Universiteit Leiden
The Netherlands

Opleiding Informatica

3D Hand Pose Estimation
on a Robotic Platform

Alec Flesher-Clark

Supervisors:
Dr. E.M. Bakker
Dr. M.S. Lew

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

26/07/2019

Abstract

Hand tracking and hand pose estimation are crucial components of Human-Robot Interaction (HRI). Recently, scientists have been solving HRI problems by applying Machine Learning and Deep Learning techniques. This thesis contributes to this field of research by implementing an existing Deep Learning method onto a robotic platform. This method consists of a deep network which learns to estimate 3D hand poses from single RGB images.

The implemented deep network consists of three smaller neural networks. The first of these networks aims to segment a hand in a 2D RGB image. The second network receives the cropped output of the first network and then determines a set of 2D hand keypoints. It does this by producing score maps to indicate the likelihood of a certain 2D hand keypoint at a certain location. The third network takes these score maps to estimate the 3D coordinates of the hand in a canonical frame. The three networks together form a larger deep network which is able to predict 3D hand poses from 2D images in a single pass.

Before implementing this network on a robotic platform, the network's accuracy in predicting 3D hand poses is analysed and validated. Slight improvements to the network configuration are proposed which are shown to yield a better accuracy. Finally, a real-time implementation of the model on a robotic platform using a servo actuated camera to simulate a robot's vision is presented. Experiments are conducted using this implementation in order to qualitatively evaluate the real-time performance of the model.

Contents

1 Introduction

1.1 Hand Pose Estimation

Computer vision is a crucial research field for Robotics as it allows robots to have a higher level of understanding the contents of images. As the dependency on large amounts of image data increases every day, the need for robots to interpret images as humans do increases proportionally. Human-robot interaction uses elements of computer vision to enable robots to recognize parts of the human body and interact with them accordingly. Hand movements are an important part in a person's every day interactions with other people and objects. A large number of computer scientists studying human-robot interaction have therefore recently studied the subjects of hand tracking, hand pose estimation and hand gesture recognition.

Within the field of HRI, 3D Hand Pose Estimation is seen as the simulation of the position of parts of the hand in 3D space. In most cases, 3D hand pose estimation is accomplished through estimating the positions of around 14 to 21 unique hand joints in 3D space. Approaches range from using using neural networks for the prediction of the locations of these hand joints to using other Machine Learning statistical models for hand joint prediction. Before the current trend of applying Deep Learning to problems like 3D hand pose estimation, researchers relied on using other Machine Learning techniques such as random forest regression [1] and nearest neighbour search [2]. The nearest neighbour search approach is accurate but requires the subject to wear a colour glove in order to recognize the 3D hand pose. The random forest regression is fast and accurate in hand classification, but its hand pose estimation is outperformed by more recent Deep Learning approaches. Deep Learning's success in research fields such as object segmentation and image localization brought with it a surge in 3D hand pose estimation research, such as the use Voxel-to-Voxel prediction network or a General Adversarial Network (discussed in Section 2). All these different Deep Learning approaches use a neural network which is trained using a large dataset of hands in different poses. Along with these images, the dataset also contains labels for each image containing the image coordinates of the different hand joints. After training the network is able to predict the 3D coordinates of a hand in an image.

In recent years hand pose estimation is implemented using a Deep Learning model where the model consists of several convolutional layers. These models, also referred to as convolutional models, are trained on supervised (labelled) test data containing thousands of images of hands. For models tackling the problem of hand detection an image containing a hand is required. If the model is properly trained it will output the same image but with an indication of the location of the hand in the image, usually by generating a Bounding Box around the hand. These hand detection models are necessary in most hand related HRI problems, as the hand must first be recognized by a robot before it can be further interpreted. For hand pose estimation many researchers such as Wu and Nagahashi [3] use a 2D detection and tracking of the hand and implement a classification model on top to determine the hand's pose.

Although this field has seen many different approaches, all approaches still encounter difficulties due to the fact that hand movements are of a complicated nature and pose major challenges to hand pose estimation methods in real-time. One of these challenges is the problem of self-occlusion

of the hand, when certain fingers or a certain rotation of the hand ensures that a part of the hand is not visible. Another difficulty in hand pose estimation is the difference in speed between a robots pose estimation and the speed of the movement of certain hand gestures. For example, in order to have a robot interpret the complex and quick hand gestures of a sign language, the underlying interpretative model must be fast enough to determine the hand's many poses in real-time.

While implementations like these perform well in a 2D space, they do not properly represent the hand pose in a 3D space. In many cases the problem of 3D hand pose estimation and hand tracking is the requirement of expensive depth-based camera's to capture the input images for the Deep Learning model. These depth-based camera's are able to capture images in which each pixel relates to the distance between the image plane and a corresponding object. While some researchers like Oikonomidis *et al.* [4] have circumvented this problem by using less expensive, and less precise, image and depth capturing setups like Microsoft Kinect's camera; the research done by Zimmerman and Brox [5] makes 3D hand pose estimation possible on regular RGB images, while only requiring the training dataset to contain depth-based images. Once trained, this approach not only has the benefit of being able to work without any special camera equipment, but can also be used outdoors whereas methods relying on depth camera's are not well-suited for outdoor use.

This thesis focuses on the implementation of 3D hand pose estimation on a robotic platform where monocular RGB images are captured frame-by-frame. The main focus of this thesis is therefore to study the model proposed by Zimmerman and Brox and to implement it on a robotic platform. Before implementing the model on a robotic platform, the model will be retrained to validate the accuracy presented in the original paper. After validation, experiments in the model's configuration will be conducted in order to try and improve the accuracy. Finally, a real-time implementation of the model will be proposed and tested in different scenario's. This real-time implementation will be running on a servo actuated camera, which will serve as a robotic platform. The next paragraph serves to outline this thesis' structure.

In Section 2 a more in depth analysis of the related work concerning 3D hand pose estimation will be offered in order to compare other publications to Zimmerman and Brox's method. Section 3 will describe Zimmerman and Brox's model architecture as well as how the accompanying dataset is used by the model for training and evaluating. In Section 4.2 Zimmerman and Brox's model's accuracy is validated. This Section also aims to improve the models accuracy. Therefore, Section 4.2 will include experiments in which the model will be retrained multiple times with adjusted hyperparameters and will be evaluated on multiple datasets. In Section 4.3 Zimmerman and Brox's model will be implemented on a robotic platform. This Section provides an implementation of the model so that 3D hand poses can be estimated in real-time. To test the real-time performance of this implementation, Section 4.3 will contain experiments on the model's performance in different real-time scenario's. Finally, in Section 5, a conclusion of this thesis is given as well as a proposal for possible further research on the subject.

2 Related Work

Before delving into Zimmerman and Brox's method for 3D hand pose estimation, an overview of the previous research on hand pose estimation is provided. Through the years, the field has seen a myriad of different implementations. This section will focus on approaches using neural networks, as the model proposed by Zimmerman and Brox also consists of neural networks. This section is divided into two subsections in order to distinguish between implementations based on depth map images and implementations based on normal RGB images. Both these subsections will consist of important contributions to their respective fields.

2.1 Depth-Based 3D Hand Pose Estimation

One of the first important implementations of Deep Learning networks for 3D Hand Pose Estimation is the DeepPrior [6] network, published by Oberweger *et al.* in 2015. The network consists of two parts. The first part estimates a 3D bounding box around the hand in a depth image. Once the hand is localized through the bounding box, the second part of the network uses this as an input and estimates the 3D joint locations of the hand. DeepPrior stands out from other contemporary methods by the inclusion of a bottleneck layer at the end of the joint estimation network. This layer forces the network to learn a lower dimensional representation of the training data, which enforces the physical constraints of hands on the estimated 3D joint locations. Because of this so called pose prior, the result of the network is a more reliable prediction of a 3D hand pose. Zimmerman and Brox have incorporated a similar pose prior at the end of their network.
Although the original DeepPrior network was quickly outperformed by other Deep Learning based methods, the model saw a resurgence when Oberweger introduced DeepPrior++ [7] in 2017. The addition of ResNet [8] layers, data augmentation and improvements to hand localization resulted in a network that could compete with and even outperform many other networks within the field of 3D hand pose estimation.

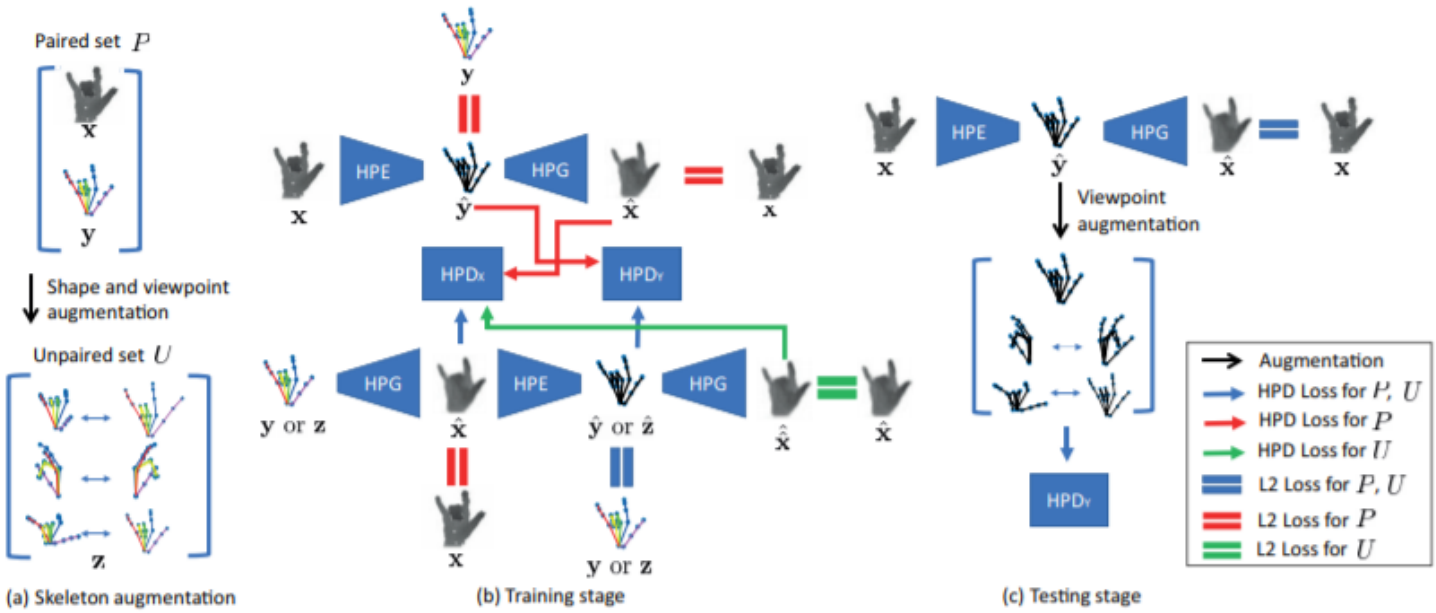


Figure 1: Overview of the model created by Baek *et al.* [9]

Baek *et al.* [9] made use of a Generated Adversarial Network or GAN network [10] to translate a depth map of a hand to a 3D representation of hand joints. The two main components of GAN networks are a discriminator and a generator, both two convolutional neural networks, which compete with each other. The discriminator is trained to classify images as fake or real whereas the generator is a network which generates fake images using a random initialization. The end goal of a GAN network is for the generator to generate images which cannot be distinguished as fake by the discriminator. Baek *et al.* made use of a specific GAN network called CyclicGAN [11], consisting of a Hand Pose Generator (HPG), a Hand Pose Estimator (HPE) and a Hand Pose Discriminator (HPD). The HPE would generate the 3D hand pose based on an input depth map. Based on this hand pose, the HPG would then generate a hand and the HPD would serve as a loss function for optimizing the consistency of both the HPG and HPE. The entire process of the CyclicGAN is depicted in Figure 1 and originates from the paper by Baek *et al.*

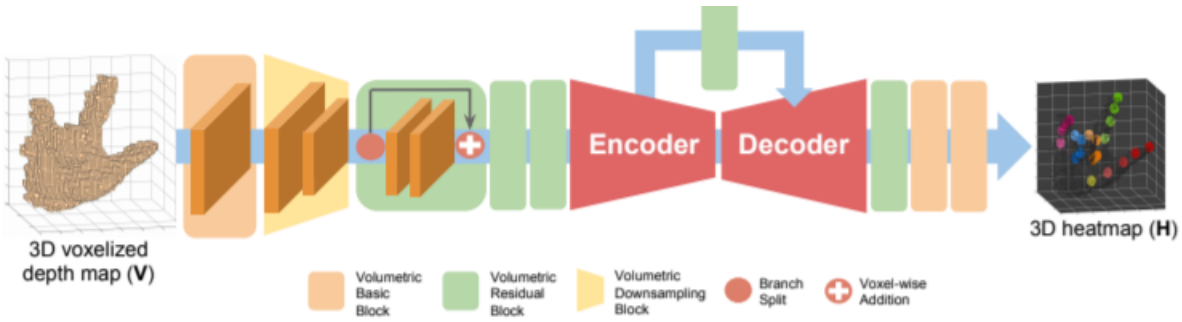


Figure 2: The Architecture of the V2V-PoseNet made by Moon *et al.*, diagram originally found in the paper [12]

A problem of the methods developed by Oberweger *et al.* and Baek *et al.* is that their models require a 2D depth image as input. The pixel values in a 2D depth map indicate the physical distances of certain object points from the depth camera. Due to this fact the depth map essentially consists of 3D data. Most depth-based approaches only use the depth maps as a 2D image for input, which can result in a distortion of the shape of the hand in 3D space by projecting it to 2D image space. Another problem of these approaches is "the highly non-linear mapping between the depth map and the 3D coordinates" [12]. This mapping makes the learning procedure more difficult while it also prohibits the network from making a precise estimation of the coordinates of keypoints.
Moon et al. [12] offer a solution to these problems with their Voxel-to-Voxel prediction network. Voxels are volumetric pixels or the 3D equivalent of pixels. This network takes a voxelized grid and estimates the likelihood of each hand keypoint per voxel. In order to achieve this they used a 3D convolutional network instead of the usual 2D convolutional networks used by other publications within the field of 3D hand pose estimation [5][6] [11] [12]. The general architecture of the V2V-PoseNet can be seen below. Moon *et al.* compare V2V-PoseNet compared with many other depth-based 3D hand pose estimation models which were considered state-of-the-art at that time, including DeepPrior and DeepPrior++. Moon *et al.* report that V2V-PoseNet achieves a Mean error (in mm) of 6.28 when evaluated on the ICVL Hand Posture Dataset, whereas DeepPrior achieves a Mean error of 10.4 and DeepPrior++ a Mean error of 8.1. From the results it is clear to see that the Deep Learning network proposed by Moon *et al.* is considered to be one of the most accurate implementations when it comes to 3D hand pose estimation on depth-based images.

2.2 Image-Based 3D Hand Pose Estimation

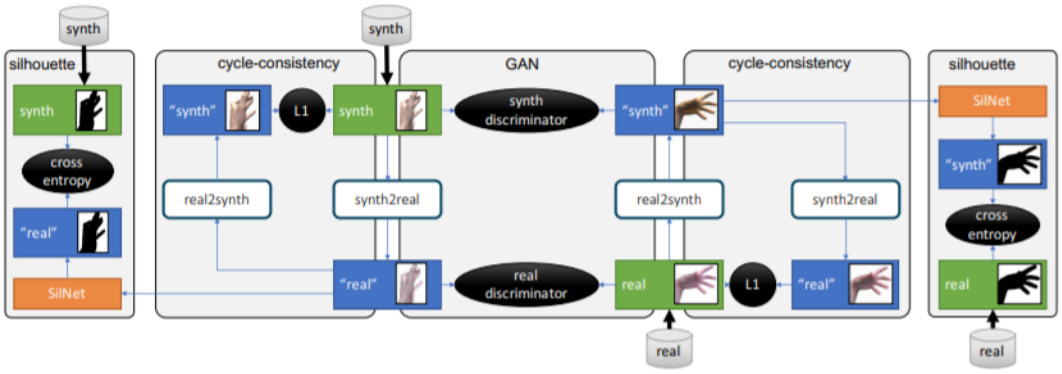


Figure 3: GeoConGAN architecture, diagram originating in [13]. Blue boxes indicate images generated by the network and images from the dataset are indicated by green boxes

The field of Image-Based 3D Hand Pose Estimation is still in its infancy. The method proposed by Zimmerman and Brox [5] was the first in this field. However, during the two years prior to the publication of this thesis there have been slight improvements on Zimmerman and Brox's method. In 2018, Mueller *et al.* built on top of Zimmerman and Brox's method by introducing a large synthetically generated dataset with automatically generated labels [13]. To translate these synthetic images to real images, Mueller *et al.* use a conditioned GAN [11] called GeoConGAN. The GeoConGAN consists two smaller GAN's called synth2real and real2synth. Here, synth2real is trained to learn the mappings from synthetically generated images to real images and real2synth the other way around. The advantage of this structure is that it is not required to have a real counterpart for a synthetically generated image or vice versa. The SilNet is a small binary classification network which serves to extract the silhouette from the images generated by real2synth and synth2real. Once the GeoConGAN is trained, all synthetically generated images from the dataset are fed through it to obtain their "real" counterparts as well as their associated ground truth 3D joint locations.

For the 3D hand pose estimation Mueller *et al.* use a convolutional neural network based on ResNet [8] called the RegNet, which estimates the 2D and 3D positions of 21 hand joints. This network, unlike Zimmerman and Brox's network, expects a cropped image of a hand and produces a heatmap for the 2D joint positions in image space. For the 3D positions of the hand joints, these are represented as the 3D coordinates relative to the root joint in the palm of the hand. After both the 2D and 3D positions are found by the RegNet, a kinematic skeleton containing these positions is fit onto the hand in the image.
This approach generalizes better than Zimmerman and Brox's approach due to the large synthetically generated dataset. It is also more accurate, as seen in the experiments provided in the paper [13]. However, this paper focuses on Zimmerman and Brox's approach as it translates better to a robotic platform due to its built-in hand segmentation model, which is absent in methods like that of Mueller *et al.* Whereas the hand pose estimation model proposed by Mueller *et al.* requires an already cropped image of a hand as an input, Zimmerman and Brox's uses a segmentation model to localize hands in images. It would be possible to implement hand segmentation in the

model proposed by Mueller *et al.* and would be an interesting topic for further research. After successfully combining hand segmentation with Mueller *et al.*'s model, the combined models could be implemented on a robotic platform. Unfortunately, Mueller *et al.* have given limited access to their model's code and therefore this thesis has focused solely on implementing Zimmerman and Brox's approach on a robotic platform.

3 Zimmerman and Brox's Network

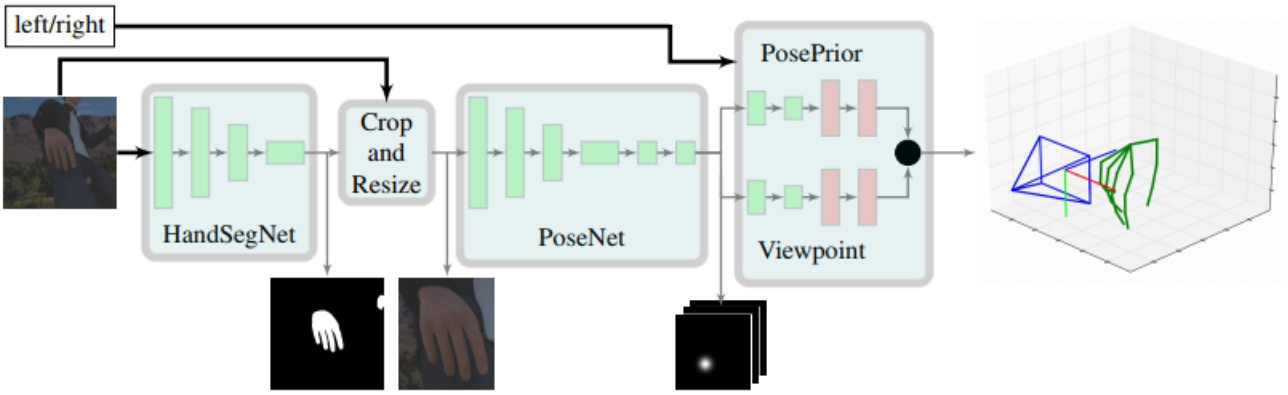


Figure 4: Architecture of model proposed by Zimmerman and Brox [5]

Zimmerman and Brox's network is split into three smaller networks which are connected to each other. A single pass through the entire network enables a hand to be localized and segmented through HandSegNet (see Section 3.2). Afterwards the hand keypoints are determined as score maps through PoseNet (see Section 3.3) and finally, the 3D structure of the hand is translated from these score maps by the PosePrior network (see Section 3.4). Before offering a detailed description of each smaller network, an elaboration will be given on the dataset used for training and evaluating each network.

3.1 The Rendered Handpose Dataset

Zimmerman and Brox introduce a new dataset specifically intended for training models to estimate 3D hand poses on RGB images: the Rendered Handpose Dataset. This dataset contains 41258 training and 2728 testing samples. All images in the dataset are synthetically generated because manually labelling 3D data often results in inaccurate labels. Each sample in the dataset contains the following:

- An RGB image: a monocular, synthetically generated image of 320x320 pixels
- A depth map corresponding to the RGB image. For training models that require depth-images as input, the depth map would be used. Since Zimmerman and Brox's model uses RGB images, the depth maps in the dataset are not used.
- Segmentation masks for the following classes: background, person and three classes for each finger and one class for each palm. See the following section 3.2 for the implementation of the segmentation masks
- A 3D kinematic model of the hand containing the 21 keypoints per hand. Here, each finger contains 4 keypoints for the joints of the finger. Finally, the keypoint close to the wrist acts as a rotation point. Each keypoint is indicated through texture coordinates for the RGB image,

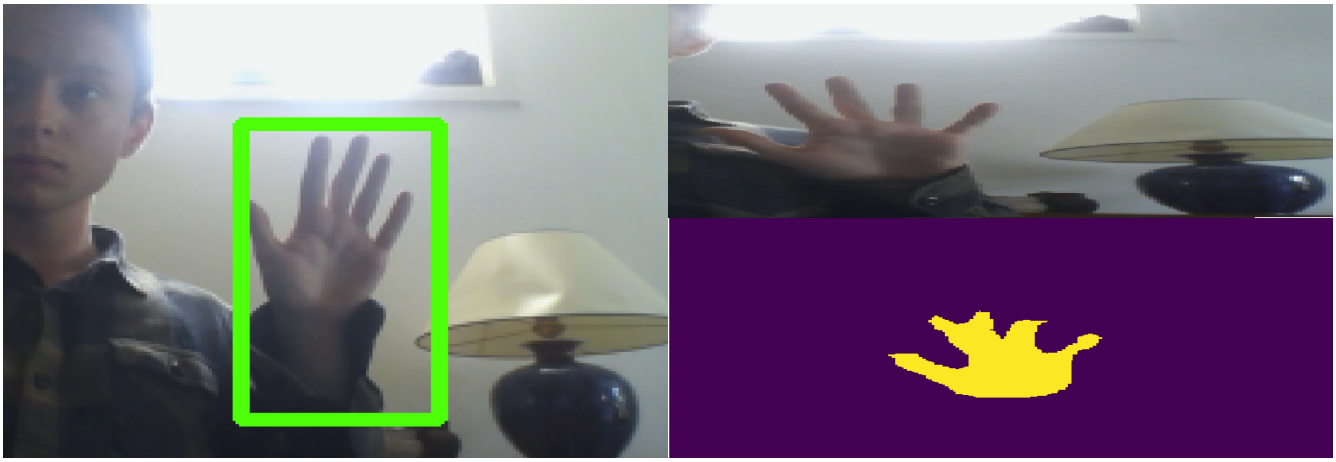
xyz coordinates in the world frame along with an indicator to tell if the keypoint is visible in the image or if it is occluded. For more information on how these coordinates are used, see Section 3.3

- An intrinsic camera matrix K which includes the focal length, optical centre and skew coefficient of the image. Like the depth map, this matrix is not used for the training of the models.

In order to validate the model's accuracy, the dataset must be split in the same way as in the original paper. Therefore, the dataset is split into a validation set and a training set. The training set contains images of 16 different characters performing 31 actions, whereas the validation set contains 4 characters performing 8 different actions.

3.2 HandSegNet

Before determining the 3D coordinates of the hand in an image, the hand must be localized and cropped before being fed through the PoseNet. The localization network Zimmerman and Brox use is called HandSegNet. This convolutional network is based on a paper on person detection written by Wei *et al.* [14]. HandSegNet is trained to estimate a 256x256 hand mask, as depicted in Figure 5b.



(a) (b)

Figure 5: Comparison of a Bounding Box Hand Detection approach (originally created by Victor Dibia [15]) with Zimmerman and Brox's hand detection using HandSegNet.
(a) Hand Segmentation using a Single Shot Multibox Detector [16].
(b) Hand Segmentation using Zimmerman and Brox's HandSegNet

As images provided by the Rendered Handpose Dataset are 320x320 pixels, every image is cropped randomly to provide a 256x256 image as input to the network. The network consists of convolutional layers with ReLu activation functions interspersed with max pooling layers to down sample the 3D image tensors. After passing through all the convolutional and max pooling layers, the 32x32x2 image tensor is transformed to a 256x256x2 tensor through a bilinear upsampling layer. The last layer, the Argmax layer, highlights the hand and produces the hand mask as a 256x256x1 tensor. The network is initialized with the same weights as the person detector made by Wei *et al.* [14] and

is trained using softmax cross-entropy loss. The weights are updated using the Adam optimization algorithm [17].

3.3 PoseNet

The hand can be interpreted as $J = 21$ joints, or keypoints, connected together. PoseNet requires the input image to have all the keypoints of a single hand in the image, so careful attention must be given to the cropping procedure before the cropped image is fed through PoseNet. The original image is cropped by estimating the centre of a bounding box using the hand mask produced by HandSegNet. Afterwards, noise is added to the centre of the bounding box by sampling a zero mean distribution with a variance of 10 pixels. Once the original input image is cropped accordingly, the resulting image is given as input to PoseNet. Posenet is a convolutional neural network that estimates $J = 21$ 2D score maps $c = \{c_1(u,v), \ldots, c_J(u,v)\}$. Each score map $c_i \in \mathbb{R}^{N \times M}$ shows the likelihood for a single keypoint being present in a certain spatial location. Figure 6 shows a sketch of the keypoints of the hand and the score maps (in the figure called confidence maps) being output by a 2D pose estimation network.

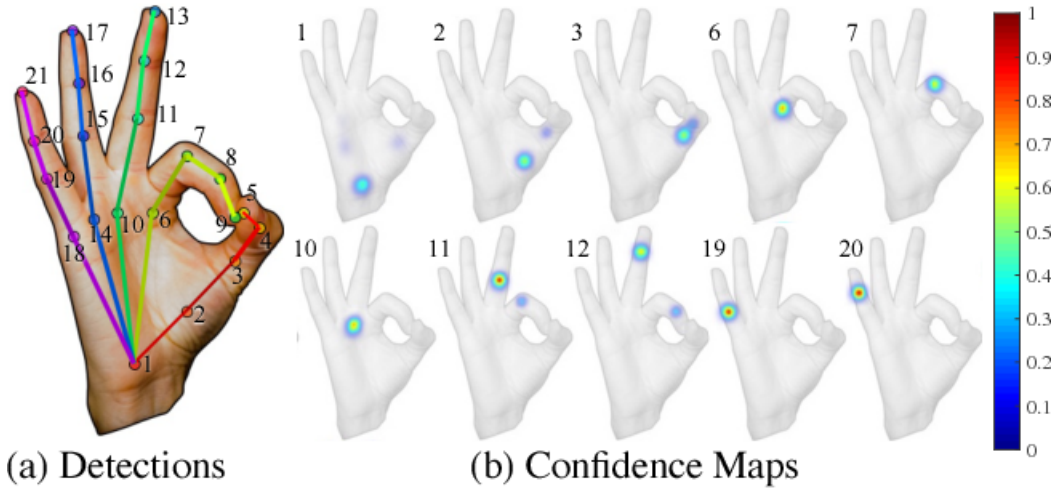


Figure 6: Figure showing keypoint detection as confidence (score) maps. Figure originally from paper by Simon *et al*[18].

The structure of PoseNet is based on an encoder-decoder structure found in the paper by Wei *et al* [14]. The encoder serves to output a feature representation of the cropped image which the decoder interprets to find the score maps. The encoder component consists of multiple convolutional layers with 3×3 filters and a couple of max pooling layers for down sampling. The decoder consists of and convolutional layers with 3×3 and 7×7 layers. In three interspersed convolutional layers, the network outputs 21 32×32 score maps, one for each keypoint.
Zimmerman and Brox trained the network using a batch size of 8 and an L_2 loss and is initialized by the weights originally from the Wei *et al* paper. These weights are updated per iteration using the Adam optimizer.

3.4 PosePrior

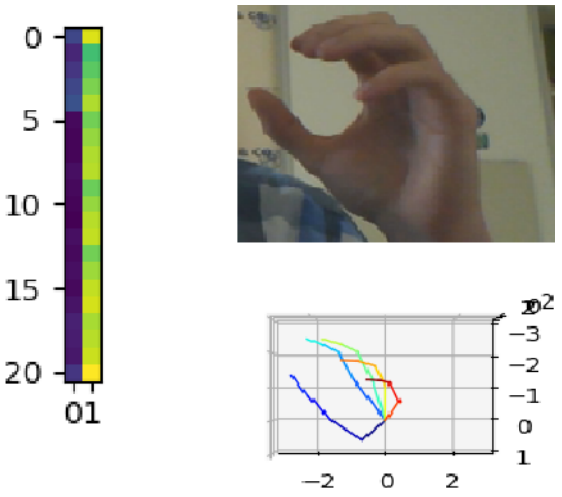


Figure 7: Original image of hand compared to visualization of 2D scoremaps (left) and visualization of 3D coordinates (bottom right)

Using the score maps $c(u,v)$ generated by PoseNet, a 3D configuration is estimated by PosePrior. The 3D hand pose is determined in terms of 3D coordinates $w_i = (x_i, y_i, z_i)$ which represent each of the 21 keypoints in 3D space. This network does not predict the absolute 3D coordinates of the keypoints, but rather predicts a translation invariant representation of hand poses, denoted as w^{rel}. Here, w^{rel} represent the relative normalized 3D coordinates. To calculate w^{rel}, PosePrior needs to predict the 3D coordinates in the canonical frame, w^c, as well as a rotation matrix $R(w^{rel}) \in \mathbb{R}^{3\times3}$. This rotation matrix serves to transform coordinates from the original frame to the canonical frame. All formulae in this section are from the original paper by Zimmerman and Brox [5]. The relation between w^{rel}, $R(w^{rel})$ and w^c is shown in the following formula. Here, w^{c^*} represents a certain keypoint in the canonical frame:

$$w^{c^*} = R(w^{rel}) \cdot w^{rel} \tag{1}$$

This formula can be rewritten to calculate w^{rel}:

$$w^{rel} = w^c \cdot R^\intercal \tag{2}$$

Because the right and left hand share a symmetric relation, right hands are flipped along the z-axis. Therefore, the canonical coordinate system w_i^c can be represented in the following manner:

$$w_i^c = \begin{cases} (x_i^{c^*}, y_i^{c^*}, z_i^{c^*})^\intercal & \text{if it is a left hand} \\ (x_i^{c^*}, y_i^{c^*}, -z_i^{c^*})^\intercal & \text{if it is a right hand} \end{cases}$$

3.4.1 Estimating Canonical Coordinates and Calculating the Rotation Matrix

Estimating both the canonical coordinates and the rotation matrix requires the PosePrior network to be split into two symmetric streams. Both of the streams contain a similar structure of six convolutional layers with 3×3 filters. Afterwards, a concat layer is added to indicate to the stream

if it is processing images of left hands or right hands. After concatenation, both streams contain two fully connected-layers to process the image further. The final layer of the two streams is also a fully-connected layer, which outputs the canonical coordinates w^c for the first stream and the rotation matrix R for the second stream. The rotation matrix is called the viewpoint, where said viewpoint is associated with "a certain sample with respect to the canonical frame" (Zimmerman and Brox, page 4 [5]).
To calculate the rotation matrix precisely, the viewpoint stream finds the rotations R_{xz} and R_y for a certain keypoint $w_a^{c^*}$. The rotation R_{xz} aligns $w_a^{c^*}$ with the y-axis of the canonical frame via the following formula:

$$R_{xz} \cdot w_a^{c^*} = \lambda \cdot (0, 1, 0)^T \tag{3}$$

Here, $\lambda \geq 0$ for a certain keypoint with index a. After calculating R_{xz} using the formula above, R_y is determined using the following equation:

$$R_y \cdot R_{xz} \cdot w_o^{c^*} = (\eta, \zeta, 0) \tag{4}$$

η, like λ before, is greater than or equal to 0 for a keypoint with index o. ζ represents the alignment with the y-axis done in Equation 3. Now that both R_{xz} and R_y are calculated, the total transformation between the canonical and original frame can be represented by:

$$R(w^{rel}) = R_y \cdot R_{xz} \tag{5}$$

The two streams require two different loss functions for training. The stream that predicts the canonical coordinates w^c uses a squared L_2 loss function. This loss function uses the predicted canonical coordinates w_{pred}^c and the ground truth canonical coordinates w_{gt}^c:

$$L_c = \parallel w_{gt}^c - w_{pred}^c \parallel_2^2 \tag{6}$$

The stream that predicts the rotation matrix uses a similar L_2 loss function with predicted and ground truth rotation matrices:

$$L_r = \parallel R_{gt} - R_{pred} \parallel_2^2 \tag{7}$$

Like the other components of Zimmerman and Brox's model, the PosePrior network uses an Adam solver for optimizing the weights during training.

4 Experiments

In this section, Zimmerman and Brox's model will be evaluated on two fronts: accuracy and real-time performance. For evaluating accuracy, all models discussed in the previous section (3) will be retrained and evaluated to see if the results of the paper can be reproduced. Afterwards, the models will be retrained using different hyperparameters to see if an improvement of accuracy can be achieved. Because Zimmerman and Brox's model is implemented onto a robotic platform, the second part of the experiments uses an implementation of the model on robotic platforms and is focused on the qualitative evaluation of the model's performance in real-time scenario's.

4.1 Experiment Setup

Component	Machine Specification
Processor	Intel Core i7-5820K 3.3 GHz 12 cores
Storage	64Gb RAM, 1Tb Storage
Graphics Card	Nvidia Geforce Titan X 12 GByte
Operating System	Ubuntu 16.04 LTS 64-bit

Table 1: Machine Specifications of the machine used for training and evaluation in the experiments of Section 4.2

In Table 1, the specifications of the machine used for training and evaluation is given. This machine is used exclusively for evaluating the accuracy of Zimmerman and Brox's model. The models are implemented using Tensorflow-GPU [19].

Component	Machine Specification
Processor	Intel Core i7-7700HQ CPU 2.8 GHz
Storage	16Gb RAM, 500Gb Storage
Graphics Card	Nvidia GTX 1050 X 4 GByte
Operating System	Windows 10 64-bit

Table 2: Machine Specifications of the machine used for the experiments described in Section 4.3

The machine specified in Table 2 was used to determine the real-time performance of Zimmerman and Brox's model. This machine was used together with the Pololu Maestro camera to simulate a robot's vision and to detect 3D hand poses in real-time by feeding video frames through Zimmerman and Brox's model.

4.2 Evaluating Accuracy

As seen in Section 3, Zimmerman and Brox's model is split into three sub-models. In order to reproduce the accuracy of the original paper, all three sub-models must be retrained and evaluated. This section will start by validating the accuracy of HandSegNet and PoseNet when detecting 2D keypoints in RGB images. After obtaining results similar to the ones reported in the original paper, experiments with HandSegNet and PoseNet will be conducted in order to improve the accuracy of the models. Afterwards, different methods for translating 2D keypoints to 3D coordinates will be compared. Among these methods are the bottleneck method by Oberweger et al. [6] and PoseNet (3.4) by Zimmerman and Brox. Finally, the versions of HandSegNet, PoseNet and 3D lifting network that yield the best accuracy will be combined and evaluated on both the Rendered Handpose Dataset and the Stereo Tracking Benchmark Dataset [20]. This combination of the three models will yield the best found accuracy for 3D hand pose estimation.

The accuracy of the models of Zimmerman and Brox's network as well as that of the different 3D lifting techniques is described through three values (also used in [5]). The average mean endpoint error (EPE Mean) as well as the average median endpoint error (EPE Median) represent the Euclidian distance in pixels between a predicted keypoint location and the ground truth location. The last evaluation value is the Area Under the Curve (AUC) on the percentage of correct keypoints (PCK) over certain error thresholds (thresholds described in pixels). The lower the EPE Median and EPE Mean are and the higher the AUC is, the better.

4.2.1 Evaluation of 2D Keypoint Detection

Table 3 shows accuracy of different configurations of HandSegNet and PoseNet when evaluated on the RHD Dataset. The same accuracy as in the original paper has been achieved, shown in the table in the top row. In order to properly understand the results, the training configurations for HandSegNet and PoseNet from the original paper must be elaborated. HandSegNet was trained for 40000 iterations with a batch size of 8. The learning rate of HandSegNet was $1e-5$ for the first 20000 iterations, $1e-6$ for the next 10000 iterations and $1e-7$ for the last 10000 iterations. PoseNet was trained for 30000 iterations, also with a batch size of 8. PoseNet's learning rate was $1e-4$ for the first 10000 iterations, $1e-5$ for the next 10000 iterations and $1e-6$ for the final 10000 iterations. The accuracy of the original configuration (used in Zimmerman and Brox's paper) of HandSegNet and PoseNet is shown in the top row of Table 3, here called Paper Configuration. To try to improve the accuracy of HandSegNet and PoseNet, changes have been made to the batch size and learning rate of the two models. Both HandSegNet and PoseNet have been retrained and evaluated using a batch size of 16. Because a higher batch size than 16 results in memory leaks of the GPU, the batch size hasn't been incremented further. Afterwards, the learning rate of HandSegNet has been updated to $1e-6$, $1e-7$ and $1e-8$. Like in the original paper, the learning rate updates after the same amount of iterations (after 20000, then after another 10000 iterations). The learning rate of PoseNet has been updated to $1e-5$, $1e-6$ and $1e-7$ per 10000 iterations.

In Table 3, the row indicated by Paper Configuration shows the results of evaluating the configuration used in the original paper. The following rows show the results for different combinations of retrained versions of HandSegNet and PoseNet. Below is a short summary of the different model configurations:

HandSegNet: Original and PoseNet: Original indicate that the original training configuration of that model was used. The Paper Configuration can be seen as a combination of HandSegNet: Original and PoseNet: Original.
HandSegNet: Batch 16 and PoseNet: Batch 16 indicate that the model was retrained using a batch size of 16.
HandSegNet: Learning Rate - 1 indicates that a configuration of HandSegNet was used with the learning rate updated to $1e-6$ for the first 20000 iterations, $1e-7$ for 10000 iterations and $1e-8$ for the last 10000 iterations.
PoseNet: Learning Rate - 1 implies a configuration of PoseNet with an updated learning rate of $1e-5$, $1e-6$ and $1e-7$ per 10000 iterations.

Model Configuration	EPE Mean	EPE Median	AUC
Paper Configuration	15.478	4.341	0.720
HandSegNet: Batch Size 16 PoseNet: Original	15.814	4.345	0.721
HandSegNet: Original PoseNet: Batch Size 16	**15.297**	**4.196**	**0.726**
HandSegNet: Batch Size 16 PoseNet: Batch Size 16	15.689	4.231	0.726
HandSegNet: Learning Rate - 1 PoseNet: Original	17.730	4.769	0.695
HandSegNet: Original PoseNet: Learning Rate - 1	18.249	6.632	0.640
HandSegNet: Learning Rate - 1 PoseNet: Learning Rate - 1	20.848	7.325	0.613

Table 3: Results for evaluating HandSegNet and PoseNet on 2728 shuffled images of the Rendered Handpose Dataset.

From the table it is clear to see that models retrained with different learning rates performed poorer than the original configuration and models with higher batch sizes performed better. In particular, having a batch size of 16 when training PoseNet yielded the best results in accuracy. PoseNet being the most important component of 2D keypoint detection explains why changes to this model significantly influences the end accuracy of HandSegNet and PoseNet combined together.

4.2.2 Evaluation of 3D Lifting Techniques

First, the different methods which lift 2D keypoints to the relative 3D coordinates w^{rel} are validated in order to prove that PosePrior has the overall highest accuracy. A more detailed explanation of these methods is found in Zimmerman and Brox's paper [5], but a short summary will be provided along with the results of the evaluation:

The direct approach attempts to lift 2D hand keypoints to w^{rel} directly without using a canonical frame. Because the network now has to learn the rotation of the hand separately, as the network does not include a rotation matrix $R(w^{rel})$, the network performs poorly.
The Bottleneck approach adds a fully-connected layer at the end of the network, which is parametrized as in Oberweger *et al's* paper on DeepPrior [6]. As seen in the table above, this method performs poorer than the direct approach.
Both the local approach and local approach with xyz loss incorporate the kinematic model of the hand in it's network. The network estimates articulation parameters of the kinematic model such as the angles between finger joints and bone length.

The results of the previous evaluation on HandSegNet + PoseNet indicate that making the learning rate smaller negatively impacted the accuracy of the model. Therefore, only the batch size was experimented with while retraining the different lifting methods. All methods were trained twice, once with a batch size of 8 and once with a batch size of 16. The models were trained for 10000 iterations with a learning rate of $1e-5$, as the difference in accuracy between PosePrior and other techniques is quickly visible.

Lifting Method with Batch Size	EPE Mean in mm	EPE Median in mm	AUC
PosePrior: Batch Size 8	23.761	**20.213**	**0.563**
PosePrior: Batch Size 16	**24.367**	20.460	0.558
Direct: Batch Size 8	33.290	31.761	0.393
Direct: Batch Size 16	32.444	30.603	0.412
Bottleneck: Batch Size 8	36.863	35.642	0.344
Bottleneck: Batch Size 16	35.409	33.745	0.361
Local: Batch Size 8	65.327	58.795	0.230
Local: Batch Size 16	61.262	53.917	0.250
Local_xyz: Batch Size 8	33.808	31.804	0.388
Local_xyz: Batch Size 16	33.142	31.163	0.398

Table 4: Evaluation of the different 2D to 3D lifting methods. Here, the EPE Mean and EPE Median are shown in mm instead of pixels.

Because PosePrior performs the best out of all lifting methods, this model was retrained for 80000 iterations as this was the amount of training iterations in the original paper. The learning rate while training was $1e-5$ for the first 60000 iterations and $1e-6$ for the next 20000 iterations. The results of different batch sizes on this configuration is visible in the Table 5.

Batch Size	EPE Mean in mm	EPE Median in mm	AUC
PosePrior: Batch Size 8	**22.308**	**18.928**	**0.585**
PosePrior: Batch Size 16	23.340	19.875	0.568

Table 5: Evaluation of different batch sizes on the PosePrior network.

From the results in the table it is clear that PosePrior with batch size 8 is the best configuration for lifting 2D keypoints to w^{rel}. As seen in Table 6, the retrained PosePrior with batch size 8 achieves an accuracy similar to that reported in the original paper.

From all experiments, a configuration of HandSegNet, PoseNet and PosePrior which yields the highest accuracy is presented. The most accurate configuration for HandSegNet and PoseNet (found in Section 4.2.1) is compared to the results reported in the original paper by Zimmerman and Brox [5] in Table 6. The most accurate configuration of PosePrior is compared to the results of the original paper in Table 7. Both results are based on solely evaluating the models on the Rendered Handpose Dataset.

HandSegNet and PoseNet Configuration	EPE Mean in pixels	EPE Median in pixels	AUC
Original Paper Results for HandSegNet + PoseNet	18.741	6.745	0.635
HandSegNet: Original PoseNet: Batch Size 16	**15.297**	**4.196**	**0.726**

Table 6: Comparison of most accurate HandSegNet and PoseNet configuration found with the results reported in the original paper.

HandSegNet and PoseNet Configuration	EPE Mean in mm	EPE Median in mm	AUC
Original Paper Results for PosePrior	22.433	**18.841**	0.585
PosePrior: Batch Size 8	**22.308**	18.928	**0.585**

Table 7: Comparison of most accurate PosePrior configuration found with the results reported in the original paper.

The results of fully evaluating the configuration with the highest accuracy on both Rendered Handpose Dataset and the Stereo Tracking Benchmark Dataset is shown in the graph below:

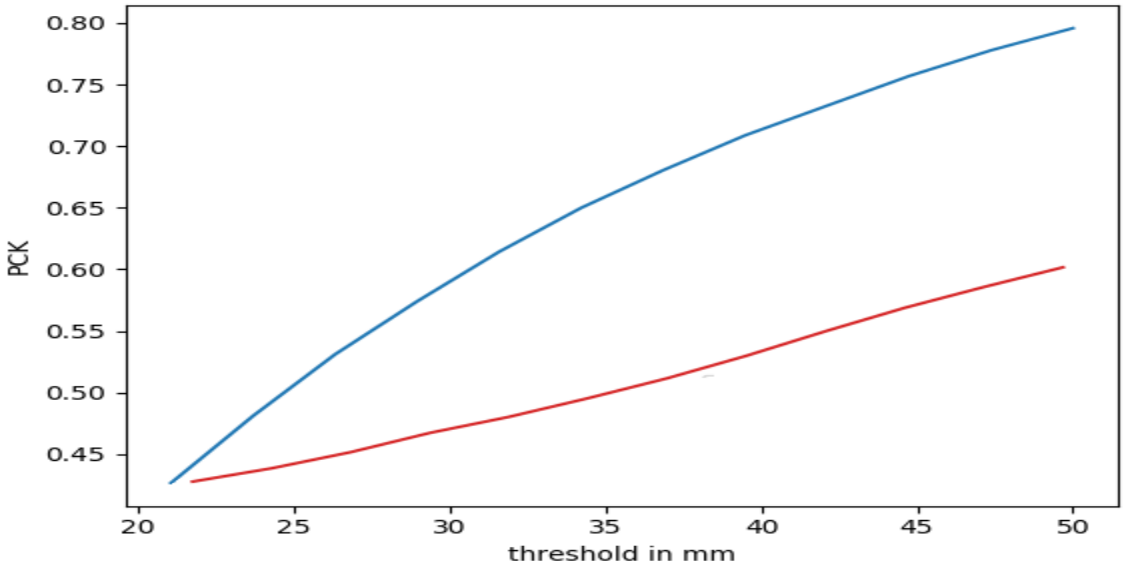


Figure 8: Evaluation of the entire configuration on the Rendered Handpose Dataset (blue line) and the Stereo Tracking Benchmark Dataset (red line).

Figure 8 shows an evaluation of the configuration proposed in Tables 6 and 7 on the Rendered Handpose Dataset with an AUC of 0.647 and on the Stereo Tracking Benchmark Dataset with an AUC of 0.287. In Zimmerman and Brox's paper, the poor accuracy on the Stereo Tracking Benchmark Dataset is resolved through partly training the model on the benchmark dataset before evaluating. Because the real-time implementation of the model does yield accurate hand pose predictions (as seen in Section 4.3), the proposed configuration trained solely on the Rendered Hand Pose Dataset is sufficient on a robotic platform. In further research on the accuracy of this model it would be interesting to see how training the model on different datasets affects the accuracy.

4.3 Evaluating Real-Time Performance

To test what the real-time performance of this configuration would be when it is implemented on a robotic platform, the model was implemented to detect 3D hand poses through the camera of a servo actuated camera. Before discussing the real-time implementation of Zimmerman and Brox's model, a definition of real-time must be given. Real-time is seen as a level of a program's responsiveness that the user of the program senses as sufficiently immediate. In the case of Zimmerman and Brox's model, 3D hand pose estimation is performed in real-time if the model is able to keep up with the user's hand movements and other moving objects in the camera frame. In this section, the overall speed of this implementation will be discussed. This section will also analyse the results of running the model in different real-time environments.

Both the original implementation and the multithreading variant use OpenCV [21] to capture the servo actuated camera's video input frame by frame. Running original implementation results in the model estimating 3D hand poses with a speed of around 2 frames per second while the multithreading variant would achieve around 5 frames per second. The detection speed of the model could be increased when running the programme on a machine with a better GPU than the machine used for this paper 2.

The multithreading implementation along with the configurations shown in Tables 6 and 7 were used in three different environments. In all environments the Pololu Maestro camera would pan from left to right while person in view would make three hand gestures with his left hand. The person would make an open hand with the palm towards the camera, then a fist with his left thumb covering his index and middle finger and finally a circle with his index finger and thumb. In all scenario's, the subject was at a distance of 1.5 meters from the camera. Distances farther than 1.5 meters would seriously impair the real-time accuracy of the model. Before showing the results, a description of each scenario is given:

- Scenario (a): The subject is standing before a blue background, with the servo actuated camera standing on a couple of books on a table. This way the camera could capture the subject's entire body. This setup intends to show the model's performance in a scenario where the subject is clearly visible and the subject's hand is distinguishable from the background. This setup should results in the best performance of the model.
- Scenario (b): The subject is standing in a classroom with a yellow wall as a background. Again the camera is elevated on books on a table. This setup is chosen to measure the model's performance in situations where the frames captured by the camera contain a lot of objects. Also a yellow background is chosen as now the hand in each frame is less distinguishable compared to the blue background in scenario (a).
- Scenario (c): The subject is standing outside in a garden in front of a tree. The camera is positioned in the same way as scenario's (a) and (b) to capture the full body of the subject. This scenario will measure the model's performance in situations where there are many moving objects in the frame, as in this scenario the wind causes the tree in the background to move. This scenario also intends to measure the model's performance it is implemented on a robotic platform that operates outdoors. Scenario's (a), (b) and (c) are visible in the rows of Figure 9.

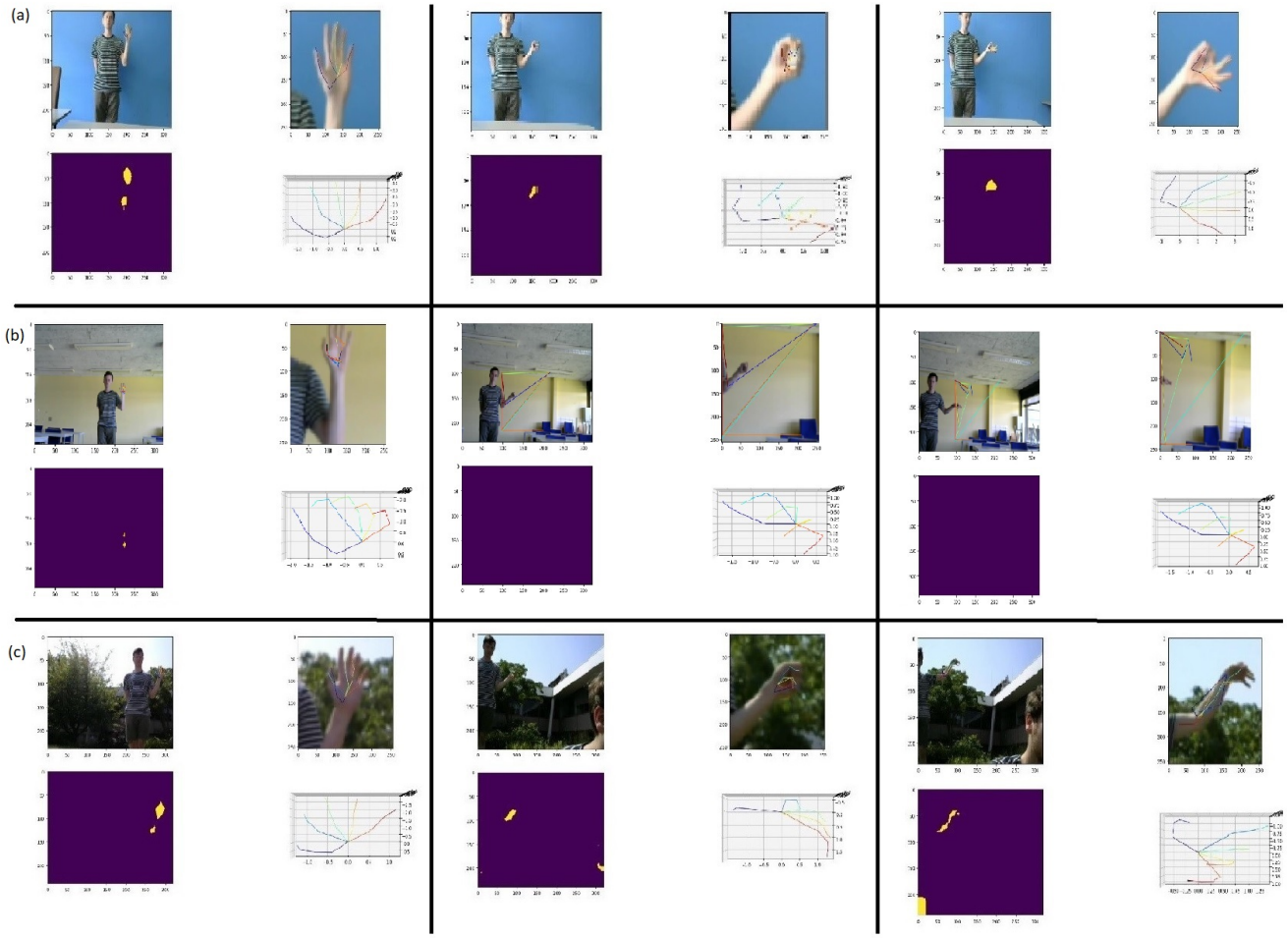


Figure 9: Evaluation of the real-time performance in three different scenario's. The performance of the model was tested in a computer room with blue walls (first row), a yellow classroom (second row) and outside with the subject in front of trees and bushes (third row).

Every cell in the figure above contains four different results of the 3D hand pose estimation model. Below is a short description of each cell's contents:

- The top right of every cell shows the result of HandSegNet and PoseNet: a cropped image of the hand with the kinematic skeleton visualizing the 21 keypoints.
- The top left shows the original image captured by the camera with the 2D kinematic skeleton fixed on top of the hand.
- The bottom left shows the result of HandSegNet: the segmenation of the hand, here indicated by a yellow colour compared to a purple background.
- Finally, the bottom right shows the estimated 3D model of the hand after lifting the 2D keypoints to relative 3D coordinates using PosePrior.

From the images above certain conclusions can be drawn on what conditions have a negative effect on the models accuracy for 3D hand pose detection. The model performed best in the blue

room, because there was a clearer contrast between background and foreground due to the blue wall behind the hand. The model performed the poorest in the yellow class room, as this room was very light and cluttered with chairs and desks. When lifting the hand up higher, the model performed better in the yellow room. This is because the hand contrasts better with the ceiling than with the yellow wall. Below, the results of HandSegNet and PoseNet can be seen when detecting keypoints of the hand in front of the ceiling:

Figure 10: Improving the model's accuracy in the yellow room by having the ceiling as a background

Moving the servo actuated camera outside resulted in a poorer performance than in scenario (a). Specifically, the model would find the location of the hand, as HandSegNet was able to distinguish the hand from the background, but the 2D kinematic model would not align well with the hand's joints. In around 10% of the frames captured outside, the kinematic model would align with the hand, yet the alignment in most frames of scenario (a) would still be more precise. Due to the fact that the model was inaccurate in predicting the 2D keypoints in scenario's (b) and (c), the resulting 3D model did not relate to the actual hand pose in the frame. The model was inaccurate in scenario (c), not only due to the higher temperature that hampered the machine's performance, but also due to the amount of moving objects in the background. Overall, darker, less cluttered rooms ensure for the best real-time performance of the model. The video captures from the servo actuated camera, the real-time implementation of the model, and the results of running the video frames through the model are available such that the qualitative experiments can be recreated and validated.

5 Conclusions and Further Research

The model by Zimmerman and Brox's has been implemented onto a robotic platform. In order to do this first an overview of other publications in the field of 3D hand pose estimation has been given. The accuracy of the model was validated and through experimenting a more accurate 2D keypoint estimation model was found. This was realized by retraining PoseNet with a batch size of 16 and retraining HandSegNet with the same configuration as in the original paper. Combining these retrained configurations of HandSegNet and PoseNet resulted in a higher accuracy of 2D keypoint detection when the model was evaluated on the Rendered Handpose Dataset, with an Area Under Curve score of 0.726 compared to the original paper's score of 0.635. However, the model proposed in the original paper generalized better when evaluated on the Stereo Tracking Benchmark Dataset, as seen in Figure 8. Therefore, the proposed improvements to the 2D keypoint detection must be retrained on other datasets in order to improve its generalization before the proposed improvements can be considered to be significant.

Using the configuration with a higher accuracy on the Rendered Handpose Dataset, this thesis presents an implementation of the model which detects 3D hand poses through video captured input. The implementation of the model causes the frames per second of the video capture to drop significantly. This decrease in speed of the video capture is due to the model estimating the 3D hand pose frame-by-frame, where processing each frame is an intense process for the GPU. Although the speed of the video capture decreases by running the multithreaded approach, the approach does provide 2D keypoint predictions which align with the actual hand joints in the frame, especially in indoor environments where the hand is clearly distinguishable from the background. These keypoints translated to 3D coordinates result in an estimation of the 3D hand pose which is close to the actual hand pose. Improvements can still be made to the multithreaded detection approach. For example, there is room for improvement in the detection accuracy of the 2D keypoints of the hand in scenarios where model has more difficulty distinguishing the hand from the background. Future research to increasing the speed of the multithreaded detection approach in frames per second could also be conducted. One of the updates to the multithreaded approach which would increase the model's detection speed is updating the input stream of the model to a newer version of Tensorflow. This would put more priority on the GPU, which would result in a program with a higher frames per second.

The lack of a large dataset for training this model hampered the accuracy of the model. Therefore, looking at the performance of Mueller *et al's* [13] model on a robotic platform would be an interesting future endeavour, seeing as the model requires a larger GANerated dataset to train. In the future it would also be interesting to see how this model would perform when incorporated in a robot capable of basic human interactions. Zimmerman and Brox show that a gesture recognition model is easily built on top of the hand pose estimation model. The model could then be combined with a gesture recognition model to teach sign language to a robot.

References

[1] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 852–863, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[2] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 63:1–63:8, New York, NY, USA, 2009. ACM.

[3] S. Wu and H. Nagahashi. Real-time 2d hands detection and tracking for sign language recognition. In *2013 8th International Conference on System of Systems Engineering*, pages 40–45, June 2013.

[4] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis Argyros. Efficient model-based 3d tracking of hand articulations using kinect. volume 1, January 2011.

[5] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single RGB images. *CoRR*, abs/1705.01389, 2017.

[6] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *CoRR*, abs/1502.06807, 2015.

[7] Markus Oberweger and Vincent Lepetit. Deepprior++: Improving fast and accurate 3d hand pose estimation. *CoRR*, abs/1708.08325, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[9] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Augmented skeleton space transfer for depth-based hand pose estimation. *CoRR*, abs/1805.04497, 2018.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[11] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, Oct 2017.

[12] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. *CoRR*, abs/1711.07399, 2017.

[13] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[14] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. *CoRR*, abs/1602.00134, 2016.

[15] Dibia Victor. Handtrack: A library for prototyping real-time hand trackinginterfaces using convolutional neural networks. *GitHub repository*, 2017.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[18] Tomas Simon, Hanbyul Joo, Iain A. Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4645–4653, 2017.

[19] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[20] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 3d hand pose tracking and estimation using stereo matching. *CoRR*, abs/1610.07214, 2016.

[21] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.