our objective, the following specific objectives have been defined:

- To study current approaches on hand pose estimation from depth data using CNNs.

- To identify a hand pose representation that encodes its geometrical dependencies.
- To select an appropriate dataset for our experimentation.
- To design and select different CNN architectures and evaluate their performance using or not the selected hand pose representation.
- To evaluate and compare the obtained results.

1.5 Methodology

In other to fulfill the above mentioned objectives, the next methodology is presented.

As a first step, previous works of hand pose estimation with CNN from single depth data will be reviewed. Due to the novelty of this topic, the scope of this review will embrace the majority of articles from the most relevant international conferences and journals in computer vision. The contribution of each article will be classified and analyzed. Also, from the same source different methods to encode 3D data structures with underlying 3D information in a plain representation will be explored in order to identify the appropriate hand pose representation. Concretely, this work will only focus on 3D representations that can be used by a CNN learning approach. One or more representations analyzed will be selected as candidate according to a best-of-fit features for the objective of this work. Furthermore, a public hand pose dataset will be obtained from the review of open-access sources. The selection of this dataset will be guided through aspects as quality of images, quantity of information and the number of other approaches that use it.

The second step will consist in the design of the CNN structure. For that, different CNN layers and structures will be analyzed from works founded on the previous step plus articles dealing with convolutional neural network topics. Some of them will be analyzed as candidates of the final architecture of this work. This analysis will comprise the computation of different tests of these designs with the selected dataset, 3D representations as well as the tuning of CNN learning parameters.

The last step will comprise the evaluation of obtained results through the evaluation of the performance of the different experimentation and the comparison with the result of other approaches in the literature.

1.6 Document structure

The structure of this document is as follow. First, the review of related works in hand pose estimation from detph images by using CNN is presented in chapter 2. After that, materials and methods used to perform this study are described in chapter 3. Then, the implementation process of the experimentation part is explained in chapter 4. Results from the experimentation are presented and discussed in chapter 5. Finally, a general conclusion of this work and future work can be shown in chapter 6.

Chapter 2

Literature review

With the emergence of low cost depth sensors([15]), hand pose estimation from depth images has received much attention, having been a topic of study for several years [20]. Nevertheless, this research is still challenging due to the complex and dexterous nature of hand articulations and their multiple degrees of freedom.

Several discriminative methods have been developed with acceptable results to resolve hand pose estimation from depth images. Among them, the random forest and its variants have proved to be reasonably accurate and fast. Following a previous work [17], the effectiveness of the use of random forest to directly regress hand joint angles from depth images is presented in [25]. The authors train a Random Decision Forest classifier to classify pixels into hand parts and infer the 3D hand pose estimation using the mean shift algorithm.

The outstanding performance of convolutional networks in computer vision applications from depth data as [5] has motivated the use of CNN for hand pose estimation. In [21] a CNN is employed for the first time with this purpose. Authors use CNN for feature extraction with the generation of heat-maps of hand joints. In order to map depth images to poses, the 2D locations of joints from heat-maps are predicted. After that, an extra model-based inverse kinematics procedure to recover the entire 3D pose from depth images is required due to the problem of joint occlusions. The entire pipeline is shown in figure 2.1.

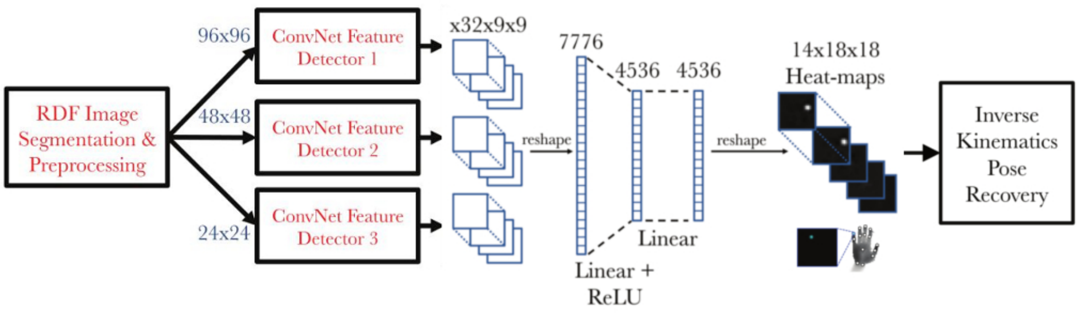
RDF Image Segmentation & Preprocessing
96x96
48x48
24x24
ConvNet Feature Detector 1
ConvNet Feature Detector 2
ConvNet Feature Detector 3
x32x9x9
reshape
7776
4536
4536
Linear + ReLU
Linear
reshape
14x18x18 Heat-maps
Inverse Kinematics Pose Recovery

FIGURE 2.1: Hand pose with heat-maps and inverse kinematics pose recovery pipeline from [21]

In [14] three different CNN network architectures for hand pose estimation by directly regressing 3D joint locations are evaluated. 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, as shown in figure 2.2 (a). The second is a deeper network with three convolutional layers followed by max-pooling layers and two fully-connected hidden layers, as shown in figure 2.2 (b). The last CNN is a multi-scale approach with several down-scaled versions of the input image as inputs to the network. It consists of a convolutional and max-pooling layer for each scale in parallel, and a single fully-connected hidden layer, as shown figure 2.2 (c). The results of this work depicts that the multi-scale approach performs better than a deep architecture, which in turn performs better than the shallow one.

Earliest improvements in hand pose estimation from CNN regression are mainly focused on **the treatment of the input data** that includes data transformation, data refinement or data retrieval. The input **data transformation** has been applied in [24]. In this work, a method to mitigate hand space variability due to global rotation during both the training phase and run-time is proposed. The main idea is that the ambiguity inherent in rotational variant features can be overcome by derotating the hand image to a canonical pose instead of augmenting a dataset with all variations of the rotational degrees of freedom as is commonly done. With this approach, the rotation is learnt using a deep convolutional neural network (CNN) in a regression context to predict full 3 DOF hand orientation of a given depth image. The results demonstrate a significant improvement in the per-frame part hand detection by reducing the variance of the pose space. The result of this algorithm so called DeROT is shown in figure 2.3

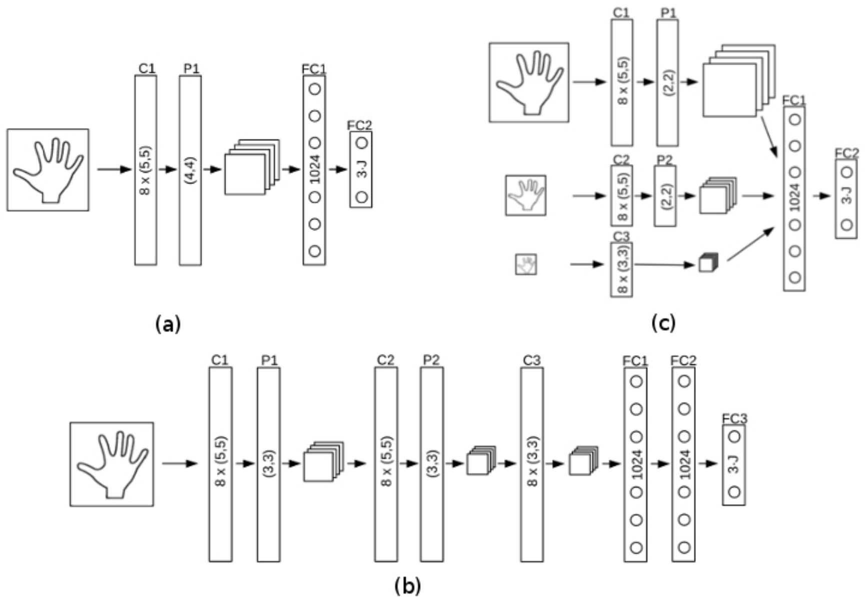

FIGURE 2.2: Different CNN architectures from [14]. (a) Shallow network, (b) deeper network and (c) multi-scale architecture

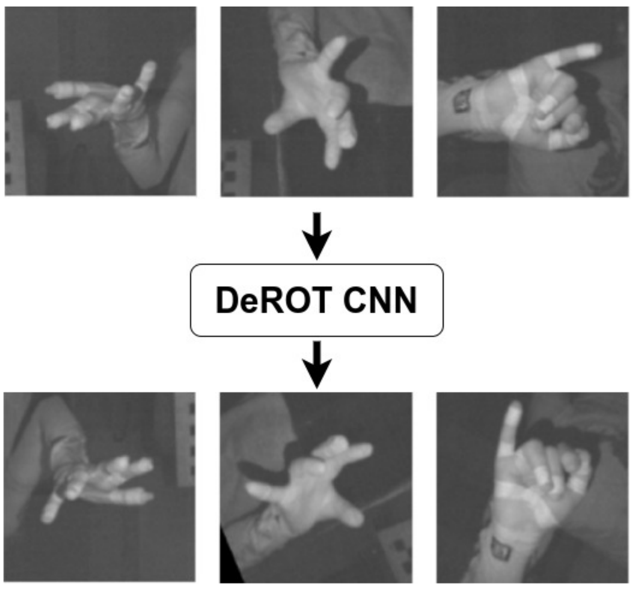

FIGURE 2.3: Input data transformation of the Derotation CNN from [24].

An input **data refinement** by incorporating semantic information from segmentation maps in the original depth images is presented in [13]. To create segmentation maps, this method uses a CNN segmentation learner that combines deep networks and patchwise nearest neighbor search (NN-search). The NN-search is performed in a patch space corresponding to semantic segmentation learned by deep networks. This segmentation maps are combined with raw depth data to perform intermediate representation, which is used by a regression learner to determine joint positions. The regression learner is a CNN with an architecture that resembles an inception unit, where several parallel feature extractors capture information at different levels of localization from the same input convolutional layer. This pipeline is shown in figure 2.4. This study demonstrates that the additional structured information of this representation provides important cues for joint regression which leads to lower errors. They also show how their baseline CNN regression architecture outperforms other architectures, as those used in [14].

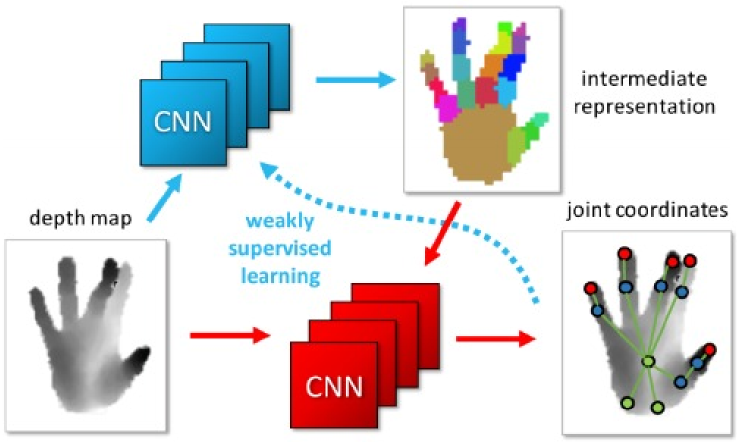

FIGURE 2.4: Input data augmentation with intermediate representation pipeline from [13].

An input **data retrieval** is applied in [9] with the extraction of the projection of the three orthogonal planes of 3D coordinates from depth images, as shown in figure 2.5. Each projection is fed into a separate CNN to generate a set of 2D heat-maps for hand joints following a similar pipeline than in [21]. Experimental results of this approach show that it can better exploit depth cues to recover fully 3D information without model fitting. The results also demonstrate a good performance and generalization that outperforms state-of-art methods, evaluated on public datasets.

Another improvement focussed on **the exploitation of 3D information** of depth images has been explored. From [9], the lost of certain information with the projection from 3D to 2D heat-maps space is highlighted. So, same authors propose in [8] a new type of improvement in hand pose estimation from CNN: the use of CNN architectures that can learn 3D information from depth maps. Concretely, they propose the use of a 3D-CNN architecture that can capture the 3D spatial structure of the input and can accurately regress full 3D hand pose. Due to the fact that the computational complexity increases with 3D-CNNs, the projection of input data to a volumetric representation called Directional Truncated Signed Distance Function (D-TSDF) is used. This transformation permits the increase

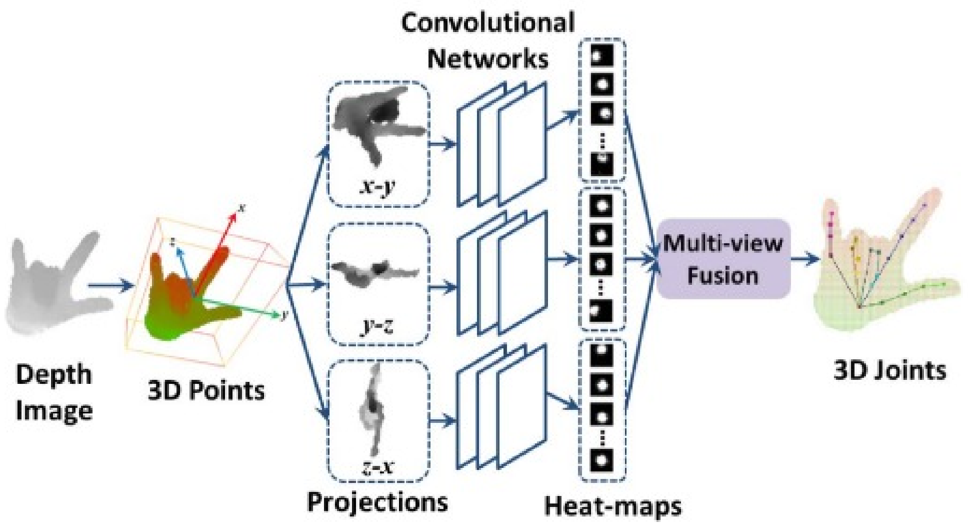

FIGURE 2.5: Input data transformation proposed in [9].

of the regression performance in order to work in real-time. The entire pipeline is shown in figure 2.6. Experimental results of this work show that this approach outperforms actual state-of-the-art methods on two public hand pose datasets, and its implementation is very efficient, running at over 215 fps on a standard computer with a single GPU.

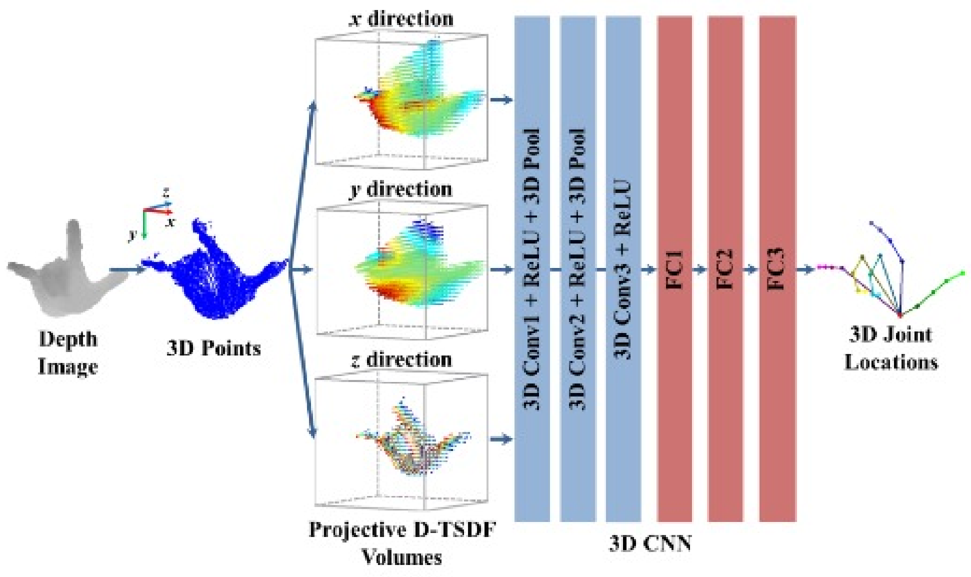

FIGURE 2.6: Input data transformation an pipeline proposed in [8].

The use of a 3D-CNN architecture to exploit the 3D spatial structure of depth data is also used in [6]. In contrast to [8], this approach transforms a depth map into a truncated signed distance function (TSDF) and feeds it into the TSDF refinement network, which removes the artifacts caused by noisy and missing depth. The refined TSDF is then fed into the 3D pose network to estimate the 3D location of each hand skeleton joint in relation to its center of mass. As the input and computation are in 3D, the system learns 3D context for pose estimation, and therefore does not require any post-processing to integrate context in predefined hand model, and thus runs efficiently. The entire pipeline is shown in figure 2.7.

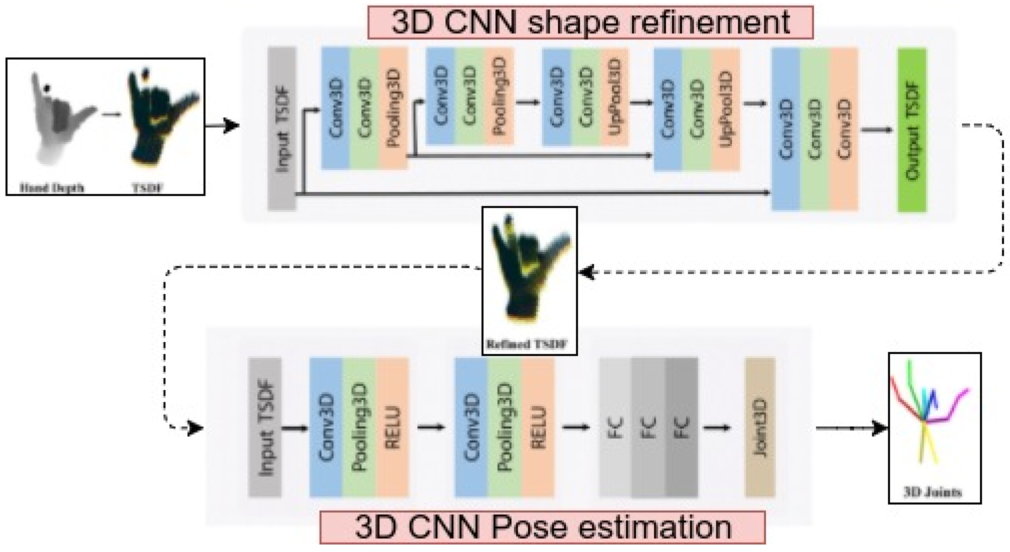

FIGURE 2.7: Input data transformation and pipeline proposed in [6].

Following new architectures to capture 3D information from depth maps, a new approach derived from ensemble methods is proposed in [10]. In this work, a tree-structured Region Ensemble Network (REN) is proposed. In REN, the convolution outputs (feature maps) are partitioned into regions and the results from multiple regressors in each region are integrated in an ensemble feature. This ensemble feature is used with n extra regression layers to infer hand pose, as shown in figure 2.8.

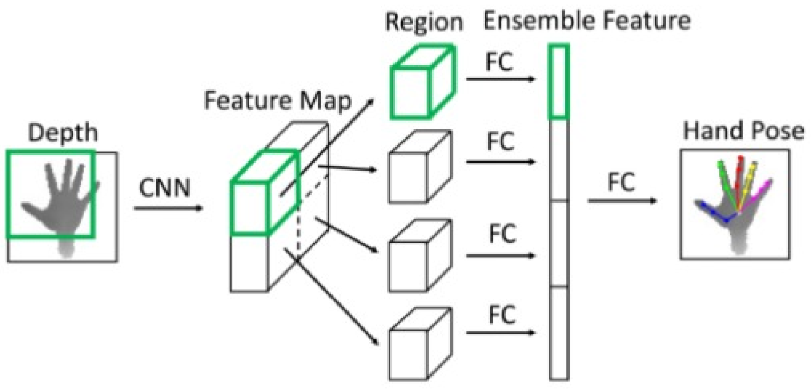

FIGURE 2.8: Input data transformation and pipeline proposed in [10].

Other approaches have been mainly focused on **the exploitation of the prior knowledge** of hand geometry. In [27], a model based deep learning approach that includes a prior kinematics of the hand is presented. For that, an end-to-end learning using the non-linear forward kinematic layer in a CNN similar to [14] is implemented. With this approach, the prior knowledge of hand kinematics is fully exploited with a simple and efficient learning step, avoiding the inconvenient and sub-optimal post processing of validating the resulting hand pose. In figure 2.9, the model based deep hand pose learning is shown.

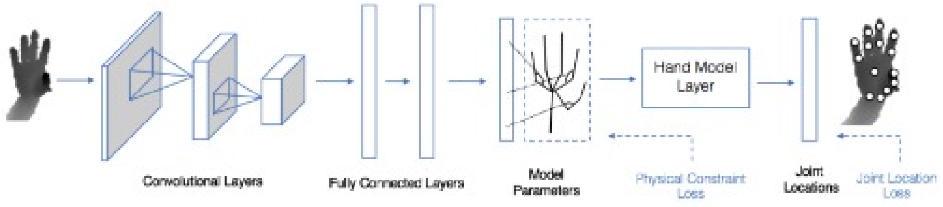

FIGURE 2.9: CNN model based deep hand pose learning proposed in [27].

Finally, another approach that relies on **the change of the target space** is performed in [18]. In this work, the target space is changed to activation features by quantifying the joint angles, which are used in a CNN classification task. After that, activation features are used in a collaborative spatio-temporal manner to estimate pose parameters using efficient nearest neighbor search and a matrix completion model.

Chapter 3

Methodology Design

This chapter contains material and methods to carry out the objectives of this work. First, the dataset and the baseline hand representation are presented. After that, a new hand representation by encoding geometrical dependencies of baseline keypoints is detailed. Finally, a method to perform an adequate experimentation of our CNN designs is explained.

3.1 Dataset

In order to achieve a good performance, considering the wide variety of configurations and perspectives in which a hand can be visualized, the estimation of its pose using convolutional neural networks requires a training with a large data set. In addition, a public dataset is preferred because it allows us to compare our performance with respect to previous works, as well as to reproduce them.

In this work, the NYU Hand Pose Dataset (NYU) is used for our experimental part. This dataset is presented in [21] and used to train a CNN for a continuous pose recovery of human hands. This dataset consists of images with depth and color data generated using three RGBD cameras separated by approximately 45 degrees surrounding the user from the front. It also contains synthetic depth data generated by a public library called *LibHand*. This data is a noise and artifact free version of the real depth data.

The data is annotated with the ground truth of each hand pose obtained by a model-based hand tracking and Particle Swarm Optimization. These annotations contain image position (u, v, d) of 36 key-points of the hand, including each joint of the hand.

Images in NYU dataset are separated into a training and test set. Both sets include three classes of images: depth, color and synthetic images. The training set includes 72757 frames for each class. So, a total of 654813 images (72757 frames $\times$ 3 classes $\times$ 3 camera views) are supplied. The testing set contains 8252 frames per class with a total of 74268 images (8252 frames $\times$ 3 classes $\times$ 3 camera views). Samples from training set are from a single user and samples from test set are from two different subjects. An example of one sample of this dataset can be seen in figure 3.1

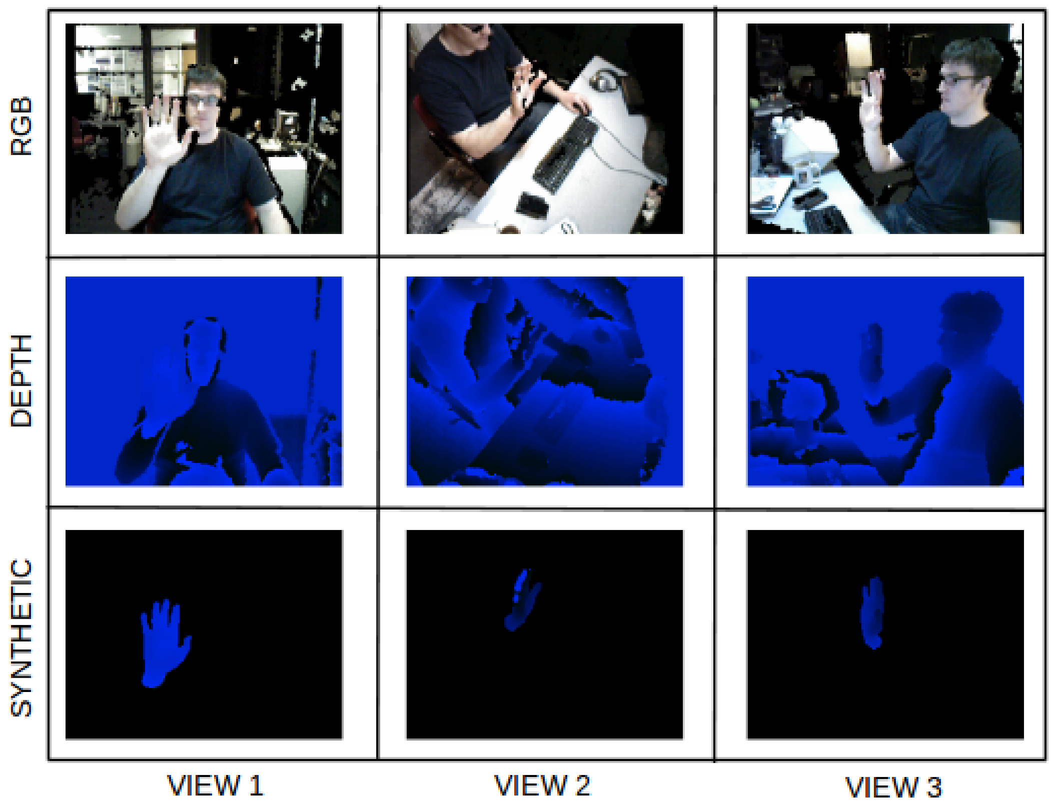

FIGURE 3.1: Sample of NYU Hand dataset. Each sample contains RGB, depth and synthetic images from three different views.

3.2 Hand representation baseline

The data annotation of the NYU hand dataset is a set of hand keypoints that defines a particular pose of the hand. For comparison purposes, the set of keypoints of our baseline pose representation is the same as in [21, 14, 27]. This representation consists in 14 key-points and its configuration is shown in figure 3.2. In this figure, 'T', 'I', 'M', 'R', 'P' denote 'Thumb', 'Index', 'Middle', 'Ring', 'Pinky' fingers and 'TIP', 'DIP' are 'tip finger', 'proximal interphalangeal' respectively.

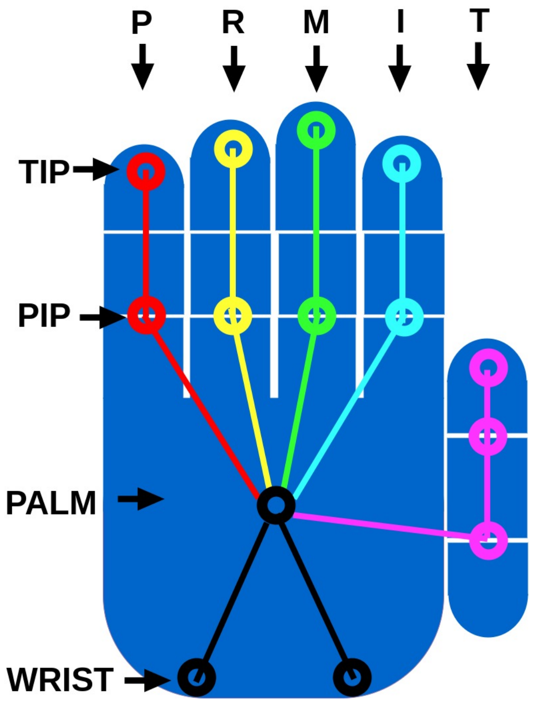

FIGURE 3.2: Hand representation baseline

Each keypoint j_i from keypoint set $J = \{j_i\}_{i=1}^{14}$ is annotated with its 3D coordinates $j_i = (x_i, y_i, z_i)$. These coordinates are described in millimeters and they are referenced from the center of mass of the hand.

3.3 Geometrical hand representation

Representing a hand pose by encoding geometrical relations between hand keypoints have been explored in our previous work [23], where a geometrical representation by encoding correspondences between pose keypoints is presented. These correspondences are computed by the areas of the triangles formed by three keypoint relations. Furthermore, different geometrical features from 3D coordinates of keypoints were explored as point-to-point, point-to-line and point-to-plane geometrical relations. These relations were tested and evaluated for human action recognition purposes.

However, the majority of these approaches can not be applied for our regression problem statement due to the difficulty to go back from geometrical features to the original 3D coordinate space of baseline keypoints.

In spite of that, geometrical relations of 3D points through pairwise distances can be applied with the Euclidean Distance Matrix (EDM). The process of extracting the original keypoint configuration from EDM has been largely explored for the development of a number of tools as multidimensional scaling (MDS)[16], reconstruction of the 3D shape of molecules [26] or sensor network localization [7].

In addition, one of the principal motivations for the use of EDM is the good results presented in [12], which uses the EDM representation for the 3D pose retrieval from intensity images using deep learning.

Consider a collection of n points in a 3-dimensional Euclidean space $X \in \mathbb{R}^{3\times n}$, $X = [x_1, x_2, ..., x_n], x_i \in \mathbb{R}^3$. Then the squared distance between x_i and x_j is given as equation 3.1, where $\|.\|$ denotes the Euclidean norm.

$$d_{ij} = \|x_i - x_j\|^2 \tag{3.1}$$

The euclidean distance matrix is an exhaustive table of distance-square d_{ij} between pairs from a list of n points. Each point is labelled ordinally, hence the row or column index of an EDM, i, j individually addresses all the points in the list.

Consider the following example of an EDM for the case $N = 3$:

$$D = [d_{ij}] = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} \tag{3.2}$$

Matrix D has N^2 entries but only $N(N1)/2$ pieces of information.

The EDM representation has interesting features over vector representations that are suited for our problem:

- **Encode structural information of the pose**. This representation provides implicit model constraints through distance information.

- **Capture correlations**. EDM captures pairwise correlations and dependencies between all keypoints.
- **Rotation and translation invariance**. The EDM representation is invariant to in-plane image rotations and translations.

3.4 Network architecture

The performance of convolutional neural networks is critically sensitive to the architecture design. Determining a proper architecture is a challenge because many structural hyperparameters are involved and it is not well understood how these hyperparameters interact with each other to influence the accuracy of the resulting model [11]. There is no mathematical formulation for calculating the appropriate hyperparameters for a given dataset. These hyperparameters include number of convolutional and fully-connected layers, location and size of pooling layers, the number of filters, stride movement, and size of fully-connected layers.

In this work, a similar architecture of Deep-Prior approach presented in [14] has been considered as a baseline for our design. This architecture has also been used as a baseline in [27]. This architecture consists of three convolutional layers followed by max-pooling layers and three fully-connected layers. The last fully-connected layer is the regression layer. All convolutional layers use Rectified Linear Unit(RELU) activation functions. The baseline architecture is shown in figure 3.3.

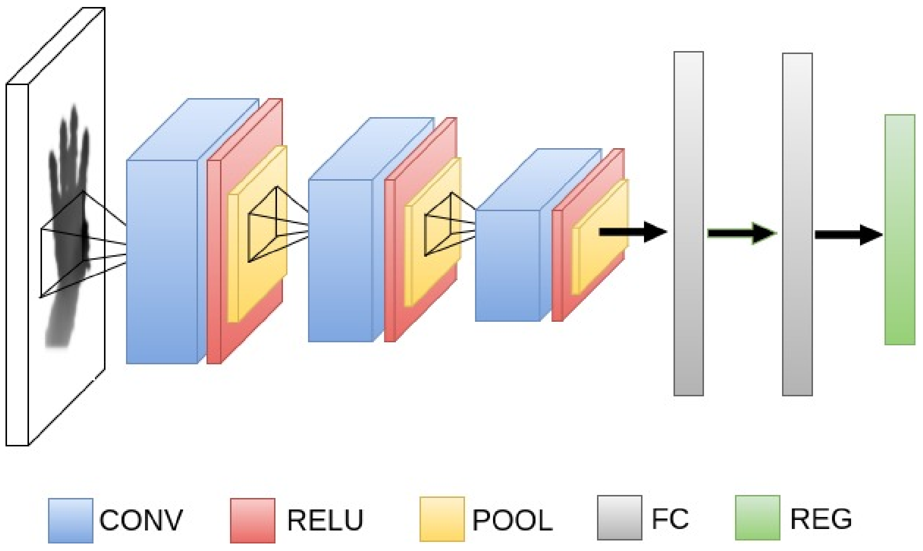

FIGURE 3.3: CNN architecture baseline

From the baseline architecture, different hyperparameters related to the network architecture that have been explored are:

- **Number of convolutional layers**. With this hyperparameter, the deep of the network has been tested. From a configuration baseline with three layers, other options with more convolutional layers have bee tried.
- **Number of fully connected layers**. The possibility of different non-linear combinations have been explored with a variation of the number of FC layers. Some variations with only one layer have been added.

From these hyperparameters, the set of architectures explored in this work in order to get an appropriate CNN model is presented in table 3.1. Three different design patterns of CNN architectures for each output target (3D Pose and EDM) are considered. The first design is the CNN architecture baseline. The second network has six convolutional layers and only one fully connected. A the third network has six convolutional layers with two fully connected layers.

All nets take an 128×128 depth image as input and have three pooling layers. Fully-connected layers have 4096 hidden units as baseline network in [27]. Similarly to [14], CNN configurations stack 3×3 convolutional kernels with zero-padding 1 and stride of 1 such that convolutions preserve the spatial extent of feature maps.

In order to prevent overfitting, the dropout regularization technique has been used ([19]. At each training iteration, a dropout layer randomly removes some nodes in the network along with all their incoming and outgoing connections. Dropout can be viewed as a form of averaging multiple models ("ensemble"), technique which shows better performance in most machine learning tasks. Dropout can be applied to hidden or input layers. In this work, a dropout layer has been included after each fully-connected layer.

CNN ARCHITECTURES					
PoseNet-1	PoseNet-2	PoseNet-3	GeoNet-1	GeoNet-2	GeoNet-3
Image-Input					
Conv1	Conv1	Conv1	Conv1	Conv1	Conv1
	Conv2	Conv2		Conv2	Conv2
Max-pooling					
Conv2	Conv3	Conv3	Conv2	Conv3	Conv3
	Conv4	Conv4		Conv4	Conv4
Max-pooling					
Conv3	Conv5	Conv5	Conv3	Conv5	Conv5
	Conv6	Conv6		Conv6	Conv6
Max-pooling					
FC1	FC1	FC1	FC1	FC1	FC1
FC2		FC2	FC2		FC2
Baseline-Output			Geometrical-Output		

Table 3.1: CNN architectures considered in this work. We denote convolutional layers with "convi" which denotes the i-th convolutional layer. We denote pooling layers with "max-pooling". Fully-connected layers are denoted with FC.

Chapter 4

Implementation

This chapter describes the implementation steps followed to achieve the methodology described in chapter 3. First, the data processing applied to the dataset to perform our experimentation is described. Secondly, the procedure to obtain the Euclidean Distance Matrix from baseline as well as the backward procedure are explained. Thirdly, the procedure to train our CNN models with different hyper-parameters is shown. Finally, the evaluation method of the trained models is presented.

4.1 Data Preprocessing

The data from the NYU hand dataset has to be preprocessed in order to meet our requirements for hand pose estimation. Depth images of this dataset samples the whole human body and the background, as can be seen in figure 4.1. Commonly, a hand detection step is required for a hand pose estimation "in the wild". In this work the detection process is already assumed since NYU dataset provides us with the position of the center of the hand.

The hand data preprocessing is similar as in previous works [14, 27]. A fixed-size 3D cube patch centered on the provided center of the hand is used to extract hand information from the whole depth data. This patch is converted from 3D real world coordinates to depth image coordinates, assuming that depth values are z-coordinate values of the real world space. Assuming the pinhole camera model, the conversion from real world to image coordinates is given by formula 4.1. The

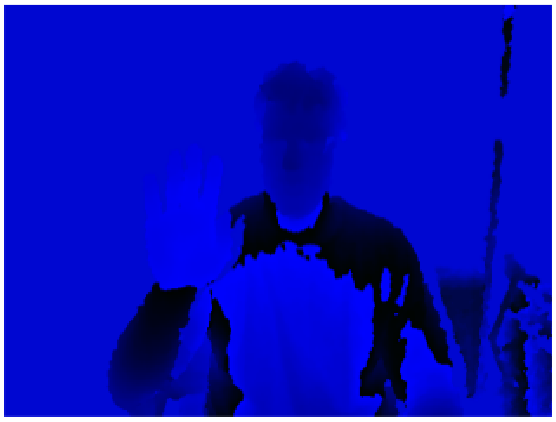

Figure 4.1: Depth image of the dataset in the RGB channel

inverse transformation is derived from the same relation. The result of this process is shown in figure 4.2.

$$\begin{bmatrix} u \\ v \\ d \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.1}$$

where $f = (f_x, f_y)$ is the camera focal length and $c = (c_x, c_y)$ is the optical center.

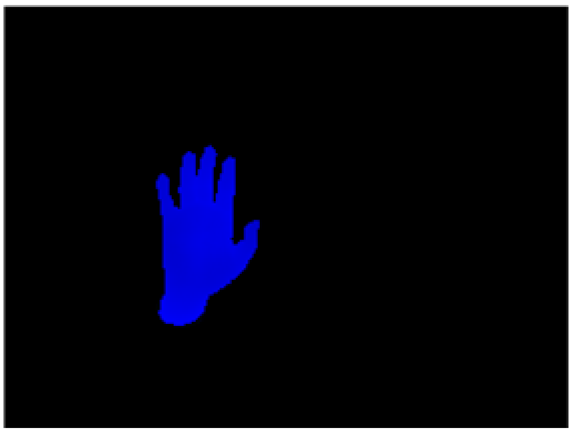

Figure 4.2: Depth hand data extracted from 3D cube patchin RGB channel

Depth images store the most significant bits of the 12 bits depth data in the green channel and the least significant bits in the blue channel. In order to obtain the complete depth information, a rectification step is performed by bit shifting the green channel to the left 8 bits and adding it to the blue channel. As a result, integers of depth values in millimeters are given. For convenience, units are transformed to centimeters.

Since depth and ground truth of hand pose keypoints in the dataset are a set of absolute positions in the whole image, the transform of them into relative positions with respect to the center of the hand is needed. This transformation is indicated below:

$$\begin{bmatrix} x'_j \\ y'_j \\ z'_j \end{bmatrix} = \begin{bmatrix} 1/k & 0 & 0 \\ 0 & 1/k & 0 \\ 0 & 0 & 1/k \end{bmatrix} \begin{bmatrix} x_j - x_c \\ y_j - y_c \\ d_j - d_c \end{bmatrix}$$

where x_j, y_j, z_j and x'_j, y'_j, z'_j are denoted as original position and normalized position of the point j, respectively. Furthermore, x_c, y_c, z_c is the absolute position of hand's center and k is the length of the half of the cube of the hand's bounding box. As a result, each point position is normalized in the range of [1,1] with respect to the center of the hand and bounding box size.

From the centering and normalizing process, depth images of 128x128 size are obtained, scaling ground truth key-points accordingly. Finally, the extraction of the subset of hand pose keypoints based on our baseline hand pose representation is performed. A depth image and keypoints of the entire process is shown in figure 4.3.

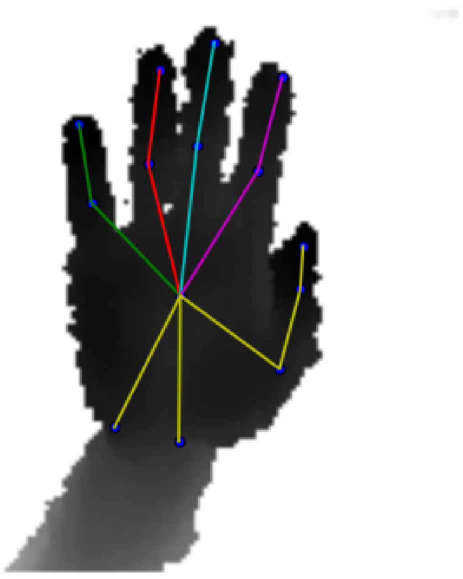

FIGURE 4.3: Depth image and keypoints used as data

4.2 Euclidean Distance Matrix

There are two principal EDM-related tasks: a forward and inverse tasks. The forward task is to compute the EDM from a set of points X in order to obtain our new target representation of the CNN. This is a simple task implemented by applying equation 3.1 to determine D from the baseline representation 3.2. The inverse task is to reconstruct the original point set from EDM. Given a subset of all the pair-wise distances between a set of points in a fixed dimension, the question is: can we estimate the relative positions of all the unknown points accurately? This problem involves solving a hard non-convex optimization problem that can be formulated as the following error minimization problem:

$$\underset{x_1,\dots,x_n}{\operatorname{argmin}} \sum_{p,q} |d_{p,q} - E\hat{D}M(X)_{p,q}| \tag{4.2}$$

where $E\hat{D}M(X)_{p,q}$ is the noisy matrix estimated by the CNN from an image input. In this work, solvable Semidefinite Programming relaxation of 4.2 has been applied, as in [12]. For that, the SDPT3 Semidefinite programming solver [22] has been used.

4.3 Training process

The training process of the convolutional neural network is implemented through a backpropagation algorithm. In this process, the model adjusts its filter weights in order to minimize the amount of prediction error. The weight adjustments are driven by a gradient descent optimization algorithm. There are three variants of gradient descent, which differ in how much data are used to compute the gradient of the objective function:

- **Batch gradient descent**.The gradients are calculated for the whole dataset to perform just one update. The decreased update frequency results in a more stable gradient error. This algorithm can be very slow and is intractable for datasets that do not fit in memory.
- **Stochastic gradient descent**. The gradients are calculated for each training example. It is therefore usually much faster and can also be used to learn

online. However, the stochastic gradient descent (SGD) performs frequent updates with a high variance that causes the objective function to fluctuate heavily and this complicates the convergence to the exact minimum. However, it has been shown that when we slowly decrease the learning rate, SGD shows the same convergence behaviour as batch gradient descent.

- **Mini-batch gradient descent**. The gradients are calculated for every mini-batch of n training examples. This variant reduces the variance of the parameter updates, which can lead to more stable convergence. It seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning. To calculate mini-batch gradient updates, SGD usually is employed.

In this work, the training process is driven by a **mini-batch with stochastic gradient descent algorithm with momentum**. In this process, next hyper-parameters have to be considered:

- **Learning Rate**. This term determines the step size of the weight updates in the direction of the gradient. A learning rate that is too small leads to slow convergence, while a learning rate that is too large can hinder convergence and cause the loss function to fluctuate around the minimum or even diverge. Also, adapting the learning rate for this procedure can increase performance and reduce the training time. This is called learning rate annealing or adaptive learning rates.

- **Momentum term**. Basically, the momentum term increases updates for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients directions diverge. As a result, we gain faster convergence and reduced oscillation.

- **Number of epochs**. This term determines how many times the algorithm is going to run over the entire training set. The number of epochs affects directly (or not) the result of the training step (with just a few epochs you can reach only a local minimum, but with more epochs, you can reach a global minimum or at least a better local minimum). Common number of epochs ranges between 50 and 100.

- **Batch size**. This term defines the number of samples that has to be propagated through the network. Common mini-batch sizes range between 50 and 256, but this number can vary for different applications.

In our experimentation, each CNN model has been trained with 100 epochs and a batch size of 256 training samples per iteration. On each mini-batch step, samples are shuffled. An adaptive learning rate by a step decay is used. In this strategy, the learning rate is reduced in function of the number of epochs. In our case, the decay is performed with a factor of 0.25 every time the algorithm achieves 25% of epochs.

The values of the learning rate, momentum and dropout probabilities are sensitive hyperparameters that affect the optimization as well as the generalization of the model. Different values for these hyperparameters can lead to overfitting, underfitting or divergence issues. So, an optimization process is required in order to see which values provide the highest performance of the network. In order to choose a set of optimal values of these hyperparameters, the common hyperparameter optimization methods are:

- **Manual exploration**. This method selects the best configuration among a set that is design manually through a large number of choices.
- **Exhaustive grid search**. In this method, some sets of values are defined for each hyperparameter and the cartesian product between all sets is computed. So, all possible combinations of hyperparameter values between the values of the sets are tested. This method is very time-consuming because it increases exponentially with the number of hyperparameters.
- **Random search**. This method proposes to sample independently each hyperparameter from a different distribution. The number of experiments is defined and different values of hyperparameters are sampled from the hyperparameter distributions for each experiment. In [2] the authors show that this method searches effectively the hyperparameter space and requires less computation time than the exhaustive grid search method.

Since the number of hyperparameters in CNN becomes big, grid search becomes so inefficient. Usually, random search is applied for hyperparameter optimization in CNN models.

In this work, the experimentation has been performed in a pipeline of two steps:

- **Architecture selection**. In this step, the network architectures described in table 3.1 are evaluated. Due to the number of models to evaluate and the time-computing limitation, in this step a manual exploration has been selected. Several experiments to evaluate the best performing architecture have been carried out. For that, values from related works [14, 27] have been taken as initial reference of the possible range for each hyperparameter.
- **Representation performance**. In this step, the best performing architectures for each output representation from the previous step is fine tuned and compared. For that, a random search to select good settings for the learning rate, the momentum term and dropout probability has been applied. A cross validation to find an appropriate set of values is applied. This method runs multiple hyperparameter settings and selects the best one based on its performance on the validation set.

For the random search scheme, the Hyperopt framework presented in [3] has been used. To run hyperopt, an objective function, the parameter space, the number of experiments to run and optionally set a constructor to keep the experiments have to be defined. There are both continuous and categorical methods to describe the parameters. The function to minimize takes hyperparameter values as input and returns a score, that is a value for error to minimize. This means that each time the optimizer decides on which parameter values it likes to check, it trains the model and predicts targets for a validation set (or do cross-validation). Then, the prediction error is computed and the algorithm selects which values to check and the cycle starts over. Hyperopt offers four distribution options: uniform, normal, log-uniform and log-normal.

Following the indications described in [2], the parameters of the random search process have been established. The algorithm samples 50 different hyperparameter combinations. For the sample distribution for the learning rate and momentum, a uniform distribution has been defined. For the dropout probability, a uniform distribution in the linear scale has been established. The initial ranges of these distributions have been obtained from the previous experimentation step. Concretely, a range between 0.001 and 0.01 for learning rate search, a range between

0.0 and 1.0 for the dropout probability and a range between 0.75 to 1.0 for the momentum.

For cross validation, the training set is splited in training and validation sets. The CNN models are trained with the training set and the selection of the best hyperparameters from random search is based on the performance on the validation set. A 3-fold cross validation is performed. So, the validation set size is the 25% of the training set size and (75%) is selected as training set. Data is sampled from a uniform distribution.

4.4 Implementation tools

This work has been implemented in Python language using Tensorflow [1]. TensorFlow is a software library for numerical computation of mathematical expressional, using data flow graphs. Nodes in the graph represent mathematical operations, while the edges represent the multidimensional data arrays (tensors) that flow between them. It was created by Google and tailored for Machine Learning. In fact, it is being widely used to develop solutions with Deep Learning.

Tensorflow have been used from a high-level neural networks API called Keras [4]. This API is written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

The experimentations have been running on an Intel Xeon equipped with an NVIDIA Titan X Pascal GPU, driven by 3584 NVIDIA CUDA cores running at 1.5GHz with 12 GB of GDDR5X memory. TITAN X packs 11 TFLOPs of brute force.

4.5 Evaluation

Two evaluation metrics have been established. The first is the average Euclidean distance between the ground truth joint location and the predicted joint location over samples of the test set, for each join. This metric is also called per-joint error averaged.

The second evaluation metric measures the fraction of test set samples for which each predicted joint is below a maximum Euclidean distance between the ground truth and the predicted joint locations. This metric is called success-rate.

Chapter 5

Experimental Results

This chapter presents the experimental results of this work. A self-comparison between different convolutional neural networks is analyzed. Also, a comparison with related state-of-art methods is presented.

5.1 Comparison between CNN architectures

For a self-comparison between the CNN architectures introduced in 3.1, results from the first experimentation step described in section 4.3 are analyzed. The experimental results of this part are separated into two main groups: results from PoseNet and results from GeoNet networks. PoseNet architectures have an output layer aimed to perform a regression process from depth images to XYZ hand keypoints, while GeoNet nets have an output layer designed to perform a regression from depth images to EDM representations. The performance of each network is measured by the Mean Square Error (MSE) between the predicted joint positions and the ground truth in the training and validation processes respectively.

In table 5.1, MSE results of the best performance from each PoseNet architecture are presented. These results indicate a better performance of the PoseNet3 architecture than PoseNet2. PoseNet2 and PoseNet3 have the same architecture, but PoseNet2 has only one fully connected layer. This implies that additional layer lets the network learn more sophisticated combinations of features that help to achieve better performance. The significant difference between MSE values of PoseNet2 indicates the lost of expressiveness of the network. This point out that our problem need a minimum of two fully-connected layers to learn non-linear combinations of the obtained features in order to minimize the prediction error. Also, the great difference between training and validation MSE indicates a lost of generalization.

The best combination of hyperparameters of PoseNet3 was 0.02 for learning rate, 0.9 for the momentum and 0.1 for dropout probability.

Architecture	Training Error	Validation Error
PoseNet1	0.00359	0.00435
PoseNet2	0.00478	0.00641
PoseNet3	0.00268	0.00331

TABLE 5.1: Mean square error of PoseNet architectures over the training and validation sets.

The Mean Square Error between the predicted joint positions and the ground truth in the training and validation steps of each GeoNet network is presented in table 5.2. Results indicate the same effect related to the depth size and the number of fully connected layers. So, the third network GeoNet3 is also the better network over the rest of architectures. Also, GeoNet2 results are worse than the rest of GeoNet networks. The best combination of hyperparameters of PoseNet3 was 0.01 for learning rate, 0.9 for the momentum and 0.05 for dropout probability.

Architecture	Training Error	Validation Error
GeoNet1	0.00329	0.00405
GeoNet2	0.00448	0.00611
GeoNet3	0.00198	0.00246

TABLE 5.2: Mean square error of GeoNet architectures over the training and validation sets.

An extra experimentation by adding one pooling layer after each convolutional layer has been added. Results demonstrates a degradation of the performance in training and validation test sets. This result depicts an information loss due to an excessive down-sampling of feature maps.

From table 5.1 and 5.2 a preliminary comparison about the two output representations indicates a better performance of EDM representation in terms of MSE error. However, the performance is very sensitive to hyperparameter values and in the first part of the experimentation, these values have been find by manual exploration. In order to perform a more accurate comparison about target representation, PoseNet3 and GeoNet3 have been fine tuned in the next experimentation step described in section 4.3. This experimentation optimizes hyperparameter values of PoseNet3 and GeoNet3 with a random search with 3-fold cross validation. As a result, best hyperparameters settings for PoseNet3 are 0.0093 for the learning rate, 0.97 for the momentum and 0.04 for the dropout probability. For GeoNet3 best hyperparameter settings are 0.009 for learning rate, 0.98 for the momentum and 0.025 for the dropout probability.

From the results of the second part of the experimentation, an evaluation with metrics described in section 4.5 has been performed. Results of this evaluation are presented in figure 5.1 and 5.2. These results have been performed from optimized models PoseNet3 and GeoNet3 and trained models from the first experimental step.

Figure 5.1 illustrates the success-rate, where the horizontal axis represents the distance threshold in millimeters and the vertical axis represents the fraction of frames (%) for which the squared error for each keypoint is below the distance threshold.

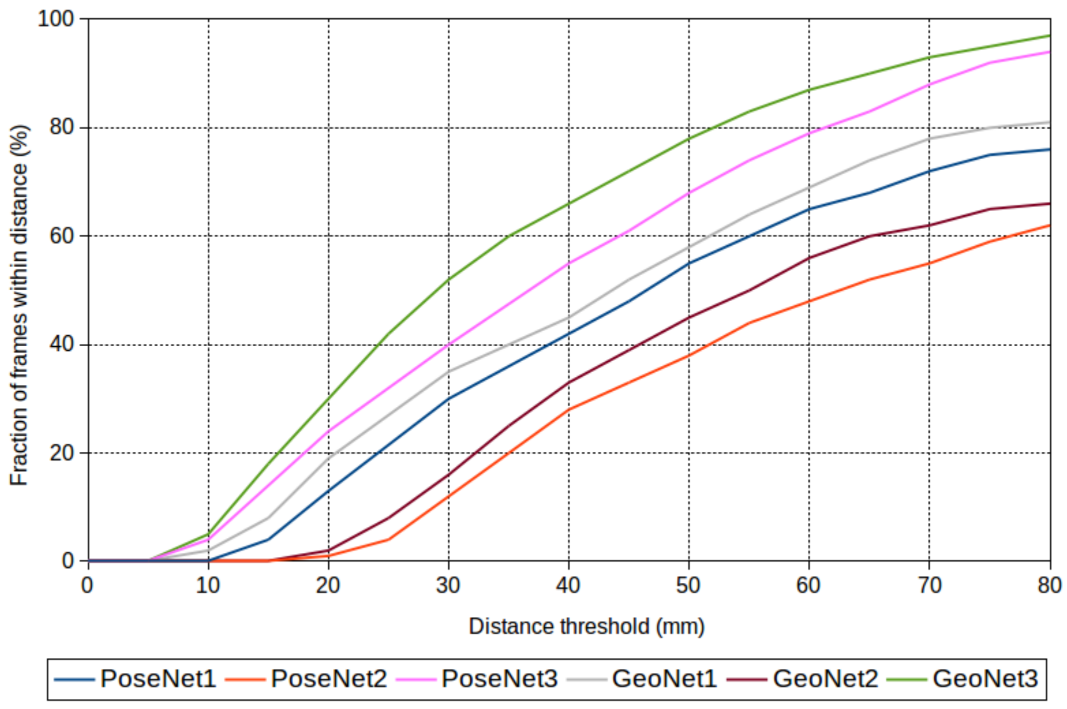

FIGURE 5.1: Success Rate of the proposed CNN architectures.

Figure 5.2 illustrates the mean joint error where the horizontal axis represents the different keypoints, and the vertical axis represents the mean error per keypoint across the test set. The rightmost bars show the mean error per keypoint averaged over all keypoints. The description for keypoints abbreviations are: pinky (P), ring (R), middle (M), index (I), thumb (T), wrist (W), palm center (C). For all fingers, the indices 1 and 2 refer to the tip finger and proximal interphalangeal respectively. The index 3 refers to the lower joint, only used in the thumb finger. For the wrist, the indices 1 and 2 refer to the left and right wrist position respectively.

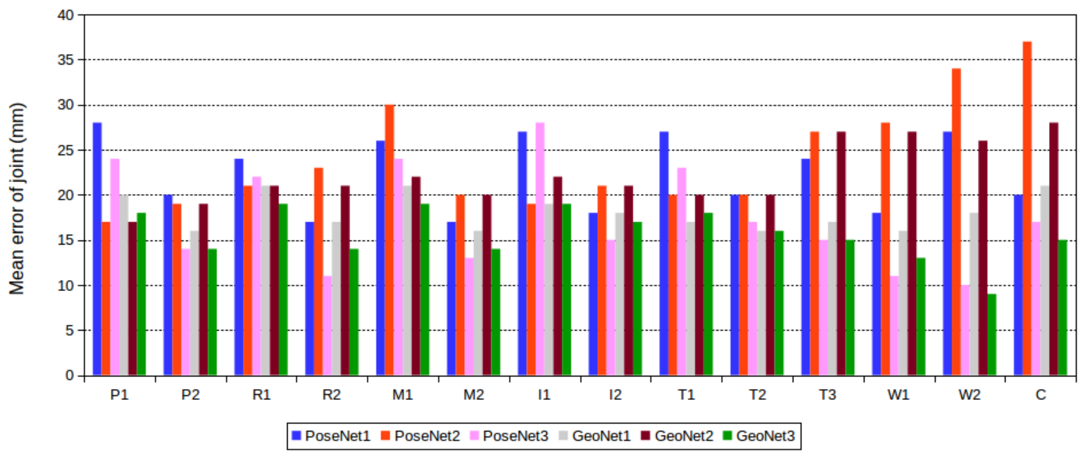

FIGURE 5.2: Mean Joint Error of the proposed CNN architectures.

As shown in figure 5.1, the success rate of GeoNet3 is higher with respect to the PoseNet3 network. In figure 5.2, GeoNet3 has a lower mean joint error averaged over all joints as well as lower error in each joint separately than PoseNet3.

Despite the slight variation between results of GeoNet3 and PoseNet3, the EDM target representation of GeoNet3 seems to be better to create correspondences between feature maps and Euclidean distances than XYZ positions of hand keypoints. This result confirms that target representations with geometrical correspondences help to reduce the ambiguity of the pose estimation problem by incorporating structural information of the hand and capturing keypoint/joint correlations.

5.2 Comparison with state-of-art

In this section, the GeoNet3 network is compared with three state-of-the-art methods applied on the same NYU hand pose dataset. The first method is the CNN proposed in [21]. The second method is the CNN baseline described in [14]. The third method is the CNN proposed in [27].

Tompson et al. [21] use a convolutional network with 2 convolutional layers to infer 2D joint positions. Oberweger et al. [14] incorporate a convolutional network with three convolutional layers to infer 3D joint positions by introducing the pose prior layer. Zhou et al. [27] use the same network as in [14] but a hand model layer is added.

As shown in figure 5.3, the success rate of our approach outperforms [21] and [14] and achieves approximately similar accuracy with [27]. The average error over all joints in figure 5.4 shows a good performance over all approaches, but our proposed architecture GeoNet3 outperforms these approaches both in terms of average error over all joints and in mean error per joint. Despite the efforts of these approaches to improve the accuracy of the predictions by involving extra steps, they are not sufficient to learn good mappings from input images to 3D hand poses. Their limited performance is mainly due to the depth of their architectures. Our approach takes two main advantages over the other approaches: a large convolutional network architecture and a good generalization performance thanks to the benefits of mapping to the EDM space.

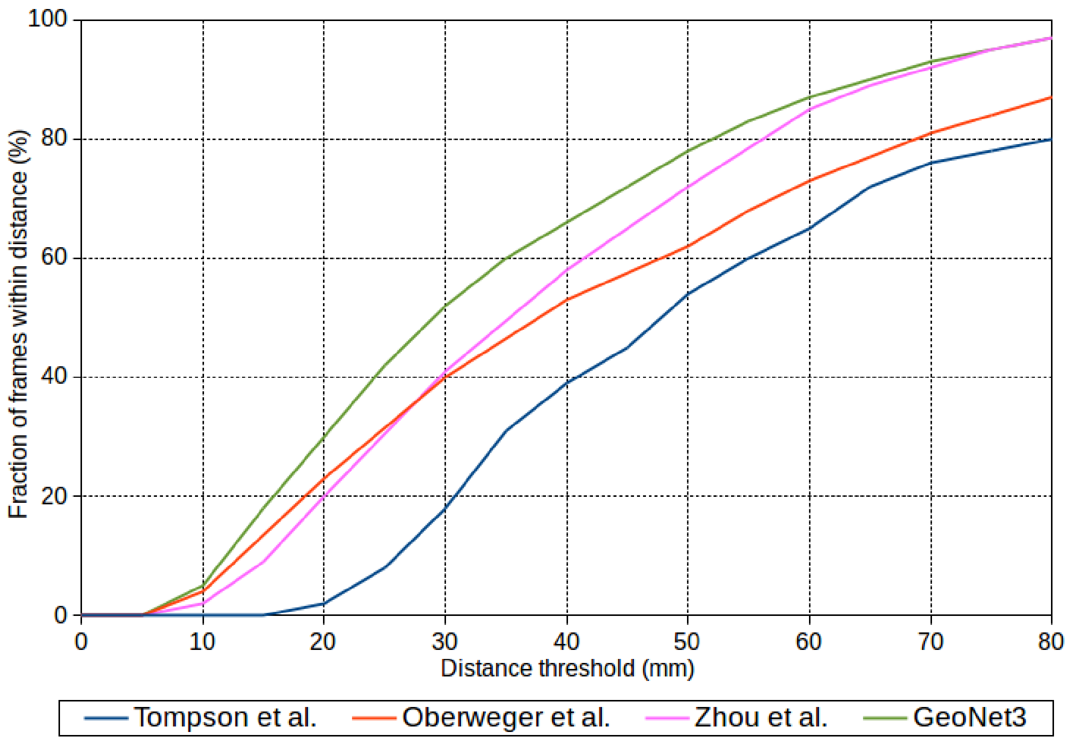

FIGURE 5.3: Success Rate of optimized GeoNet3 and related state-of-art CNN architectures.

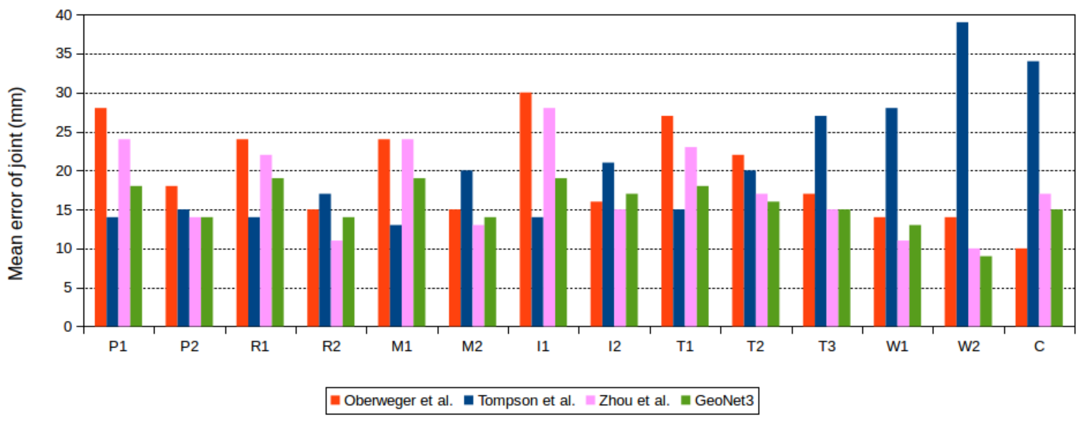

FIGURE 5.4: Mean Joint Error of optimized GeoNet3 and related state-of-art CNN architectures.

Chapter 6

Conclusion

In this project, the problem of hand pose estimation with convolutional neural networks from depth data has been studied. Two main contributions to this problem have been achieved. First, different convolutional neural network architectures from a network baseline in order to find the better prediction performance of hand pose estimation have been evaluated. Concretely, the importance of the depth of the network and the use of polling layers have been analyzed. Furthermore, an optimization with a random search and cross validation in order to find the best setting of the hyperparameters of the network has been performed. In our experimentation, the deeper network has been the best performing architecture and the correlation of the depth of the network and the network performance has been observed. As a conclusion, the depth of the network is and important factor towards more accurate hand pose estimation. The 3D hand pose estimation constitutes a high complexity problem that requires large deep models that can learn appropriate features from depth images to the target space. Also, the allocation and number of pooling layers is another important factor in the network performance due to the effect of an information loss caused by a high downsampling at the last layers.

The second contribution is related to the research about the effect of the output space or target representation in the network performance. For that, a new 3D pose representation that encodes geometrical relations between keypoints of the hand has been proposed. The challenge to find a model representation that can be transformed from 3D coordinates of hand keypoints in the real world and go back has been faced. As a result, the Euclidean Distance Matrix (EDM) has been proposed due to the feasibility to retrieve the original keypoint configuration with semidefinite programming. Furthermore, this representation encodes structural information of the pose and captures local correlations and dependencies between hand keypoints. The different network architectures have been compared with 3D coordinates of hand keypoints (baseline) and EDM representation from 3D coordinate keypoints respectively. Experimental results showed that EDM representation as output layer in the convolutional network increases the performance of all proposed architectures in terms of mean square error in training and testing sets. Also, best results in success-rate and error per keypoint can be appreciated. As a conclusion, the EDM representation as output of the convolutional network leverages more useful information for more accurate hand pose estimation. This result confirms that target representations with geometrical correspondences help to reduce some ambiguities of the pose estimation problem by incorporating structural information of the hand and capturing keypoint/joint correlations.

In a future work, the predictive performance of complex networks as residual networks, Region Ensemble networks or 3D Convolutional Neural Networks with EDM representations will be explored.

Bibliography

[1] Martín Abadi et al. *"TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems"*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] James Bergstra and Yoshua Bengio. "Random Search for Hyper-parameter Optimization". In: *J. Mach. Learn. Res.* 13 (2012), pp. 281–305.

[3] James Bergstra, Daniel Yamins, and David Cox. "Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms". In: *SciPy 2013: Scientific Computing in Python* (2013).

[4] François Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015.

[5] C. Couprie, C. Farabet, L. Najman, and Y. Lecun. "Indoor semantic segmentation using depth information". In: *Proceedings of the 2013 International Conference on Learning Representations, April 2013*. ICLR '13. 2013.

[6] Xiaoming Deng, Shuo Yang, Yinda Zhang, Ping Tan, Liang Chang, and Hongan Wang. "Hand3D: Hand Pose Estimation using 3D Neural Network". In: *arXiv preprint arXiv:1704.02224* (2017).

[7] P. Drineas, A. Javed, M. Magdon-Ismail, G. Pandurangant, R. Virrankoski, and A. Savvides. "Distance Matrix Reconstruction from Incomplete Distance Information for Sensor Network Localization". In: *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*. Vol. 2. 2006, pp. 536–544. DOI: 10.1109/SAHCN.2006.288510.

[8] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. "3D Convolutional Neural Networks for Efficient and Robust Hand Pose Estimation from Single Depth Images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. submitted for publication. Honolulu, HI, USA, 2017.

[9] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. "Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, USA, 2016, pp. 3593–3601.

[10] Hengkai Guo, Guijin Wang, Xinghao Chen, Cairong Zhang, Fei Qiao, and Huazhong Yang. "Region Ensemble Network: Improving Convolutional Network for Hand Pose Estimation". In: *arXiv preprint arXiv:1702.02447* (2017).

[11] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. "Hyperband: A novel bandit-based approach to hyperparameter optimization". In: *arXiv preprint arXiv:1603.06560* (2016).

[12] Francesc Moreno-Noguer. "3D Human Pose Estimation from a Single Image via Distance Matrix Regression". In: *CoRR* abs/1611.09010 (2016).

[13] N. Neverova, C. Wolf, F. Nebout, and G. Taylor. "Hand Pose Estimation through Weakly-Supervised Learning of a Rich Intermediate Representation". In: *CoRR* abs/1511.06728 (2015). URL: http://arxiv.org/abs/1511.06728.

[14] M. Oberweger, P. Wohlhart, and V. Lepetit. "Hands Deep in Deep Learning for Hand Pose Estimation". In: *Proceedings of 20th Computer Vision Winter Workshop*. CVWW '15. Styria, Austria, 2015, pp. 21–30.

[15] J. Schöning and G. Heidemann. "Taxonomy of 3D Sensors - A Survey of State-of-the-Art Consumer 3D-Reconstruction Sensors and their Field of Applications". In: *Proceedings of the 2016 International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. VISIGRAPP '16. Rome, Italy, 2016.

[16] Yi Shang and W. Ruml. "Improved MDS-based localization". In: *IEEE INFOCOM 2004*. Vol. 4. 2004, 2640–2651 vol.4. DOI: 10.1109/INFCOM.2004.1354683.

[17] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. "Real-time Human Pose Recognition in Parts from Single Depth Images". In: *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '11. Washington, DC, USA, 2011, pp. 1297–1304.

[18] Ayan Sinha, Chiho Choi, and Karthik Ramani. “Deephand: Robust hand pose estimation by completing a matrix imputed with deep features”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA, 2016, pp. 4150–4158.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2670313.

[20] J. Supancic III, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan. “Depth-Based Hand Pose Estimation: Methods, Data, and Challenges”. In: *Proceedings of the 2015 IEEE International Conference on Computer Vision*. ICCV ’15. Santiago, Chile, 2015.

[21] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. “Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks”. In: *ACM Trans. Graph.* 33.5 (2014), 169:1–169:10.

[22] R. H. Tutuncu, K. C. Toh, and M. J. Todd. “Solving semidefinite-quadratic-linear programs using SDPT3”. In: *Mathematical Programming* 95 (2003), pp. 189–217.

[23] Manuel Vinagre, Joan Aranda, and Alicia Casals. “A new relational geometric feature for human action recognition”. In: *Informatics in Control, Automation and Robotics*. Springer, Cham, 2015, pp. 263–278.

[24] A. Wetzler, R. Slossberg, and R. Kimmel. “Rule Of Thumb: Deep derotation for improved fingertip detection”. In: *CoRR* abs/1507.05726 (2015). URL: http://arxiv.org/abs/1507.05726.

[25] C. Xu and L. Cheng. “Efficient Hand Pose Estimation from a Single Depth Image”. In: *Proceedings of the 2013 IEEE International Conference on Computer Vision*. ICCV ’13. Sydney, Australia, 2013, pp. 3456–3462.

[26] Zhizhuo Zhang, Guoliang Li, Kim-Chuan Toh, and Wing-Kin Sung. “3D Chromosome Modeling with Semi-Definite Programming and Hi-C Data”. In: 20 (Nov. 2013), pp. 831–46.

[27] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. "Model-based Deep Hand Pose Estimation". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. IJCAI'16. New York, New York, USA, 2016, pp. 2421–2427.