32nd European Modeling & Simulation Symposium
17th International Multidisciplinary Modeling & Simulation Multiconference

ISSN 2724-0029 ISBN 978-88-85741-44-7 © 2020 The Authors.
DOI: 10.46354/i3m.2020.emss.040

Using Deep Learning for Depth Estimation and 3D Reconstruction of Humans

Alexander Freller[1,2,*], Dora Turk[3] and Gerald A. Zwettler[1,2]

[1]Research Group Advanced Information Systems and Technology (AIST), Research and Development Department, University of Applied Sciences Upper Austria, Softwarepark 11, 4232 Hagenberg, Austria
[2]Department of Software Engineering, Faculty of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Softwarepark 11, 4232 Hagenberg, Austria
[3]AMB GmbH, Hafenstraße 47-51, 4020 Linz, Austria

[*]Corresponding author. Email address: Alexander.Freller@students.fh-hagenberg.at

Abstract

Deep learning for depth estimation from monocular video feed is a common strategy to get rough 3D surface information when an RGB-D camera is not present. Depth information is of importance in many domains such as object localization, tracking, and scene reconstruction in robotics and industrial environments from multiple camera views. The convolutional neural networks UpProjection, DORN, and Encoder/Decoder are evaluated on hybrid training datasets enriched by CGI data. The highest accuracy results are derived from the UpProjection network with a relative deviation of 1.77% to 2.69% for CAD-120 and SMV dataset respectively. It is shown, that incorporation of front and side view allows to increase the achievable depth estimation for human body images. With the incorporation of a second view the error is reduced from 6.69% to 6.16%. For the target domain of this depth estimation, the 3D human body reconstruction from aligned images in T-pose, plain silhouette reconstruction generally leads to acceptable results. Nevertheless, additionally incorporating the rough depth approximation in the future, concave areas at the chest, breast, and buttocks, currently not handled by the silhouette reconstruction, can result in more realistic 3D body models by utilizing the deep learning outcome in a hybrid approach.

Keywords: Depth Estimation; Deep Learning; Convolutional Neural Networks; Human Body 3D Reconstruction

1. Introduction

Depth estimation is the task of trying to recover the 3D geometry of a scene with only a 2D image available. Depth data is used in a wide variety of applications from autonomous driving to robotics. At present acquiring this depth information often requires expensive equipment like RGB-D cameras or lidar sensors. Being able to estimate depth from images, which can be taken with inexpensive cameras, would make using depth information cheaper and easier and would also create new possibilities. One of these could be to recover spatial information from images of humans. This would allow creating 3D models of people from images at a higher accuracy compared to silhouette reconstruction only. Data like this could have many uses from clothing manufacturers quickly surveying the sizes of a target group to online stores only presenting items to customers that fit them or individuals who want to track changes in their bodies.

When taking a 2D image of a 3D environment, a lot of spatial information is lost. This raises the question of whether using multiple images of a person from different angles can help estimate depth more accurately. This article aims to answer this question and also presents and analyzes three deep learning models for the task of depth estimation.

2. State of the art

Laina et al. (2016) use a CNN to directly regress the depth per pixel from an input image. Their network is split into two parts: a feature extractor and an upsampling network. The feature extractor is based on the ResNet architecture (He et al., 2016) with the fully connected classification layers removed. This transforms a high-resolution image into a feature tensor which is then upsampled to the desired resolution. Upsampling is done by a series of unpooling and convolution steps. With this method, they achieve an output resolution of about half the input resolution.

Fu et al. (2018) use a feature extractor in a similar way but produce a higher resolution feature map to preserve spatial information. This feature map is subsequently fed into multiple parallel blocks. The first is a full image encoder which provides a global understanding of the input. The other blocks are dilated convolutions with varying dilation rates. The output of all parallel blocks is concatenated and upsampled to the target resolution. The network is optimized with ordinal regression which splits the depth range into classes of different sizes. This improves the prediction accuracy for low to medium depth values.

Jiao et al. (2018) introduce a split network that predicts a depth map and semantic labels at the same time. They do this by using two networks that produce two different outputs. The networks are connected with their proposed Lateral Sharing Units. These aim to exchange knowledge between both branches of the network. Jiao et al. (2018) also introduce a depth aware loss function. This loss function pays more attention to distant regions of the depth map. Through this, the resulting network more accurately predicts high depth values.

Eigen and Fergus (2015) use a multi-scale deep network to produce feature maps with different resolutions. The first scale uses fully connected layers to produce a feature map with low resolution. The use of a fully connected layer allows the network to understand the image at a global level. This output is passed to the second part of the network. This part uses the output from the previous layers and the original input image to produce a depth map with higher resolutions. The last layers again use the output from the previous part and the original image to further increase the resolution and produce the final prediction.

This article aims to answer two research questions. The first is how accurately can the geometry of a human be estimated from an image with deep learning methods. To answer this, three different models are trained and analyzed on multiple datasets. The second question is whether using two images of the same person from different angles can improve the accuracy of the depth estimations. The same model is trained on two datasets where one provides one image of the subject and the other provides two images.

3. Materials and Methods

3.1. Deep Learning Models for Depth Estimation

The following models are implemented and trained on three depth estimation datasets. The models are chosen because they achieved state of the are results on other depth estimation tasks.

3.1.1. UpProjection Network

This network is based on the UpProjection Network developed by Laina et al. (2016). It uses the Encoder/Decoder strategy. The inputs are first downsampled to a low resolution but a high number of channels by the encoder. In the decoder part of the network the UpProjection blocks (Laina et al., 2016) are used to increase the resolution and reduce the number of channels with every step.

The encoder takes as input a 3-channel image which is then passed through a series of convolution and pooling operations. The data is first downsampled and then processed by several convolutions which are shown in Figure 1.

After the final pooling step, the data is passed to the decoder.

The decoder takes the low-resolution output of the encoder and upsamples it to the final resolution. This step is repeated five times, the same as the number of pooling steps in the encoder. Therefore the output resolution is equal to the input resolution. The number of channels in the output needs to be one, which is ensured by the last convolution in the decoder.

The upsampling in the decoder is done with the UpProjection blocks introduced by Laina et al. (2016). These blocks first use nearest neighbor upsampling to double the resolution of the input. The result of this step is processed by two convolutions as seen in Figure 1. The same result is also passed to another simultaneously executed convolution. The outputs of both convolutions are then combined with an elementwise addition. All convolutions up to this point have the same number of filters as there are channels in the input. The last convolution changes the number of channels to be half of the number of channels in the input. A ReLU is applied after the last convolution.

3.1.2. DORN Network

The DORN network was introduced by Fu et al. (2018) and stands for deep ordinal regression network. The name comes from the fact that they use ordinal regression which divides the depth values into classes and uses the ordinal property of those classes to improve

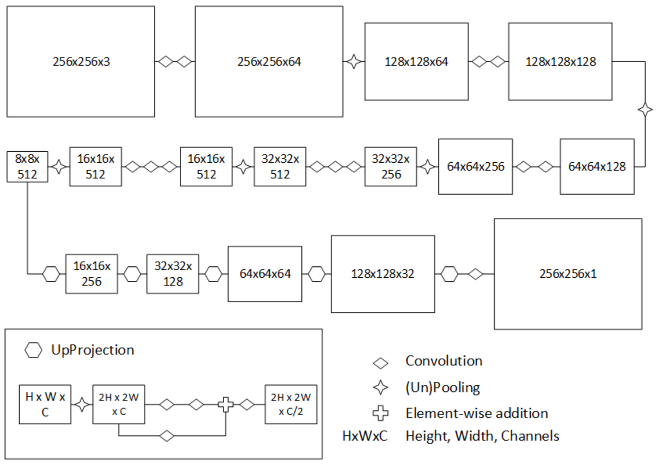


Figure 1. The architecture of the UpProjection Network.

the accuracy. However, to keep results comparable between all three networks this technique is not used.

This network is split into an encoder and decoder similar to the UpProjection network. The encoders architecture is the same as in the first method. It uses pooling and convolutions to extract features from the input image. In this network, the image is not downsampled as often as in the first network which results in a higher resolution of the feature map. This is done to better retain the spatial information contained in the input. When downsampling images to a low resolution and a high number of channels a lot of the spatial information is lost which makes it harder for the network to reproduce the output with fine details. To address this problem, the encoder applies pooling less often and therefore produces a higher resolution result.

The output from the encoder is then processed by five different parallel components as seen in Figure 2. All of those aim to capture a different aspect of the input image. The first component is called Full Image Encoder and was introduce by Fu et al. (2018). It uses a fully connected layer to gather information about the whole image at once.

The second parallel component of the network is a standard convolution meant to capture local details of the feature map to be able to get more details in the final output. The last three parallel components are dilated convolutions with varying rates of dilation. These are used to capture details in a larger area than a normal convolution could. Since all dilated convolutions have different dilation factors, they can extract information at different scales. The convolutions have a size of 3×3 and dilation rates of 6, 12 and 18. After the outputs of all parallel components have been computed they are concatenated and passed to the decoder.

The decoder upsamples the combined outputs to the resolution of the input image. This is done with a series of unpooling and convolution operations as shown in Figure 2.

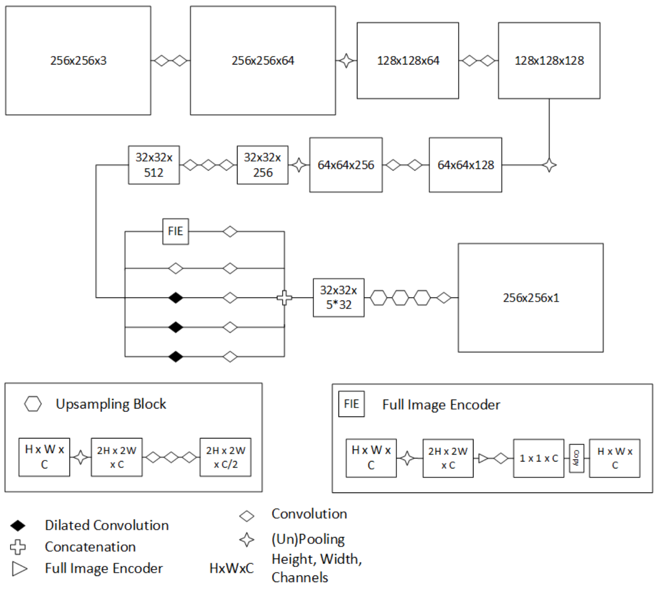


Figure 2. The architecture of the DORN Network.

The network contains three upsampling blocks such that the output and input resolutions are equal. The output of the last upsampling block is passed to a convolution with one filter.

3.1.3. Encoder/Decoder

The Encoder/Decoder network uses a standard Encoder/Decoder architecture but adds skip connections to the network structure. These connections give the decoder access to the feature maps produced in the encoder. The upsampling and downsampling blocks shown in Figure 3 can have different structures depending on what kind of pre-trained network is used.

The encoder uses a VGG16 network with pre-trained weights to produce the feature map. This architecture uses MaxPooling and multiple convolutions in its downsampling block.

The downsampling blocks always produce an output with twice the number of channels and half the resolution of the input. The image is passed through multiple downsampling blocks until the output has a low resolution but a high number of channels.

The output of the encoder is passed to the decoder for upsampling. The upsampling blocks in this network use the same architecture as the ones in the DORN network. The input is first processed with nearest neighbor upsampling and then transformed with multiple convolutions. In every upsampling block, the resolution of the input is doubled while the number of channels is halved. This means at every step in the decoder there is a corresponding step in the encoder that has the same resolution and number of channels as the data at the current step. This allows for the data of the encoder to be added to the data in the decoder. This is done with an element-wise addition after every upsampling block.

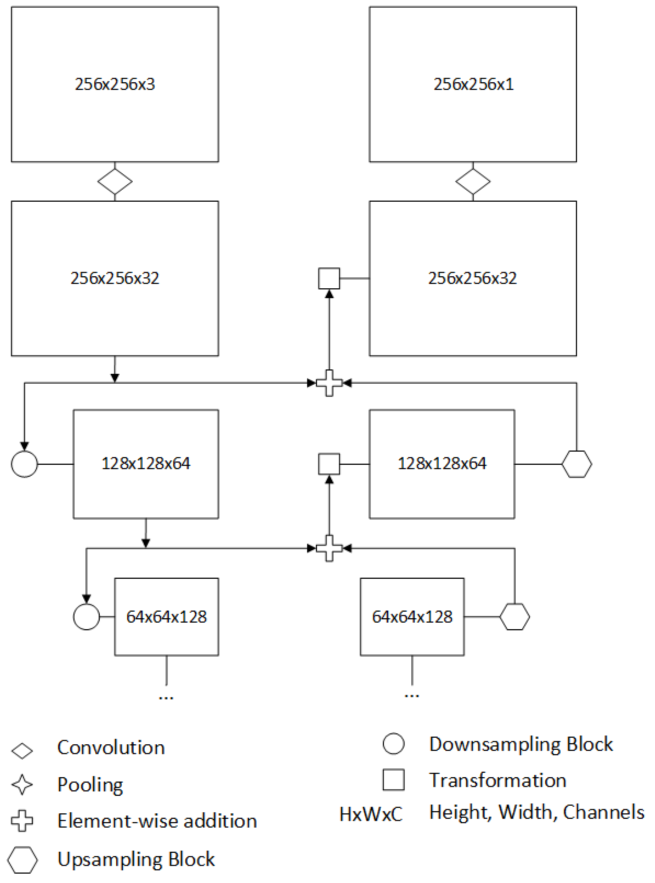


Figure 3. The architecture of the Encoder/Decoder Network.

After combining the data, it is transformed with batch normalization and three consecutive convolutions.

These steps of upsampling, combination and transformation are repeated until the resolution of the decoder is equal to the input resolution. Finally, the data is passed through one last convolution to reduce the number of channels to one.

3.2. Datasets

Three datasets are used to analyze the performance of the models under different circumstances. Figure 4 shows a sample from all datasets.

The first dataset is the Cornell Acitivity Dataset CAD-120 (Koppula et al., 2013) which contains 120 videos of people performing various every day actions like making cereal or taking medicine. The dataset provides amongst other things RGB and depth data taken with an RGB-D camera. Due to the limitations of the technology the depth data contains some invalid data which has to be considered when training the model.

The second dataset is the Synthetic Mixed View Dataset (SMV) which is a computer-generated dataset specifically created for the task of training a neural network for depth estimation. The samples consist of an image of a person in a 3D scene and the associated depth map. The images are generated by randomizing certain variables. These variables include the position and orientation of the person in the scene and the position of the camera. Since this dataset is artificially generated it contains no invalid values or noise. This is useful to test the models under ideal conditions.

The third dataset is the Synthetic Double View Dataset (SDV) which is generated in the same manner as the previous dataset. The difference is, that this dataset contains two views of the subject one from the front and one from the side and the depth data from the front. This dataset is used to investigate whether the second view can improve the prediction accuracy.

For all datasets, the depth values are normalized such that all values are between 0 and 1. The validation and test sets each consist of 10 % of the whole dataset.

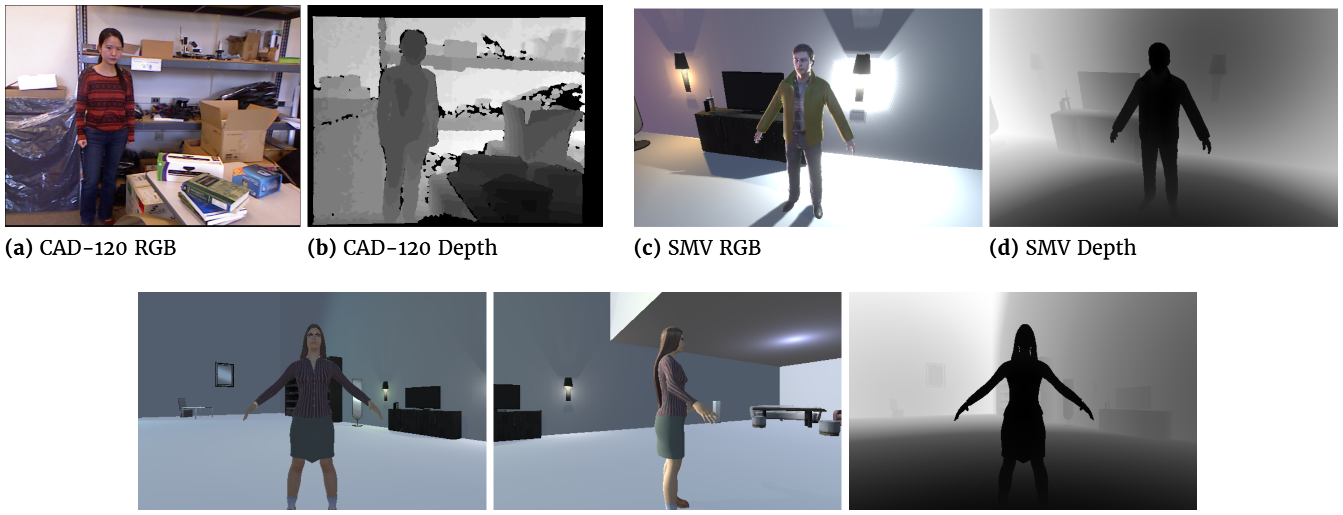
4. Implementation Details

All models implemented using Python 3.7 and TensorFlow 2.0. Most of the models use the functional Keras API built into Tensorflow, additionaly some low-level TensorFlow operations are used when necessary. Data loading and preprocessing is done using the TensorFlow Dataset API.

To speed up the training process all models use a pretrained network as the encoder. The VGG16 network is used with weights that are pretrained on the ImageNet dataset. The standard VGG16 architecture has a fully-connected classification layer at the end which is not needed for this application so it is removed. The weights of the pretrained network are not updated during training which leaves only the weights of the decoder as the trainable parameters of the networks. The UpProjection network has about 25 million trainable parameters while the DORN and Encoder/Decoder networks have 18 million and 60 million trainable parameters respectively.

Listing 1 shows the implementation of the UpProjection block. The `inputs` variable is the input tensor that is first upsampled using the `UpSampling2D` layer which doubles the height and width using nearest neighbour upsampling. The upsampled tensor is then transformed with `Conv2D` layers such that the number of channels is equal to the desired number out output channels as specified by the variable `channels_out`. All convolutions use 3×3 filters and the option `padding='same'` to ensure that the height and width of the tensor does not change when applying the convolution. The results are combined with the skip connection using the Python `+`-operator which is implemented as element-wise addition in TensorFlow.

The implementation of the Full Image Encoder of the DORN Network is shown in Listing 2. First, average pooling is applied to the input which halves the width and height of the tensor. Then the tensor in flattened into a vector. This is necessary to use it as input for



(a) CAD-120 RGB **(b)** CAD-120 Depth **(c)** SMV RGB **(d)** SMV Depth

(e) SDV RGB Front **(f)** SDV RGB Side **(g)** SDV RGB Depth

Figure 4. One sample from all of the datasets. Training and testing material uses RGB and depth data from the CAD-120 dataset (a-b), the SMV dataset with RGB and depth respectively (c-d) as well as the SDV dataset providing views from the front and side besides ground truth depth (e-g).

Listing 1. Implementation of the UpProjection block.

```
upsampled = layers.UpSampling2D()(inputs)
x = layers.Conv2D(channels_in,3,padding='same',activation='relu')(upsampled)
x = layers.Conv2D(channels_out,3,padding='same')(x)

skip_connection = layers.Conv2D(channels_out,3,padding='same')(upsampled)
x = x + skip_connection
x = layers.Conv2D(channels_out,3,padding='same')(x)
outputs = layers.ReLU()(x)
```

a dense layer. The dense layer outputs a tensor with size 16×128 which is the batch size and the output size of the dense layer. To use this tensor as the input to a convolution it needs to be changed such that it has a size of $BATCH \times WIDTH \times HEIGHT \times CHANNELS$. This is done by passing `None` to the indexer which adds two unit axes to the tensor. It now has a shape of $16 \times 1 \times 1 \times 128$ and can be transformed by the convolution. After the convolution, the tensor is broadcast to a shape of $16 \times 32 \times 32 \times 32$ so it is compatible with the results of the other parallel components.

5. Results and Discussion

5.1. Experimental Setup

To compare the performance of all models, the same training and evaluation process is used for all experiments. All models are optimized with the RMSProp optimizer with a decay rate $\rho = 0.9$ and $\delta = 10^{-7}$. The learning rate ϵ starts at a value of 10^{-4}. After every epoch the learning rate is reduced by a factor of 1.25. All models use a batch size of 16 and are trained for 12 epochs. The batch size and number of epochs are mainly constrained by the available hardware. All networks are trained on a NVIDIA GeForce GTX 970 with 4 GB of GPU memory. Training for one epoch on the CAD-120 dataset takes about one hour which is why training was stopped after 12 epochs for all networks. The number of epochs was chosen because the at that point the loss does not go down significantly anymore, so even longer training would likely only result in marginal improvements of the results.

The loss function that is utilized for training and evaluation measures the relative average deviation from the ground truth depth. This is done with

$$l = \frac{1}{n} * \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} \tag{1}$$

where y and $\hat{y}$ are the vectors of ground truth and predicted values and n is the number of pixels. The loss represents the average relative deviation from the ground truth value. Since the CAD-120 Koppula et al. (2013) dataset contains some invalid zero values, these

Listing 2. Implementation of the Full Image Encoder.

```
x = layers.AveragePooling2D()(inputs)
x = layers.Flatten()(x)
x = layers.Dense(128,activation='relu')(x)
x = layers.Conv2D(32,1,activation='relu')(x[:,None,None,:])
x = tf.broadcast_to(x,(16,32,32,32))
```

are ignored when computing the loss, otherwise the network learns to predict where the sensor can not measure the depth which is not desired. All validation and test sets are comprised of 10 % of the full dataset.

5.2. Metrics

All models are evaluated with several metrics to judge their performance. For all metrics y is the ground truth value, $\hat{y}$ is the predicted value and n is the number of pixels.

The first metric is the average relative deviation (denoted rel) which is calculated with

$$rel = \frac{1}{n} * \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} \; . \tag{2}$$

The second metric is the average absolute deviation (denoted abs) which is given by

$$abs = \frac{1}{n} * \sum_{i=1}^{n} |y_i - \hat{y}_i| \; . \tag{3}$$

The third metric is the root mean squared error (denoted rms) that is calculated with

$$rms = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \; . \tag{4}$$

The final metric is the accuracy within threshold (denoted δ_t) which is based on the definition by Cao et al. (2017) but altered to be

$$\delta_t = \frac{1}{n} * \sum_{i=1}^{n} \begin{cases} 1 & \text{if } \frac{|y_i - \hat{y}_i|}{y_i} < t \\ 0 & \text{otherwise} \end{cases} \; . \tag{5}$$

This metric gives the percentage of pixels with a relative deviation from the ground truth of less than t. For example, $\delta_{5\%} = 99\%$ means that 99% of predictions deviate less than 5% from the real value.

5.3. Results

To determine, which of the models achieves the best performance, all of them are trained on the CAD-120 and SMV datasets. Tables 1 and 2 show the metrics for

Table 1. Evaluated metrics for the CAD-120 dataset.

Model	$\delta_{25\%}$	$\delta_{10\%}$	$\delta_{5\%}$	rel	abs	rms
UpProj	99.14%	98.13%	94.15%	0.0177	0.0066	0.0235
DORN	99.14%	97.08%	90.25%	0.0250	0.0089	0.0322
EncDec	96.64%	91.18%	76.44%	0.0500	0.0182	0.0420

Table 2. Evaluated metrics for the SMV dataset.

Model	$\delta_{25\%}$	$\delta_{10\%}$	$\delta_{5\%}$	rel	abs	rms
UpProj	99.20%	97.91%	92.39%	0.0269	0.0059	0.0120
DORN	98.29%	95.15%	85.37%	0.0315	0.0085	0.0161
EncDec	95.01%	89.10%	70.35%	0.0718	0.0156	0.0321

Table 3. Evaluated metrics for UpProjection network trained on the SDV dataset.

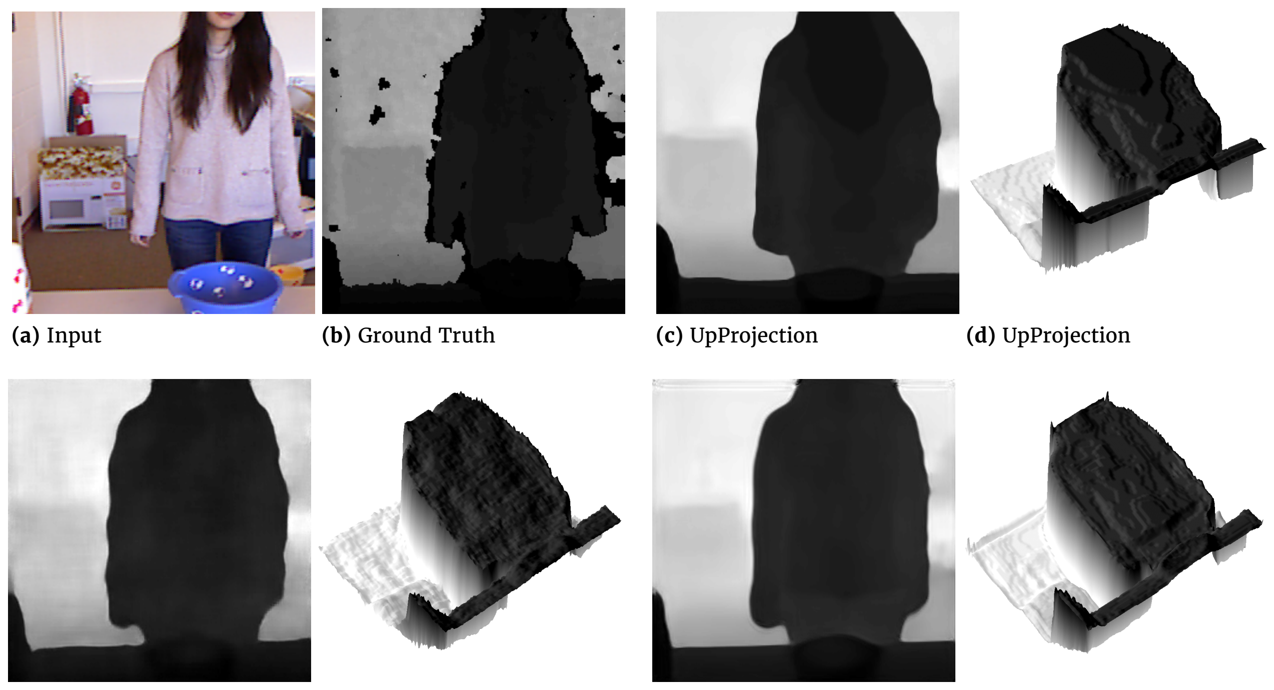
View	$\delta_{25\%}$	$\delta_{10\%}$	$\delta_{5\%}$	rel	abs	rms
Single	97.01%	84.25%	57.90%	0.0669	0.0208	0.0489
Double	97.35%	85.70%	60.81%	0.0616	0.0191	0.0412

all models on both datasets. The metrics show, that the UpProjection network achieves the best performance on both datasets with an average relative deviation of 1.77% on the CAD-120 test set and 2.69% and the SMV test set. Figure 5 shows a prediction and the resulting 3D surface for all models on the CAD-120 dataset. The outputs show, that all models can predict the larger differences in depth between the subject and the background but only the UpProjection network can capture the details of the person. While the prediction of the DORN and Encoder/Decoder networks show a flat surface at the torso the UpProjection network is able to reconstruct the concave shape more accurately.

Since the UpProjection Network achieves the best results it is used to investigate whether a second view of the same subject can improve the results. The network is trained twice on the SDV dataset. The first time only the front view is used to establish a baseline and the second time both views are used in training. The results are listed in Table 3. The comparison shows that the second view does improve the accuracy of the prediction from a relative deviation of 6.69% to 6.16%.

6. Conclusion and Outlook

This research work evaluates the achievable accuracy of relevant depth approximation networks for human shape datasets. Results proof, that incorporation of



(a) Input **(b)** Ground Truth **(c)** UpProjection **(d)** UpProjection

(e) DORN **(f)** DORN **(g)** Encoder/Decoder **(h)** Encoder/Decoder

Figure 5. Prediction and 3D surface of all models on the CAD-120 dataset.

front and side view allows increasing achievable accuracy. The depth approximation of the human body shows extremities such as legs and arms and the upper body as a non-planar and elevated profile. Based on this rough relative depth approximation, the accuracy of 3D body reconstruction utilizing silhouette reconstruction will be improved in the future. After aligning the body and skeletal limbs to a reference T-pose model, silhouette reconstruction leads to solid results close to a convex body shape. Utilizing a rough depth approximation, the concave areas at the chest, breast, and the buttocks, which are currently not handled by the silhouette reconstruction, can result in more realistic 3D body models by utilizing the deep learning outcome in a hybrid approach.

7. Funding

This research is initiated and guided by AMB Technology GmbH through project TrueSize S.M.B.S. 21650377 with basic funding program number 872105, sponsored by the Austrian Research Promotion Agency FFG, fed by research budget of the Federal Republic of Austria. Contact: Anna Maria Brunnhofer, CEO.

References

Cao, Y., Wu, Z., and Shen, C. (2017). Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3174–3182.

Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658.

Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018). Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Jiao, J., Cao, Y., Song, Y., and Lau, R. (2018). Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 53–69.

Koppula, H. S., Gupta, R., and Saxena, A. (2013). Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8):951–970.

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. (2016). Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE.