UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

3D Reconstruction of Video Sign Language Dictionaries

Author:
Roman RIAZANTSEV

Supervisor:
Maksym DAVYDOV

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

“Computer Science is no more about computers than astronomy is about telescopes.”

Edsger W. Dijkstra

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

3D Reconstruction of Video Sign Language Dictionaries

by Roman RIAZANTSEV

Abstract

Today virtual and augmented reality applications become more and more popular. Such a trend creates a demand for 3D processing algorithms which may be applied to many areas. This work is focused on sign language video sequences. There are a lot of prerecorded photo and video dictionaries that can be transformed into 3D and unified in one place.

We research nuances of hand pose video sequence analysis as well as the influence of results refinement for 2D and 3D keypoint detection. Besides that, we designed a solution for the parametrization of hand shape and engineered system for 3D hand pose reconstruction.

Model show good results on train data but lack generalization. Retraining on multiple datasets and usage of various data augmentation techniques will improve performance.

Acknowledgements

I am honored to work alongside my supervisor Maksym Davydov and grateful for his skillful direction of my research. I also want to thank ADVA Soft for provided computational resources. Besides, I was lucky to meet my classmates and will never forget their consulting and moral support. And if we are talking about education - it was a pleasure to study under the supervision of the Applied Sciences Faculty team. Their efforts to improve the educational process in Ukraine is a source of inspiration. And last but not least, I have the best brother in the world. His existence gives me the strength to overcome difficulties.

Contents

Declaration of Authorship ii

Abstract iv

Acknowledgements v

1 Introduction 1
- 1.1 Context . . . 1
- 1.2 Problem . . . 1
- 1.3 Descriptions of the thesis chapters . . . 2

2 Background and Related Works 3
- 2.1 Background overview . . . 3
  - 2.1.1 An artificial neural networks . . . 3
  - 2.1.2 Blocks and cells . . . 4
- 2.2 Methods for 3D shape estimation . . . 6
  - 2.2.1 Networks for 2D key-point detection . . . 6
  - 2.2.2 Systems for 3D reconstruction . . . 8
  - 2.2.3 MANO model . . . 9

3 3D hand pose reconstruction 10
- 3.1 Overview of proposed methods . . . 10
- 3.2 Neural networks for 2D key point detection . . . 10
- 3.3 ANN for depth estimation (Network B) . . . 12
- 3.4 Network for estimation of shape parameters . . . 14

4 Experiments and results 15
- 4.1 Datasets . . . 15
- 4.2 Data preprocessing algorithms . . . 16
- 4.3 Training 3D shape estimation stages . . . 16
  - 4.3.1 Training and testing of networks for 2D key-point detection . . 16
  - 4.3.2 Training and testing of networks for Depth detection . . . 18
  - 4.3.3 Training and testing of networks for shape parameterisation . . 19
- 4.4 Annotation system analysis . . . 19

5 Conclusion 22
- 5.1 Future work . . . 22

Bibliography 23

List of Figures

2.1 Convolution 5
2.2 Residual learning: a building block [9]. 5
2.3 LSTM cell 6
2.4 Generalized schema of 3D hand pose estimation 6
2.5 a) UNet architecture [10] , b) Stacked Hourglass Network [11] 7
2.6 Illustration of parallel computational pipelines used in Open Pose [14] 8
2.7 Scheme of the method described in paper 3D Hand Shape and Pose Estimation from a Single RGB Image. [15] 9

3.1 Building blocks and assembled networks for 2D key point detection: a)U-Net architecture; b)Hourglass Like Module c)Architecture of STH_2; d)Architecture of STH_3. 11
3.2 Studied building blocks and networks for depth estimation: a)Feature Extractor; b)Dense layers with additional depth inputs; c)Dense layers with additional depth outputs. 13
3.3 Studied architectures for shape parameterization. a) Shema of MANO input-output mapping; b)Architecture of C_RNN network; c)Architecture of C_MLP network. 14

4.1 Examples of FreiHand 3D reconstructions 20
4.2 Examples of SynthHands_A 3D reconstruction 21
4.3 Examples of ASL 3D reconstruction 21

List of Tables

3.1 Differences between nets A 12

4.1 Datasets info 15
4.2 Mean errors on train dataset for networks A 17
4.3 Mean errors on test dataset for networks A 17
4.4 Mean errors on SynthHand dataset for networks A 17
4.5 Mean errors on SynthHand_A dataset for networks A 17
4.6 Mean errors for networks B on train dataset 18
4.7 Mean errors for networks B on test dataset 18
4.8 Mean errors for networks B on SynthHand dataset 19
4.9 Mean errors for networks B on SynthHand_A dataset 19
4.10 Mean errors for C networks 19
4.11 Intermediate and final 3D key point errors 20

List of Abbreviations

CNN	**C**onvolutional **N**eural **N**etworks
RNN	**R**esidual **N**eural **N**etworks
MSE	**M**ean **S**quared **E**rror
ASL	**A**merican **S**ign **L**anguage
LSTM	**L**ong **S**ort **T**erm **M**emory
MANO	hand **M**odel with **A**rticulated and **N**on-rigid def**O**rmations
MLP	**M**ulti**L**ayer **P**erceptron
HGM	**H**our**G**lass Like **M**odule
ANN	**A**rtificial **N**eural **N**etwork

Dedicated to my brother

Chapter 1

Introduction

1.1 Context

Today virtual and augmented reality technologies (AR/VR) are becoming more and more popular. Prominent mobile applications like Snapchat or the Pokemon Go reflect that. Such a trend creates a high demand for 3D image data processing, which applies to many areas.

Samsung, in August 2019, released an application called ’3D Scanner’ that allows users to scan objects and create their 3D model, which then can be shared with others. For such a program, it is convenient to use algorithms based on data from a depth camera. However, algorithms based on the use of depth data cannot be easily applied to a vast amount of 2D data recorded using RGB sensors. And despite the growing availability of depth cameras and the information obtained from them, 2D data and RGB sensors themselves are still more accessible. We propose methods that can be used for a similar kind of 3D modeling applications.

1.2 Problem

Sign language dictionaries are widely available but lack the methods for their conversion into 3D. There is plenty of single images as well as video sequences in 2D. Even though there is a cases of dictionaries recorded from multiple views, often people who want to learn sign language see only the front view of the hands. However, views from all angles carry value, as they reflect the nuances between similar words. Besides that, existing dictionaries don’t unify into one universal solution. We don’t observe a sequence of works dedicated to the reconstruction of existing sign language dictionaries in one unified 3D database. Our goal is to speed up research of 3D reconstructions of available sign language lessons for further usage in AR/VR applications.

We need to cover the processing of both sequences and single images to make future reconstruction into 3D video dictionaries more flexible. Working with video implies computational problems related to blurred frames which exist due to high speed of movement, and complex hand poses with overlapping hand parts along the z-axis.

Besides that, sign language is a very broad field. As an initial step, we decided to create 3D annotations for the fingerspelling. Fingerspelling is a process of showing words by letters, which often used to show titles, people’s names, brands, etc.

To analyze the nuances of video hand pose reconstruction and engineer solution for subset of ASL we set research questions and engineering goals.

Research questions

- How addition of synthetic video frames affect performance of 3D reconstruction methods.
- How refinement of results affects performance of 3D reconstruction methods.

Engineering goal

- Crate system for annotation of American Sign Language subset.

Contributions

- We researched how additional input data influence 3D keypoint estimation.
- We researched how additional supervision reflects on 3D keypoint estimation results.
- We used RNN to parameterize hand shape. We didn't find mentions of this approach in literature.
- We designed a modular system for training and evaluation of hand pose reconstruction.

1.3 Descriptions of the thesis chapters

In **chapter 2**, we review the main technical concepts on which based techniques for 3D hand pose reconstruction. The chapter also provides an overview of existing end-to-end systems for hand.

In **chapter 3**, we propose our solution for hand pose estimation. And discuss differences between architectures that can be used for intermediate computations of key points in 2D and 3D. We are also covering datasets suitable for the training and evaluation of our algorithm.

In **chapter 4**, we show training results and discuss how differences in architectures influence performance. This chapter also showcases performance on the American Sign Language dataset.

In **chapter 5**, we are summing up research outcomes and discussing how the system for sing language reconstruction can be improved.

Chapter 2

Background and Related Works

2.1 Background overview

The task of determining the position of an object in space is not new. Over the past 20 years, a large number of works have been aimed at solving this problem [1], [2]. A lot has changed with the advent of depth sensors and neural networks. These technologies introduce new approaches to comprehensive scene analysis. Depth cameras produce information about the distance to an object, which allows reconstructions of more accurate 3D models, and neural networks calculate complex correlations in image patterns. Since 2012, neural networks started to overperform most of the classical methods in segmentation and classification problems. A large number of methods use a combination of depth-camera output and neural network for 3D reconstruction of the body position [3], [4]. The above technologies also apply widely to the hands. Often, researchers use a combination of depth sensors and gloves, which record the 3D position of the hand. Several sensors are used for the collection of fully labeled training samples for 3D reconstruction, which may include depth map, joint angles, and 3D positions [5]–[7].

Because we are solving the problem without the usage of depth sensors, from two stated technologies, we will provide only neural networks review.

2.1.1 An artificial neural networks

An artificial neural network is a mathematical model well suited for the approximation of highly nonlinear functions. The main component of a neural network is a neuron, which usually consists of a linear regression followed by the activation function. Activation functions are responsible for the addition of nonlinear transformations. Neurons can be combined into various layers, for example, fully connected layers or convolutional layers.

Neurons in a fully connected layer extend the idea of a linear model, where output is a linear combination of fixed nonlinear basis functions $\phi(.)$, passed to the nonlinear activation $f(.)$ [8]:

$$y(x, w) = f(\sum_{j=1}^{M} w_j \phi_j(x)) \tag{2.1}$$

The extension of the idea consists of two facts. First is that nonlinear basis function is parametrized too and optimized alongside parameters w. Second is that basis function has the same form as 5.1.[8]

The basic neural network model can be described as a series of functional transformations. First, we construct M linear combinations of the input variables $x_1, \ldots, x_D$

in the following form:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + 2_{j0}^{(1)} \tag{2.2}$$

where j = 1, . . . , M, and the superscript (1) indicates that the corresponding parameters are in the first 'layer' of the network. The output of 2.2 called activation and transformed using a differentiable, nonlinear activation function h(.):

$$z_j = h(a_j) \tag{2.3}$$

where z_j is called hidden unit.

A popular choice of an activation function for multiple binary classification problems is a sigmoidal activation. It is a convenient activation function for computer vision tasks such as image segmentation or key-points detection:

$$\sigma(a) = \frac{1}{1 + exp(-a)} \tag{2.4}$$

So then, a multilayer neural network with sigmoidal activation function can be represented by the equation 2.5:

$$y_k(x, w) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{k0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{2.5}$$

where w is a matrix of all trainable parameters [8].

Convolutional layers are a fundamental part of most networks described in this thesis. The core component of a layer is the sliding kernel matrix (filter) with trainable parameters. Elements of the kernel matrix multiplied element-wise with a region of the input matrix. The following equation describes the process:

$$G[m, n] = (f * h)[m, n] = \sum_j \sum_k h[j, k] f[m - j, n - k] \tag{2.6}$$

where f - is an input matrix, h kernel, m - rows, n - columns

Multiplication results are summated with bias, and output is represented as an element of the feature map matrix. Feature map usually transformed with some activation function. Because for one feature map exist only one kernel matrix with trainable parameters, a number of weights are significantly smaller than in a fully connected layer. The trainable convolution can be described as linear regression trained on segments of the input data, not all data at once. By changing the dimension of the kernel or sparseness of the input region, we can determine the logic of pattern recognition performed by the layer.

In the simplest case, the trainable parameters of the network are only weights of a linear regression that optimized using gradient descent. Various techniques, such as batch normalization, dropout, or skip connections, are used to prevent overfitting or speed up the optimization of parameters.

In all the reviewed and proposed methods for 3D hand modeling, convolutional neural networks (CNNs) play an essential role.

2.1.2 Blocks and cells

In this section, we review two structures built from layers. Both of them mainly solve the problem of vanishing gradient. The first one is called residual block and

FIGURE 2.1: Convolution

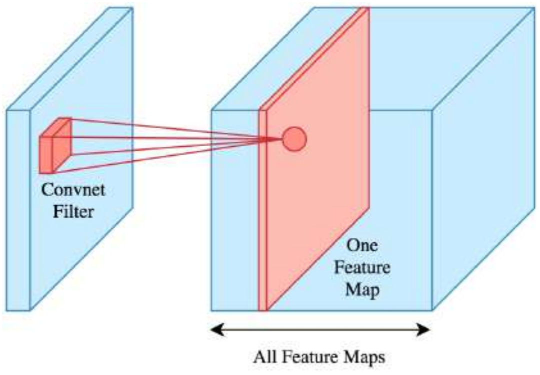
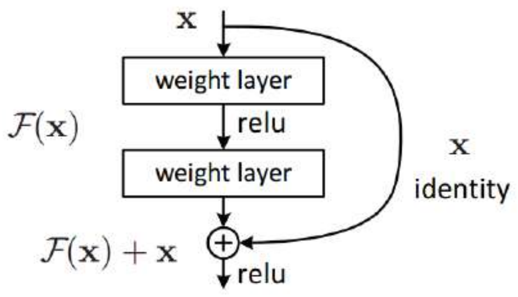


FIGURE 2.2: Residual learning: a building block [9].


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

was introduced as a solution for efficient training of deep networks with depth in hundreds of layers [9] . Residual block returns not the output of the single final layer but the sum of multiple outputs. This design allows the optimizer to update weight at the beginning of the network more efficiently.

The second structure is called LSTM cell and related to Recurrent Neural Networks that weren't touched before in a thesis. RNNs are used for sequence modeling where there is a task of mapping one sequence to another. Each element of a sequence is fed to network cell, which extracts patterns of elements. The problem, in this case, again vanishing gradient for the case of long sequences and can be solved by LSTM cell. The sequence of computational operations is listed on equations 2.7-2.12 and illustrated on Fig 2.3.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{2.7}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2.8}$$

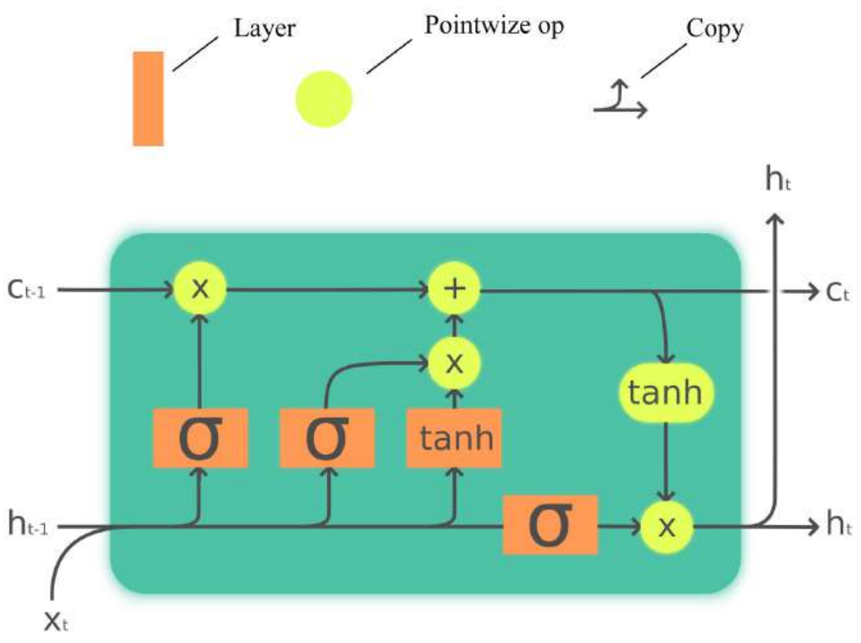
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{2.9}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{2.10}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{2.11}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{2.12}$$

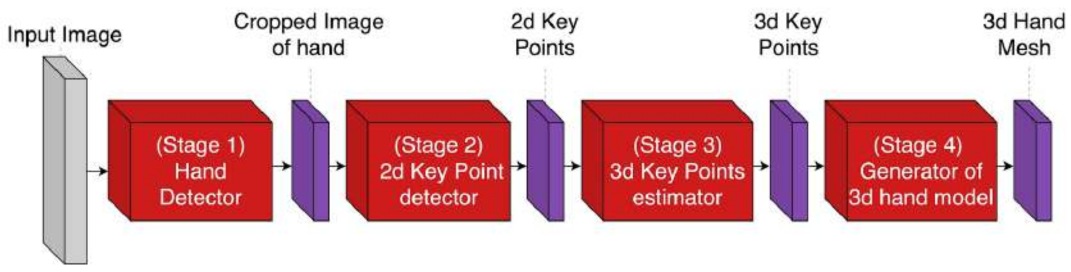
FIGURE 2.3: LSTM cell



2.2 Methods for 3D shape estimation

Most methods for 3D hand pose generation from a single RGB image can be generalized into four stages. The first stage is the detection of hands in the input image and cropping localized area; the second is the detection of hand key-points in 2D; the third is a mapping of 2D locations into 3D, and the fourth is a generation of the 3D hand model.

FIGURE 2.4: Generalized schema of 3D hand pose estimation


Input Image
Cropped Image of hand
2d Key Points
3d Key Points
3d Hand Mesh
(Stage 1) Hand Detector
(Stage 2) 2d Key Point detector
(Stage 3) 3d Key Points estimator
(Stage 4) Generator of 3d hand model

2.2.1 Networks for 2D key-point detection

Estimation of 3D structures such as 3D key-points or 3D hand model is the hardest to train part of the pipeline. For higher accuracy of those stages, it is essential to accurately estimate 2D key-points of the hand, because this data contains two-thirds of the 3D locations. We will review a few models suitable to address tasks of 2D key-points estimation. We will discuss three CNNs, namely U-Net, Stacked Hourglass Network, and Open Pose.

The UNet is an architecture which first extracts image features by downsampling layers and then upsamples them alongside sequential concatenation of outputs from

FIGURE 2.5: a) UNet architecture [10] , b) Stacked Hourglass Network [11]

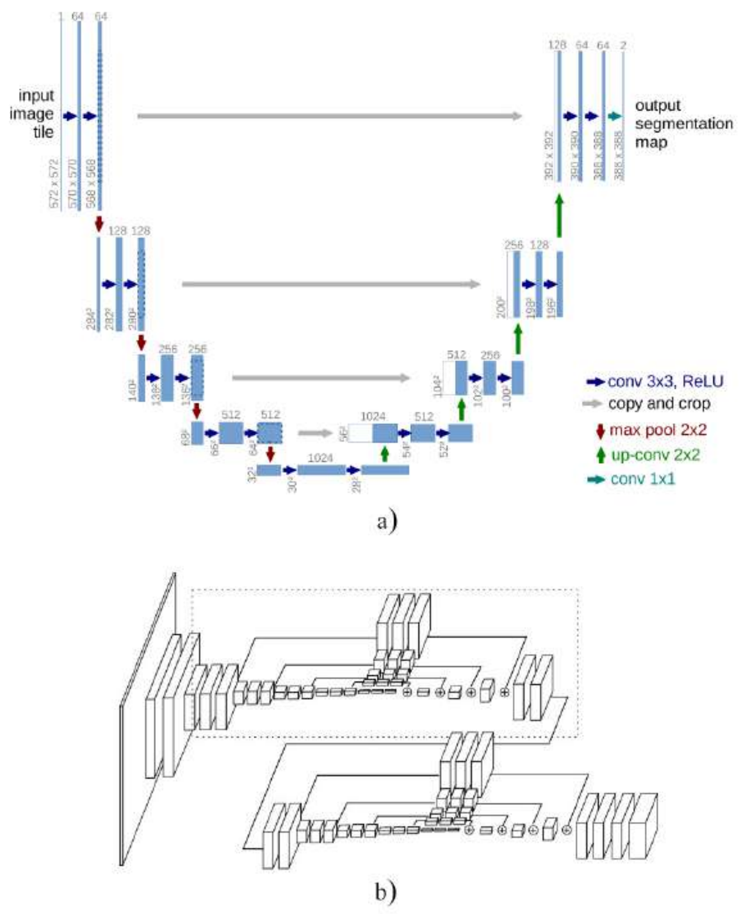
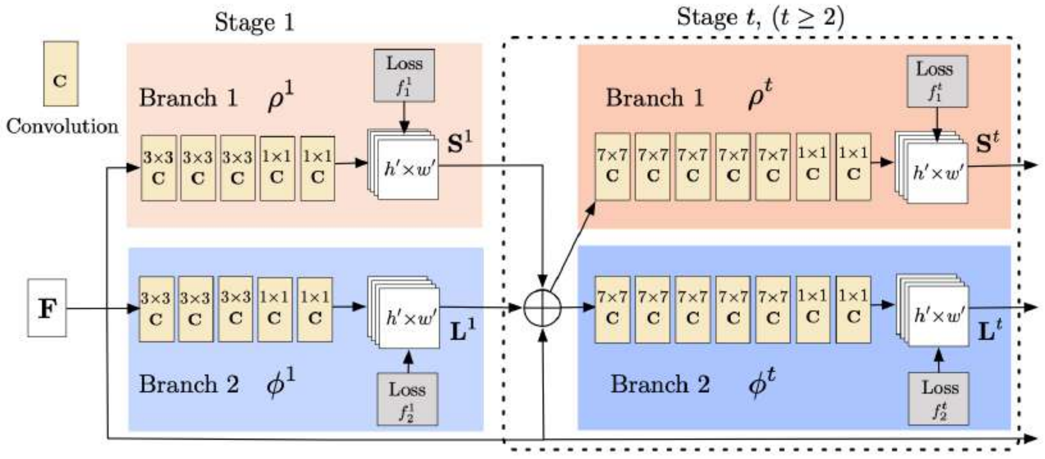


FIGURE 2.6: Illustration of parallel computational pipelines used in Open Pose [14]



previous layers [10]. U-Net like architectures have proven to work with the task of 2D key-points detection [12].

Stacked Hourglass Network extends the idea of U-Net and stacks several architectures of a similar type into one pipeline with supervision after each step [11]. Building blocks are residual and skip connections are implemented by the addition of feature maps rather than concatenation.

Open pose detector is a CNN that consists of multiple computational stages and two parallel computational pipelines. Each stage is estimating both locations of 2D key-points and connections between them. All stages except the first one are estimating information about joints based on previous results. In such a way, the detector is reffing computational results with each iteration.

2.2.2 Systems for 3D reconstruction

The paper [13] introduces a three-stage algorithm that localizes the hands and determines the key-points in 2D at the first two stages and calculates 3D reconstruction at the third.

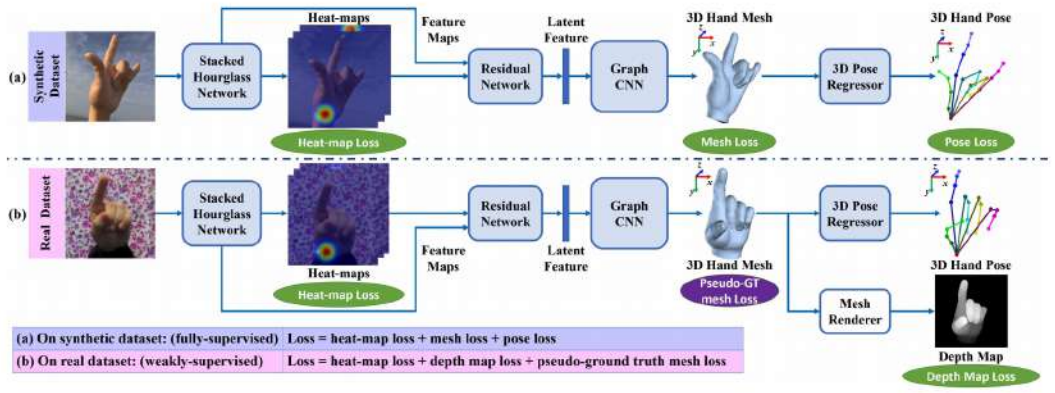
The first stage is the YOLOv2 neural network ('you only look once'), which identifies the position of the hands. YOLOv2 is a CNN which used for localization and classification of multiple objects at once.

The localized hands are passed to the OpenPose detector.

These two neural networks localize 21 2D key-points on the video frames, which are then used as a target in the inverse kinematic optimization problem. A distinct drawback of this method is the limitation caused by the error of the OpenPose detector. This error causes the algorithm to optimize 3D locations using the wrong 2D key-points. Nevertheless, the addition of a hand position from a different view makes it possible to improve the optimization problem, and hence the accuracy. The runtime of the method on Nvidia GTX 1070 GPU is close to 53 ms.

The work [15] describes one of the few methods which fully reconstructs the 3D shape of the hand. It introduces a graph convolutional neural network (CNN) for generating 3D mesh. The work uses centered images of hands as input, thus hand detection was not necessary. Therefore, the first part of the approach is 2D key-point detection, which is based on Stacked Hourglass Networks. The second is the encoding of 2D features, and the third is 3D reconstruction with a graph CNN

FIGURE 2.7: Scheme of the method described in paper 3D Hand Shape and Pose Estimation from a Single RGB Image. [15]



network. The network outperforms state-of-the-art methods on RHD [16] and STB [17] datasets. The runtime of the method on Nvidia GTX 1080 GPU is, on average, 19.9ms. The pretrained model is available, but the training dataset is not.

2.2.3 MANO model

MANO is a differentiable parametric model of the hand, which can be integrated into computational pipelines. MANO learned from around 1000 3D scans of hands in various poses. The model takes as input parameters of pose and shape, which then mapped to the cloud of points, which represent the hand surface [18].

Chapter 3

3D hand pose reconstruction

3.1 Overview of proposed methods

The work is aimed at solving three tasks: studying the influence of preliminary frames on the reconstruction of video sequences; investigating the process of results refinement; image reconstruction of hands displaying the letters of American Sign Language. We are solving all by the introduced 3-stage computational pipeline, where every stage is a neural network designed to solve a small task.

The proposed method can be described by the schema of 3D hand pose estimation close to specified in Fig 2.4. Except it does not have the first stage of the hand localization stage and work directly with centered hands. In that way, the introduced method can be generalized to the sequence of architectures A, B, and C. We sequentially compute locations of 2D key-points by network A, depth for each key point by B, and after that, calculate the vector of MANO parameters from key points by C.

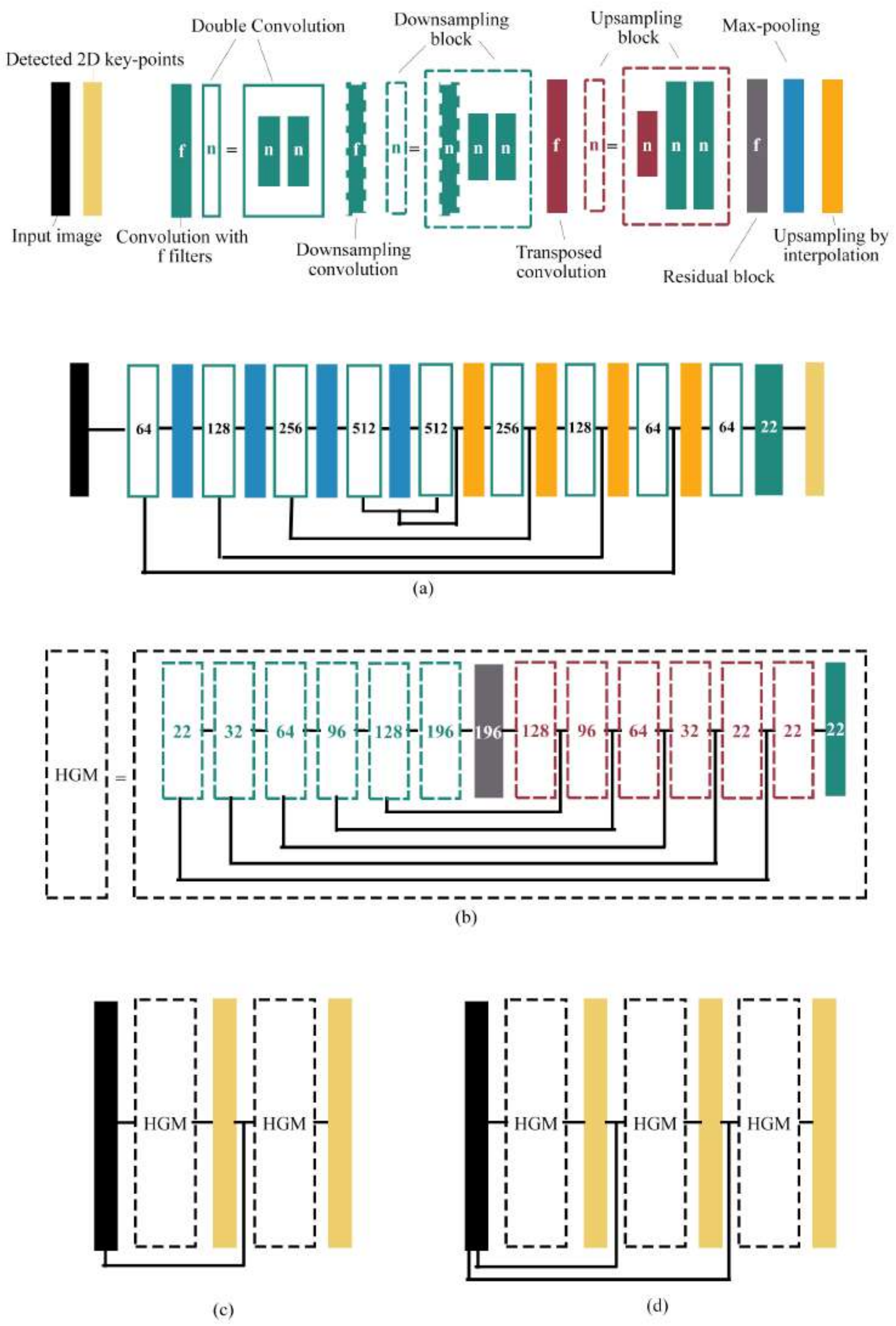
In total, 13 ANN architectures were tested for tasks A, B, and C combined. Six designs were studied as candidates for network A, five - for B, and two for C. The networks A and B can be considered as one architecture for 3D key-point detection. For A and B, we were focused on the influence of two factors on the training process. Those factors are additional frames at input and refinement of predictions. To investigate the difference in the training trends was developed a system for parallel training and evaluation of multiple networks. Network C stands alone from networks A and B because it doesn't process images. Its task is to estimate the vector of MANO hand surface generation parameters from a set of 3D points. This stage finishes the computational pipeline, which allows us to annotate a subset of American Sign Language.

3.2 Neural networks for 2D key point detection

Network A takes as an input RGB information and outputs 22 heatmaps of size 224x224, {H1, H2,..., H22}, where each heatmap Hk from indicates the probability of observing the kth keypoint and for k=22 probability of no key points. A similar approach has proven to work efficiently in [14].

Two Stacked Hourglass Like and one U-Net architecture are candidates for main network at stage A for 2D key-point detection. Each candidate has a variation for video and image processing. In total, there are 6 architectures for 2D key point localizations with the following id's: UNET_VID, STH_2_VID, STH_3_VID, UNET_PIC, STH_2_PIC, STH_3_PIC. Where 'STH_' stands for Stacked Hourglass like architecture; _VID means that input contains concatenated RGB frames and 2D points of the previous frame; _PIC means that input contains a single RGB image.

FIGURE 3.1: Building blocks and assembled networks for 2D key point detection: a)U-Net architecture; b)Hourglass Like Module c)Architecture of STH_2; d)Architecture of STH_3.



Let kc denote the convolutional layer with k filters and stride 1, kd - the convolutional layer with k filters and stride 2, ku - the transposed convolutional layer with k filters and stride 2, kfc - fully connected layer with k neurons, mp - max pooling with 2x2 filter, up2 is x2 upsampling by interpolation, kr - residual block from Fig 2.2 with k filters. All convolutional layers except for the last one are followed by batch normalization and ReLU activation.

Implementation of U-Net is: 64c, 64c, mp, 128c, 128c, mp, 256c, 256c, mp, 512c, 512c, mp, 512c, 512c, up2, 256c, 256c, up2, 128c, 128c , up2, 128c, 128c, up2, 64c, 64c, up2, 64c, 64c, 22c. Outputs of layers are concatenated in order shown in Fig. 3.1 a).

The architecture of proposed HGM is: 22d, 22c, 22c, 32d, 32c, 32c, 64d, 64c, 64c, 128d, 128c, 128c, 196d, 196c, 196c, 196r, 128u, 128c, 128c, 96u, 96c, 96c, 64u, 64c, 64c, 32u, 32c, 32c, 22u, 22c, 22c, 22u, 22c, 22c, 22c. Outputs of layers summed in order shown in Fig. 3.1 b).

The architecture of STH_2 consist of two hourglass like modules where the second module takes as input image concatenated with the initial prediction from the first module, as illustrated in Fig 3.1 c). The architecture of STH_3 set in the same way as STH_2, except there is one additional module for refinement, as shown in Fig 3.1 d).

It is important to accentuate that architectures drastically differ in size.

net_id	num_of_parameters	num_of_refinements	skip-connection type
STH_2_PIC	6106056	2	addition
STH_2_VID	6115956	2	addition
STH_3_PIC	9161262	3	addition
STH_3_VID	9176112	3	addition
UNET_PIC	13396694	1	concatenation
UNET_VID	13411094	1	concatenation

TABLE 3.1: Differences between nets A

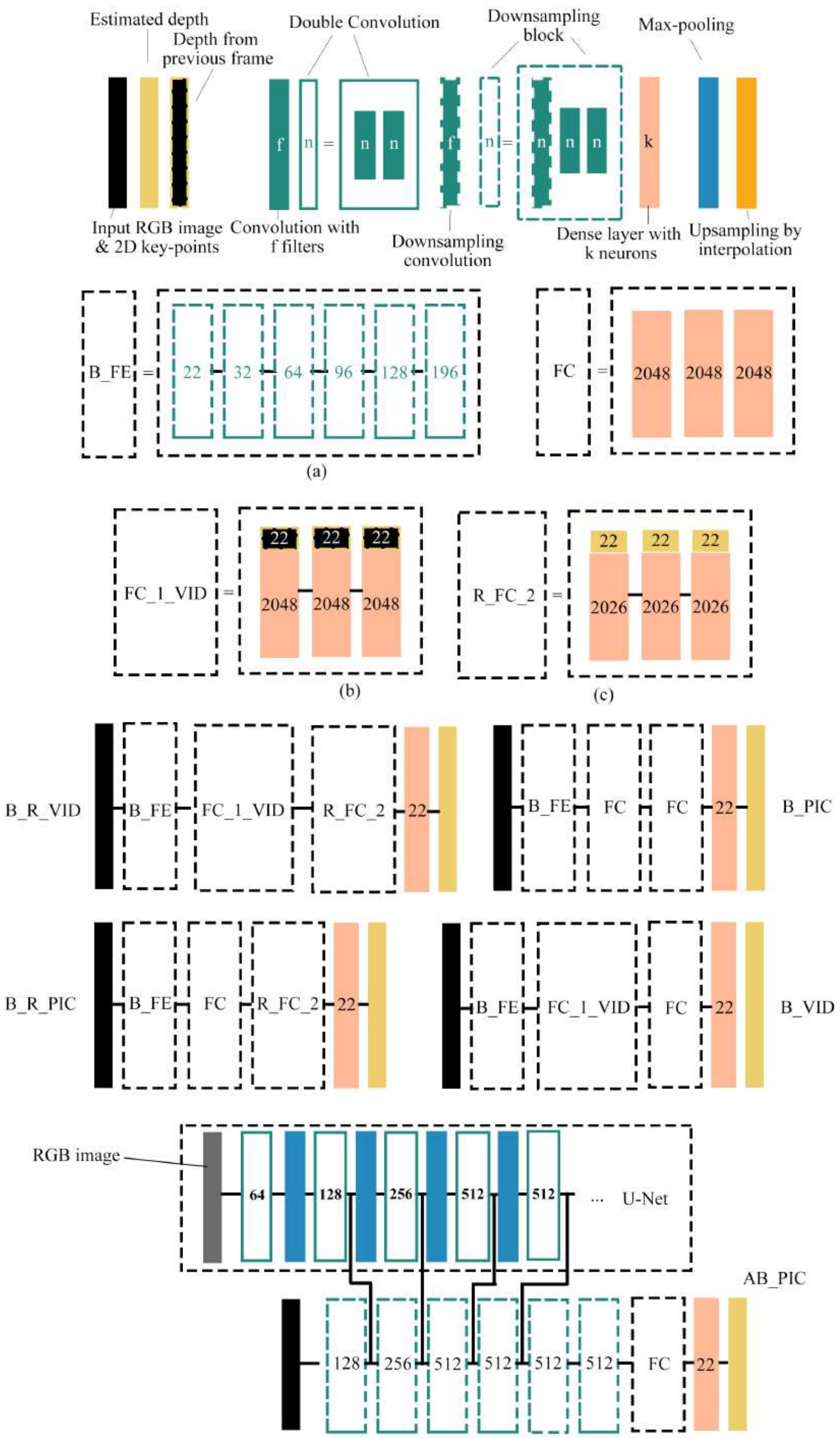
3.3 ANN for depth estimation (Network B)

Network B is used for the estimation of 21 depth values for each key point. We propose 5 architectures for depth estimation marked as B_PIC, B_VID, B_R_PIC, B_R_VID, AB_PIC. Where B_R means that output is calculated multiple times (refined), AB stand for additional usage of features from network A, _VID means that input contains concatenated RGB frames and 2D points from current and previous frame, as well as depth from the previous frame; _PIC means that input contains concatenated RGB data with 2D key points from a single image.

All networks have two parts. The first part is Feature Extractor (FE), and the second is a set of fully connected blocks (FC). The set of convolutional layers followed by dense layers is an effective solution for the image to vector problems [19]. Architecture of FE is same for B_PIC, B_VID, B_R_PIC, B_R_VID: 16d, 16c, 16c, 32d, 32c, 32c, 64d, 64c, 64c, 128d, 128c, 128c. Feature Extractor of network with id AB_PIC has following architecture: 128d, 128c, 128c, 256d, 256c, 256c, 512d, 512c, 512c, 512d, 512c, 512c, 512d, 512c, 512c, 512d, 512c, 512c. Outputs of Feature Extractor layers for network AB_PIC concatenated with outputs of UNET layers in a way shown at Fig. 3.2

Sets of fully connected blocks are different for all networks. Networks B_PIC, B_VID, B_R_PIC, B_R_VID ends with 7 dense layers. Networks that work with video sequences have additional inputs at first three blocks, as shown at Fig 3.2 b).

FIGURE 3.2: Studied building blocks and networks for depth estimation: a)Feature Extractor; b)Dense layers with additional depth inputs; c)Dense layers with additional depth outputs.

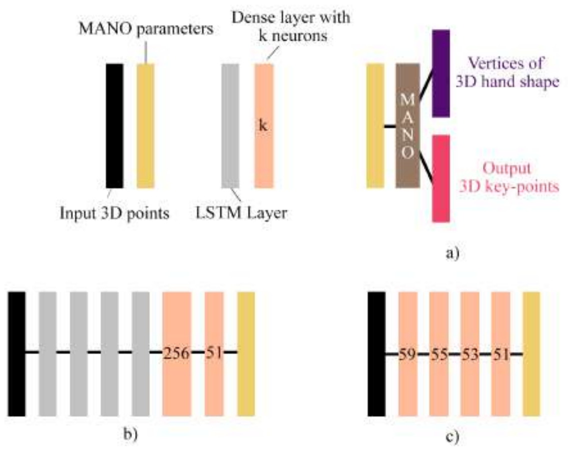


Id's of networks with refinement of results marked with _R and optimized so that each of 4 ending layers has 22 neurons responsible for depth info. Network AB_PIC has 4 fully connected layers with no additional info or refinement, and the output of the last layer only is depth.

3.4 Network for estimation of shape parameters

The 3D shape of a hand is represented as a mesh with 778 vertices. Those vertices are encoded in 51 parameters and decoded by the MANO model. We convert 21 3D key-points into MANO parameters by network C.

FIGURE 3.3: Studied architectures for shape parameterization. a) Shema of MANO input-output mapping; b)Architecture of C_RNN network; c)Architecture of C_MLP network.



We considered two architectures as candidates for network C. The first model has id 'C_RNN' and consists of 4 LSTM layers and fully connected output. The second model with id 'C_MLP' is a set of fully connected layers. Illustrated in Fig. 3.3.

Chapter 4

Experiments and results

4.1 Datasets

Several datasets were examined, and the most suitable was selected for our task. Large portion of datasets for 3D reconstruction contain depth maps, key points, but not RGB image: NYU, ICVL, MSRA15, BigHand2.2M, SynHand5M, FHAD, MSRC(FingerPaint), HandNet, Hands in Action, MSRA14 [5]–[7], [20]–[26]. For the problem of reconstruction from a single image, most appropriate datasets are those that feature both RGB records and key points: FreiHAND, GANerated Hands, EgoDexter, SynthHands, STB, Dexter+Object, UCI-EGO, MHP [17], [20], [27]–[30]. The possible complication of combining different datasets is that the number of key points, record types, and camera parameters may not match. We left only datasets with a central position of a hand and 21 labeled key points.

FreiHAND Dataset is a hand pose dataset for hand pose estimation from a single image. The dataset contains shots with 4 different data augmentations annotated with 21 key points for 2D and 3D spaces. The total amount of training samples is 130240 for single view training. So 32560 poses for training [20].

GANerated Hands Dataset contains 330,000 examples annotated with 21 key points for 2D and 3D spaces. The downside of this dataset is that images are synthetically generated and have distorted edges of hands. All of them are recorded from one viewpoint [27].

SynthHands is a synthetic dataset, which provides information about 63,530 frames recorded from 5 views. Learning examples contain both RGB and depth records and represent records with and without object interaction. Data annotated for 21 points in 3D space. Hands were generated using Unity3D engine but animated using data captured from real motion [28].

We chose FreiHAND Dataset and SynthHands dataset for training and evaluation, respectively. Also, we used Home Object Dataset for background augmentation of SynthHands dataset. Home Object Dataset [31] contains 450 photos of random objects. Augmented SynthHand dataset denoted in work as 'SynthHands_A'.

Dataset name	Number of images	Synthetic / Real	Number of camera views
FreiHAND	32,560	Real	1
GANerated Hands	330,000	Synthetic	1
SynthHands	63,530	Synthetic	5

TABLE 4.1: Datasets info

Besides annotated datasets for training and testing, we used the ASL dataset for reconstructions. Kaggle ASL dataset [32] contains 87000 of hand poses. Images were labeled for the task of sign language recognition. There are 26 classes for letters and three classes with images for SPACE, DELETE, and NOTHING.

4.2 Data preprocessing algorithms

As there were no 3D annotated video dictionaries of sign language available at the time of the study, an artificial sequencing procedure was proposed. It is proposed to sort images of hands with annotations to mimic real-world transition of hands from frame to frame. For each record in a dataset, we search one with the closest 3D position and use both in the training process. Because frame n and n-1 have close 3D positions in the real world, such sorting allows us to use a non-video dataset for usage in tasks of video processing. Usage of fake video sequences different from real-world applications in two ways. First - the transition between hands not as smooth as in the real world. Second - in our study, we used ground truth annotation for the previous frame at the input for the network. Because the previous frame is more random than in real videos - ground truth data in a way mimic noisy prediction for the previous frame. But, of course, training on annotated synthetic video does not suffer from an accumulation of noise as much as if for each next frame, we would use the previous prediction of the network.

To solve the task of shape reconstruction, we are adding additional data preprocessing. First, we are masking density estimation of MANO annotations to sample more data. We are creating a new dataset by sampling vectors from distribution, passing it to a MANO model, and obtain corresponding 3D shape and key-points. The new dataset allows us to train network C on a large number of sampled vectors.

4.3 Training 3D shape estimation stages

For training and testing was used Frei Hand Dataset split for test and train subsets in ratio 1:4. As an additional dataset for evaluation, we used SynthHand dataset with and without the addition of background.

It is needed to account for different scales of key point spaces. Datasets or hand models encode certain magnitude of hand, so simultaneous work with different coordinates requires unification. In our case MANO model, Synth Hand Dataset and Frei Hand Dataset are all in different scales. Frei Hand Dataset hands are 990 times smaller than MANO reconstructions. In contrast, Synth Hand Dataset is 0.94 times smaller than MANO hands. It does not influence training for 2D key-points because data projected to the image plane and unified in that way. For depth, it is more important to have annotation in one scale. Networks for depth estimation learn both ratios between fingers depth as well as absolute depth. We have transformed the absolute depth of SynthHand Dataset annotations to match the average depth of FreiHand Dataset. For calculation of 3D error, we transformed all coordinates to MANO model space. It is the shortest way to achieve unification given that vector of MANO parameters is part of our pipeline, and that scale is used for training.

4.3.1 Training and testing of networks for 2D key-point detection

We trained and tested 6 architectures in parallel. Adam optimizer was used with learning rate 0.0001 and trained on mini-batches of size 4 from FreiHand dataset for

40820 steps. We tested Mean Squared Error of predicted probabilities of location for each key point as well as absolute distance between the predicted location of key point and ground truth in 2D. For each network, we tested performance on 300 random images from 4 datasets. Results of testing on the train and test subsets of FreiHand Dataset are presented in tables 4.4 and 4.5, respectively. Metrics measured on Synth Hand Dataset presented in Table 4.6 for data without background, and table 4.7 tests on images with augmented backgrounds.

Results in tables 4.2-4.5 show that providing additional video frames decreases mean errors in most cases by 5-30%. And additional refinement improves results only after a certain number of iteration. The proposed variation of Stack Hourglass Model has better results than larger UNET architecture on train subset as well on Synth Hand data-set with and without background. But since it has a larger error on test subset on real data, we assume that UNET generalizes better.

	STH_2_PIC	STH_2_VID	STH_3_PIC	STH_3_VID	UNET_PIC	UNET_VID
mse (points)	2.64E+02	1.63E+02	1.92E+02	1.67E+02	2.15E+02	1.16E+02
l1 (points)	10.097	7.198	7.861	7.545	7.192	6.259
mse (heatmaps)	8.11E-04	7.68E-04	7.89E-04	7.82E-04	7.60E-04	7.62E-04

TABLE 4.2: Mean errors on train dataset for networks A

	STH_2_PIC	STH_2_VID	STH_3_PIC	STH_3_VID	UNET_PIC	UNET_VID
mse (points)	2.86E+02	2.01E+02	2.17E+02	1.89E+02	1.92E+02	1.67E+02
l1 (points)	10.582	8.448	8.507	8.59	6.906	7.757
mse (heatmaps)	8.26E-04	8.11E-04	8.09E-04	8.24E-04	7.77E-04	7.99E-04

TABLE 4.3: Mean errors on test dataset for networks A

	STH_2_PIC	STH_2_VID	STH_3_PIC	STH_3_VID	UNET_PIC	UNET_VID
mse (points)	8.66E+02	6.62E+02	7.47E+02	3.66E+02	8.91E+02	4.02E+02
l1 (points)	20.327	15.222	18.373	11.857	21.03	12.5
mse (heatmaps)	8.87E-04	8.47E-04	8.86E-04	8.33E-04	9.15E-04	8.42E-04

TABLE 4.4: Mean errors on SynthHand dataset for networks A

	STH_2_PIC	STH_2_VID	STH_3_PIC	STH_3_VID	UNET_PIC	UNET_VID
mse (points)	8.61E+02	5.34E+02	8.33E+02	5.41E+02	1.15E+03	4.24E+02
l1 (points)	21.201	14.359	20.305	14.486	24.221	13.042
mse (heatmaps)	9.04E-04	8.69E-04	9.12E-04	8.76E-04	9.13E-04	8.68E-04

TABLE 4.5: Mean errors on SynthHand_A dataset for networks A

4.3.2 Training and testing of networks for Depth detection

We trained and tested in parallel 4 architectures, their id's start with B. We have used Adam optimizer with learning rate 0.01 and trained on mini-batches of size 64 from FreiHand dataset for 5719 steps. We trained networks to predicted relative distances between fingers as well as the tilt of the hand as the 22nd parameter. We measured the absolute distance between predicted and ground-truth values of depth for each key point. Also, we measured errors of predictions for relative positions of fingers and hand tilt.

Network AB_PIC was trained for the labeling of sign language letters. We pre-trained UNET architecture for 141800 steps with batch size 16 and used Adam optimizer with a learning rate 0.001. We have used U-Net as a basis on top of which trained depth estimator for 44000 steps with batch size 8 and used Adam optimizer with learning rate 0.001.

Measurement on splitted FreiHand dataset shown in table 4.6 for training data, 4.7 for test data. Tables 4.8 and 4.9 present values of metrics on Synth Hand Dataset for default and augmented data, respectively.

From tables we can see that the usage of additional frames improves the quality of depth estimation. Additional refinement improves quality for all cases except synthetic data without data augmentation. AB_PIC has the best results among architectures for single image analysis.

	B_VID	B_PIC	B_R_VID	B_R_PIC	**AB_PIC**
mse (network output)	1.13E-02	1.83E-02	1.69E-02	1.91E-02	**1.07E-02**
l_1 (network output)	7.29E-02	9.57E-02	9.22E-02	9.97E-02	**7.70E-02**
l_1 (depth)	1.21E-02	1.40E-02	1.45E-02	1.54E-02	**1.04E-02**

TABLE 4.6: Mean errors for networks B on train dataset

	B_VID	B_PIC	B_R_VID	B_R_PIC	**AB_PIC**
mse (network output)	1.75E-02	2.19E-02	2.76E-02	2.08E-02	**1.49E-02**
l_1 (network output)	9.12E-02	1.01E-01	1.15E-01	1.06E-01	**8.92E-02**
l_1 (depth)	1.33E-02	1.55E-02	1.69E-02	1.52E-02	**1.15E-02**

TABLE 4.7: Mean errors for networks B on test dataset

	B_VID	B_PIC	B_R_VID	B_R_PIC	**AB_PIC**
mse (network output)	2.92E-02	1.90E-01	1.87E-02	1.81E-01	**8.63E-02**
l_1 (network output)	1.31E-01	3.51E-01	1.01E-01	3.54E-01	**2.47E-01**
l_1 (depth)	2.05E-02	7.14E-02	1.61E-02	4.10E-02	**2.79E-02**

TABLE 4.8: Mean errors for networks B on SynthHand dataset

	B_VID	B_PIC	B_R_VID	B_R_PIC	**AB_PIC**
mse (network output)	2.45E-02	2.01E-01	2.54E-02	1.63E-01	**8.10E-02**
l_1 (network output)	1.18E-01	3.65E-01	1.17E-01	3.25E-01	**2.24E-01**
l_1 (depth)	1.79E-02	4.32E-02	2.07E-02	3.83E-02	**2.79E-02**

TABLE 4.9: Mean errors for networks B on SynthHand_A dataset

4.3.3 Training and testing of networks for shape parameterisation

At first we tested multilayer perceptrons with various architectures and various losses, but results always were unsatisfying and could be generalized to two scenarios. The first one is the total inability of the network to learn hand 3D parametrization, and second is remembering of average hand. We tested dropout and normalization layers, loss of parameters only, loss of both parameters, and hand shape reconstruction. In the end, we have selected the best performing of our perceptrons and compared its performance with the LSTM network.

'C_MLP' was trained for 1183 steps. Training setting: Adam optimizer, learning rate = 0.01, batch size = 16384. LSTM was trained for 2400 steps with the same learning rate, but batch size 2048.

Results of how well networks preserve the position of 3D key-points after decoding of estimated parameters are shown in Table 4.10. 'C_RNN' network turned out to be more precise in performed tests.

	RNN	MLP
mse	145.25008	286.72867
l_1	5.887265	9.51802

TABLE 4.10: Mean errors for C networks

4.4 Annotation system analysis

We have combined AB_PIC network with proposed RNN for shape parameterization and tested how the addition of shape influences predicted positions of key points. The results of the experiments are shown in Table 4.11.

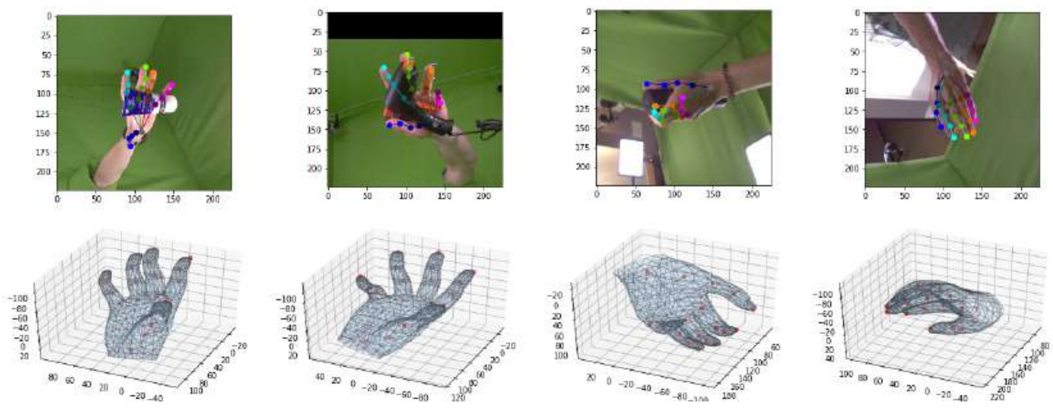
	TRAIN	TEST	SynthHands	SynthHands_A
MSE AB_PIC	267.01346	329.91776	2865.1726	3540.065
l1 AB_PIC	11.493821	12.303339	39.867397	44.1231
MSE MANO	322.8869	360.14267	3814.0867	4494.155
l1 MANO	12.073699	12.662715	46.253853	50.823338
Deviation	4.8%	2.9%	13.8%	13.2%

TABLE 4.11: Intermediate and final 3D key point errors

First two rows inform how far predicted points deviate from the ground truth. We see the same pattern of descending accuracy for unseen data as in tables from 4.2 to 4.9. Rows 3,4 show the accuracy of key-points extracted from the MANO 3D model (Fig 3.3 a)). The final row reflects the impact of adding hand surface on key points position.

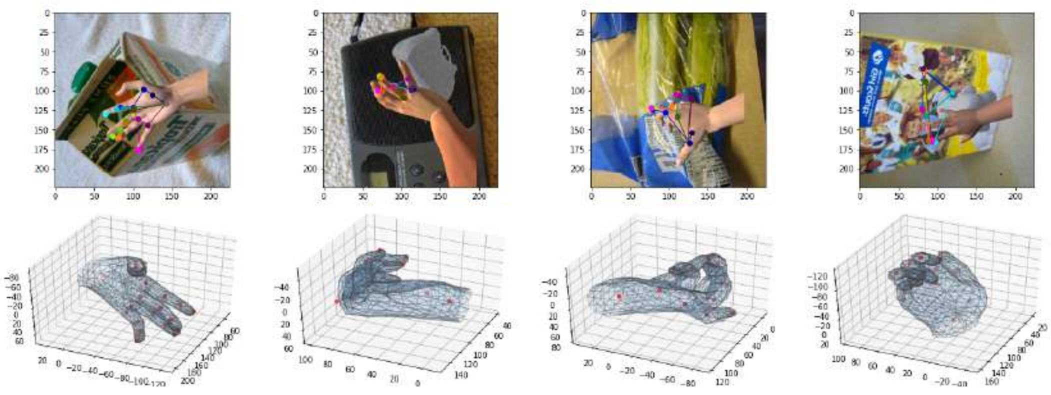
From tables from 4.2 to 4.11 we see that error on SynthHands Dataset always significantly larger. Fig 4.1 to 4.3 show images presented in 3 datasets. On top, we see detected 2D key points and at the bottom predicted 3D shapes. 3D shapes displayed reflected and rotated due to the camera position of rendering software. Predicted shape and points are actually not rotated that way and transformed only by scaling and translation.

FIGURE 4.1: Examples of FreiHand 3D reconstructions



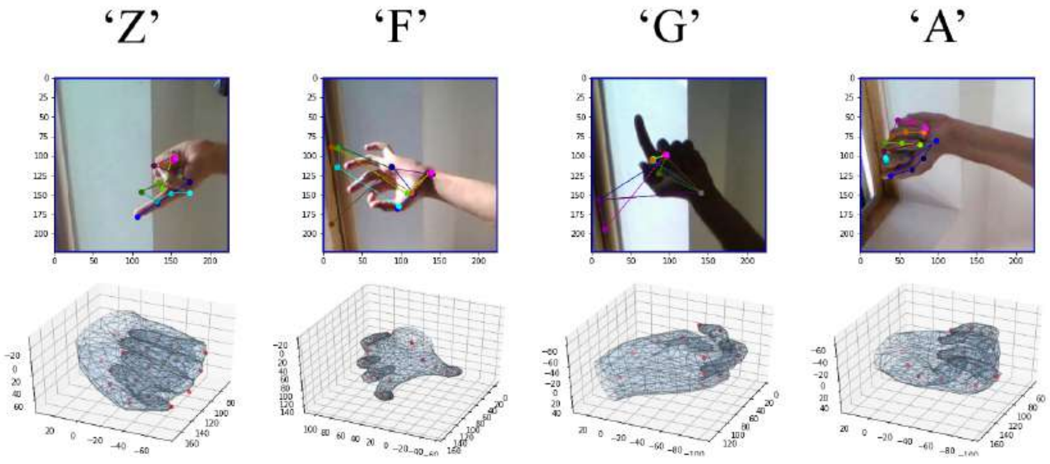
On Fig 4.1 we see accurate detection of key-point on Frei Hand Dataset. Even occluded points predicted on 2D and 3D. For instance, 3rd image from the left at Fig 4.1 show the accurate prediction of occluded little finger, which is seen at reflected reconstruction at the bottom.

FIGURE 4.2: Examples of SynthHands_A 3D reconstruction



In Fig 4.2 we see images from Synth Hand Dataset with an augmented background. The errors obtained on the SynthHands_A dataset was the largest among all tests. Top images show a misdetection of 2D points. Now we can better understand a larger deviation of 3D model from 3D key-points detected by network AB_PIC. Network C_RNN estimates MANO parameters that encode some anatomy by design. The final shape it creates is more realistic than detected 3D points by AB_PIC.

FIGURE 4.3: Examples of ASL 3D reconstruction



On Fig 4.3 we see predicted hand shapes for ASL. Most shapes mismatch originals. Even though in some cases, as for letter 'A' we see that the system can accurately reconstruct sign language pose. It is also a good example of how the final shape can fix misdetected points. 2D Key Points and depth were undetected on an index finger, but the final shape filled them to match the anatomy of the hand.

Chapter 5

Conclusion

Several architectures of neural networks were studied for the purpose of 3D hand shape reconstruction from video. It was shown that the addition of approximated hand positions at network input improves the quality of the final results. Also, it was observed that additional refinement of results is less efficient than the usage of extra annotated frames for both 2D key-point detection and depth estimation on all used datasets.

Also, a modular method for 3D shape estimation was introduced [33]. The method differs from the nearest known analog by RNN, which maps detected key-points to space of shape parameters. Method can be iteratively improved by making changes to certain parts and be adapted for 3D reconstruction from both photo and video sign language dictionaries. Implementation can be changed by the integration of architecture for a particular stage.

The selection of architecture for each stage heavily influences the quality of the reconstructed 3D hand shape. Although modular methods suffer from error accumulation, integration of networks that refine output from previous layers can help to overcome such performance degradation.

Achievements

- Thesis introduces a method for hand shape parametrization.
- Complete 3D hand shape reconstruction method from video sequences was developed, and its performance was studied with different ANN architectures based on UNET, STH, and introduced RNN.

5.1 Future work

The next logical step would be to retrain the system on multiple datasets with data augmentation. After that, it needs to be compared with state of the art methods. Also, currently, the model does not distinguish between left and right hands and can not perform simultaneous detection of left and right hand. The integration of Part Affinity Fields can solve the limitation of single-hand detection [14].

Bibliography

[1] V. Athitsos and S. Sclaroff, “Estimating 3d hand pose from a cluttered image”, *CVPR*, 2003.

[2] X. Chen, G. Wang, H. Guo, and C. Zhang, “Pose guided structured region ensemble network for cascaded hand pose estimation”, *Neurocomputing*, Aug. 2017. DOI: 10.1016/j.neucom.2018.06.097.

[3] M. Marín-Jiménez, F. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, “3d human pose estimation from depth maps using a deep combination of poses”, *Journal of Visual Communication and Image Representation*, vol. 55, Jul. 2018. DOI: 10.1016/j.jvcir.2018.07.010.

[4] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, “Accurate 3d pose estimation from a single depth image”, *IEEE*, 2017.

[5] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, “Real-time continuous pose recovery of human hands using convolutional networks”, *SIGGRAPH*, 2014.

[6] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim, “Latent regression forest: Structured estimation of 3d articulated hand posture”, *CVPR*, 2014.

[7] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, “Cascaded hand pose regression”, *CVPR*, 2015.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, *arXiv*, 2015.

[10] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation”, *arXiv*, 2015.

[11] A. Newell, K. Yang, and J. Deng, “Stacked hourglass networks for human pose estimation”, vol. 9912, pp. 483–499, Oct. 2016. DOI: 10.1007/978-3-319-46484-8_29.

[12] Y. Wu, X. Ruan, Y. Zhang, H. Zhou, S. Du, and G. Wu, “Lightweight architecture for real-time hand pose estimation with deep supervision”, *Symmetry*, 2019.

[13] P. Padeleris, I. Oikonomidis, and A. Argyros, “Using a single rgb frame for real time 3d hand pose estimation in the wild”, pp. 436–445, Mar. 2018. DOI: 10.1109/WACV.2018.00054.

[14] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields”, pp. 1302–1310, Jul. 2017. DOI: 10.1109/CVPR.2017.143.

[15] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, and J. Y. Jianfei Cai, “3d hand shape and pose estimation from a single rgb image”, *CVPR*, 2019.

[16] C. Zimmermann and T. Brox, “Learning to estimate 3d hand pose from single rgb images”, *ICCV*, 2017.

[17] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu, and Q. Yang, "3d hand pose tracking and estimation using stereo matching", *arXiv preprint*, 2016.

[18] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together", *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 245:1–245:17, vol. 36, no. 6, Nov. 2017.

[19] K. Alex, S. Ilya, and E. Hg, "Imagenet classification with deep convolutional neural networks", *Proceedings of NIPS, IEEE, Neural Information Processing System Foundation*, pp. 1097–1105, Jan. 2012.

[20] J. Malik, A. Elhayek, F. Nunnari, K. Varanasi, K. Tamaddon, A. Heloir, and D. Stricker, "Deephps. end-to-end estimation of 3d hand pose and shape by learning from synthetic depth", *3DV*, 2018.

[21] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim, "Big hand 2.2m benchmark: Hand pose data set and state of the art analysis", *CVPR*, 2017.

[22] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim, "First-person hand action benchmark with rgb-d videos and 3d hand pose annotations", *CVPR*, 2018.

[23] C. K. Toby Shar and, D. Robertson, J. Taylor, and J. Shotton, "Accurate, robust, and flexible real-time hand tracking", *CHI*, 2015.

[24] A. Wetzler, R. Slossberg, and R. Kimmel, "Rule of thumb: Deep derotation for improved fingertip detection", *BMVC*, 2015.

[25] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall, "Capturing hands in action using discriminative salient points and physics simulation", *IJCV*, 2016.

[26] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth", *CVPR*, 2014.

[27] F. Mueller, F. Bernard, O. Sotnychenko, S. S. Dushyant Mehta, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular rgb", *CVPR*, 2018.

[28] ——, "Real-time hand tracking under occlusion from an egocentric rgb-d sensor", *ICCV*, 2017.

[29] G. Rogez, J. Supancic, M. Khademi, J. Montiel, and D. Ramanan, "3d hand pose detection in egocentric rgb-d images", Nov. 2014.

[30] F. Gomez-Donoso, S. Orts, and M. Cazorla, "Large-scale multiview 3d hand pose dataset", *Image and Vision Computing*, vol. Accepted, Dec. 2018.

[31] *Home object dataset*. [Online]. Available: `http://www.vision.caltech.edu/pmoreels/Datasets/Home_Objects_06/` (visited on 01/07/2019).

[32] *Asl alphabet*. [Online]. Available: `https://www.kaggle.com/grassknoted/asl-alphabet` (visited on 01/07/2019).

[33] R. Riazantsev, *Master thesis repository*. [Online]. Available: `https://github.com/roman-riazantsev/sign-pose` (visited on 01/07/2019).