Deep Orientation: Fast and Robust Upper Body Orientation Estimation for Mobile Robotic Applications

Benjamin Lewandowski[1], Daniel Seichter[1], Tim Wengefeld[1], Lennard Pfennig[1],
Helge Drumm[2] and Horst-Michael Gross[1]

***Abstract*— An essential feature for navigating socially with a mobile robot is the upper body orientation of persons in its vicinity. For example, in a supermarket orientation indicates whether a person is looking at goods on the shelves or where a person is likely to go. However, given limited computing and battery capabilities, it is not possible to rely on high-performance graphics cards to run large, computationally expensive deep neural networks for orientation estimation in real time. Nevertheless, deep learning performs quite well for regression problems. Therefore, we tackle the problem of upper body orientation estimation with small yet efficient deep neural networks on a mobile robot in this paper. We employ a fast person detection approach as preprocessing that outputs fixed size person images before the actual estimation of the orientation is done. The combination with lightweight networks allows us to estimate a continuous angle in real time, even using a CPU only. We experimentally evaluate the performance of our system on a new, self-recorded data set consisting of more than 100,000 RGB-D samples from 37 persons, which is made publicly available. We also do an extensive comparison of different network architectures and output encodings for their applicability in estimating orientations. Furthermore, we show that depth images are more suitable for the task of orientation estimation than RGB images or the combination of both.**

I. Introduction

In this paper, we tackle the problem of orientation estimation of persons with a mobile robot in a supermarket environment. In continuation of our previous research [1], our long term goal is to let the robot navigate through the store during opening hours autonomously, i.e. in the presence of customers and employees, in order to automatically detect out-of-stocks. Since supermarkets consist of very narrow and long aisles, it is important to have a robust situational awareness that enables a social navigation behavior. An essential feature in this context is the upper body orientation of humans indicating e.g. whether a person will enter an aisle or not. Furthermore, the upper body orientation is useful for calculating a person's social space [2] or to approach persons from the right direction [3] when asking them to step aside.

In order to estimate the upper body orientation as a continuous angle around the axis perpendicular to the ground, we make use of modern deep learning techniques. Starting with the AlexNet [4], deep learning was very successful in solving computer vision problems over the last few years. Especially its robustness, the huge generalization capabilities, and the fact, that relevant features are learned automatically, contribute to the great success of deep learning. The major drawbacks on the other hand are the huge amount of data required for training and its computational complexity that often requires high-performance graphics cards in order to reach an inference in real time. Since mobile robots are usually restricted with respect to computation and battery capabilities, we are not able to use such high-performance hardware on our platform. Therefore, we reduce computational costs by relying on lightweight networks. In addition, we separate the person detection task from orientation estimation. Instead of applying an end-to-end approach, we only process fixed size image patches of persons previously extracted by a person detector as shown in Fig. 1. These patches are free of background which simplifies the acquisition of training data and makes the approach generic for unknown environments.

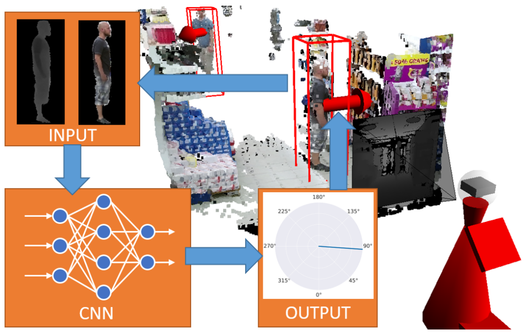


Fig. 1: Overview of our approach: Person images segmented with a person detector [5] are fed into a fast convolutional neural network (CNN) to allow precise real-time orientation estimation on mobile platforms.

Modern robots can detect people in RGB images as well as in 3D data due to RGB-D sensors like the Kinect2. Therefore, we examine and compare the performance of our system with different input types. In particular, we trained several neural networks for orientation estimation using depth and RGB images as well as a combination of both. Since there is no public data set available with these kinds of input and with highly precise ground truth labels, we recorded our own.

In summary, our main contributions are:

1) a new data set with more than 100,000 RGB-D samples of 37 persons for upper body orientation estimation
2) we modify the existing network architecture of [6] and do an extensive experimental comparison to the more recent MobileNet v2 [7], different input types (RGB,

[1,2]Authors are with Neuroinformatics and Cognitive Robotics Lab (1) and University Computer Center (2), Technische Universität Ilmenau, 98694 Ilmenau, Germany. `fg-nikr@tu-ilmenau.de`

This work has received funding from the German Federal Ministry of Education and Research (BMBF) to the project ROTATOR (grant agreement no. 03ZZ0437D), and to the project 3D-PersA2 (grant agreement no. 03ZZ0460) in the program Zwanzig20 – Partnership for Innovation as part of the research alliance 3Dsensation.

depth, RGB-D), output encodings, and runtimes

3) a complete system for continuous orientation estimation suitable for real-time mobile robotic applications on CPU or GPU, comprising of a lightweight network architecture, uncertainty modeling, and a data set
4) we have made our data set and our code including network weights publicly available to the community[1]

II. Related Works

Extracting the upper body orientation from a single input image can be realized in two ways. It can be directly estimated or it can be derived from a skeleton estimation, in which the orientation is part of its solution. The idea of the latter one is to calculate joints and bones of the human body in a first step and then to derive the orientation, i.e. the normal vector of the sternum, via geometrical relations. There are various approaches for skeleton estimation in the literature, e.g. [8], [9], [10], [11]. For extracting orientation from skeleton data, 3D joint positions are necessary. Obviously, the accuracy of extracted orientations strongly depends on the precision of the joint estimations. However, an accurate skeleton estimation is computational expensive and often requires powerful graphics cards for real-time inference. Hence, it is less suitable for a mobile platform.

For our application environment, a direct estimation of the upper body orientation is therefore more appropriate. In this case, features extracted from the input image are mapped to an orientation [12], [13], [14], [15]. More recent approaches directly feed the input into a deep neural network [16], [17], [18]. Similar to estimating the upper body orientation is the estimation of the head orientation [19], [6]. The head pose could be an indicator of the direction in which a person is looking. Since we want to let our robot operate in a supermarket, we think this is a less robust feature for social navigation because people keep changing their view from shelf to shelf. However, the idea for estimating the head's orientation is the same: mapping an input image to one or more orientation angles. Therefore, some approaches were developed for estimating both orientations [15], [17].

Estimating orientation directly, head or torso, can be done in two ways, either by classification or by regression. A large part of the literature formulates orientation estimation as a multi-class classification problem [12], [13], [15], [14], [16], [17]. For this purpose, continuous orientation angles, ranging from $0°$ to $360°$, are discretized into a fixed set of orientation classes. Often eight classes are used, each covering a range of $45°$. A drawback of treating orientation as classification is that during training, neighboring classes are evaluated just as incorrectly as more distant classes, even if two samples differ only by a few degrees. In [20] this problem is addressed by introducing a cost relaxation for neighboring classes into the SVM classifier. Nevertheless, the inherent discretization error is quite large when doing classification of orientations. Given $45°$-bins, a perfect classifier and a set of equally distributed data samples between $0°$ and $360°$, the mean absolute error would be at least $11.25°$. Increasing the number of classes, on the other hand, increases computation time and also the number of required data for training.

Estimating continuous angles, i.e. formulating it as regression problem, promises more accurate estimates since there is no discretization error. Examples for such approaches are [19] and [21]. One issue with regression is the periodicity of the angle. Predicted angles above $360°$ must be equal to their corresponding angles in the range of $[0°, 360°)$. In [6] this problem is addressed by using two output neurons in a neural network encoding the sine and cosine of the estimated angles. Calculating the arctangent subsequently results in angles in the range of $[0°, 360°)$. The approach was successfully adopted for orientation estimation of objects in street scenes [22]. Furthermore, in [6] a specially tailored loss function, called von Mises loss, is used to assess multiple values of the same angle equally during training. Two output neurons are also used in [18], in which both represent the x- and y-components of the direction vectors of persons.

Based on related literature, we decided to rely on the direct orientation estimation from a single input for our system. Since we want to be as accurate as possible, we employ a deep neural network with two output neurons for regression. Futher details on our approach are given in Sec. IV.

III. RGB-D Orientation Data Set

Over the years, a considerable amount of data sets to train and benchmark orientation estimation of persons were made publicly available with RGB [23] and RGB-D [24] data. However, to the best of our knowledge, none of them meets our requirements of synchronized depth and RGB data streams in combination with highly precise ground truth labels suitable for regression. Therefore, we decided to record our own data set using a highly precise external ARTTRACK tracking system [25], which tracks markers using four IR-cameras with a positional precision of $0.4\text{mm} \pm 0.06\text{mm}$ [26]. One major drawback of external tracking systems is their limited field of view, so that recorded samples usually contain repetitive backgrounds. We tackle this issue during data generation by subtracting a learned background model based on depth information. We recorded our data set with five synchronized Kinect2 RGB-D devices, placed almost in a half circle around the recording area (see

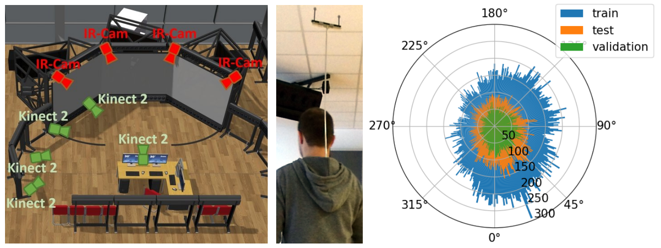
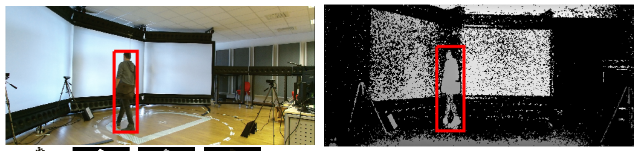
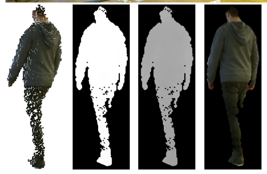


Fig. 2: Left: Camera setup used for recording. Middle: Rod with IR markers attached at the back of each subject. Right: Binned histogram of ground truth angles in our data sets visualized as polar plot. The sample count is encoded in radial, the ground truth label in the angular dimension.

[1]Our code and data set are available at: https://www.tu-ilmenau.de/neurob/data-sets-code/rgb-d-orientation/

data set	persons male	persons female	samples
training	14	7	57717
validation	4	3	19368
test	4	5	31420

Fig. 3: Data provided in our data set. Top: Synchronized RGB and depth images. Lower left (left to right): Cropped point cloud, mask, masked depth image and masked RGB image. Lower right: The distribution of our data set.

Fig. 2). This sensor configuration provides an acceptable distribution of ground truth labels and, more importantly, prevents interferences between the depth measurements of opposing sensors. In order to generate ground truth labels, two IR markers on each subject were captured by the external tracking system. To reduce the influence of the markers in our data, they were attached above the persons' heads, using a thin metallic rod under the clothes on the back (Fig. 2). A ground truth label was then generated by projecting the orthogonal vector from the IR markers onto the ground plane in camera coordinates.

We aimed to split the data set in a 50-25-25 manner for training, validation, and test, respectively. However, we made sure that no person appears in more than one set. Furthermore, we tried to equalize the persons' attributes regarding gender, clothing, and especially the clothing color. Since it is not possible to meet all these constraints, we have given a preference to the test set in order to get a better statement for generalization capabilities. Therefore, the final split of our data set slightly differs from our intended ratio. Statistics and examples of our data set are shown in Fig. 3. All recordings took place in December with volunteers of our university. Hence, the clothes vary from thick jackets to normal office wear for this time of the year. The majority of persons are of Caucasian origin. However, one person with Middle Eastern and two with Asian appearance are included too. To get a high pose variance, we recorded different every day actions like walking around, using a cell phone, or talking to the instructor.

IV. Deep Learning-Based Orientation Estimation

Our goal is to estimate the orientation of persons with a mobile robot in real time. Therefore, we only consider lightweight network architectures, neglecting computationally expensive state-of-the-art architectures such as ResNet [27] or ResNext [28]. Furthermore, since treating orientation estimation as classification task limits precision, we focus on continuous regression only. Due to the promising results for continuous head pose estimation in [6], we decided in favor of the biternion net architecture as baseline for deriving a suitable network architecture for upper body orientation estimation.

TABLE I: Biternion net architecture [6] for different input sizes. Abbreviations: Conv - 3×3-convolution with stride 1 and no padding, BN - batch normalization, ReL - rectified linear activation function, Pool - non-overlapping max pooling (pool size), D - dropout (dropout probability), FC - fully connected layer, Act - output activation function.

	square small n_{in}×46 ×46	square medium n_{in}× 68 ×68	large n_{in}× 123 ×54
Conv+BN+ReL	24×44 ×44	24× 66 ×66	24× 121 ×52
Conv+BN+ReL	24×42 ×42	24× 64 ×64	24× 119 ×50
Conv+BN+ReL	-	-	24× 117 ×48
Pool(2×2)	24×21 ×21	24× 32 ×32	-
Pool(3×2)	-	-	24× 39 ×24
Conv+BN+ReL	48×19 ×19	48× 30 ×30	48× 37 ×22
Conv+BN+ReL	48×17 ×17	48× 28 ×28	48× 35 ×20
Conv+BN+ReL	-	-	48× 33 ×18
Pool(2×2)	48× 9 ×9	48× 14 ×14	-
Pool(3×3)	-	-	48× 11 ×6
Conv+BN+ReL	64× 7 ×7	64× 12 ×12	64× 9 ×4
Conv+BN+ReL	64× 5 ×5	64× 10 ×10	64× 7 ×2
Pool(2×2)	-	64× 5 ×5	-
D(0.2)+FC+ReL	512	512	512
D(0.5)+FC+Act	n_{out}	n_{out}	n_{out}

In the following, we first revisit the biternion net architecture. Afterwards, we apply it to the task of upper body orientation estimation and examine the impact of different input types and output encodings. Finally, we introduce an adapted version of the biternion net architecture slightly boosting the performance, and compare it to recently published MobileNet v2 [7].

A. *Biternion Net Architecture as Baseline*

The biternion net architecture is inspired by the ImageNet winning VGG architecture [29], which in general consists of two stages. In the first stage, multiple convolutional layers followed by one max-pooling layer are used to extract meaningful features. Unlike the original VGG architecture, Beyer *et al.* [6] further added batch normalization [30] in order to reduce the covariate shift in the internal activation to speed up training. In the second stage, two fully-connected layers perform the head orientation estimation task. In contrast to classical regression, directly estimating the angle in radians using a single output neuron ($n_{out} = 1$), Beyer [6] proposed to encode the angle using both sine and cosine component ($n_{out} = 2$), which they call biternion encoding. This output encoding is bijective in the interval of $[0°, 360°)$ and takes periodicity into account. In order to prevent co-adaption and also to improve generalization abilities, dropout [31] is applied before each fully-connected layer. In total, Beyer [6] proposed three slightly modified versions of the biternion net architecture, each tailored to the fixed size of the RGB images ($n_{in} = 3$) of the data set applied to. In the following, we refer to the different input sizes as *square small*, *square medium*, and *large*. For further architecture details, see Tab. I.

Besides handling periodicity using biternion encoding only, [6] also addresses discontinuity in the loss function by turning to the von Mises cost function. The von Mises cost function is derived from the von Mises distribution [32], which is a close approximation of the normal distribution on

TABLE II: Mean absolute errors in degrees (averaged over 3 runs) obtained on test set by applying the biternion net architecture using different input types and output encodings. For each combination, the best result across the input sizes is printed in bold. The overall best approach is underlined.

	single output			biternion output		
	square small	square medium	large	square small	square medium	large
Depth	11.15	**10.30**	10.68	6.25	5.89	**5.68**
RGB	28.53	27.21	**23.65**	19.92	16.99	**16.74**
RGB-D	**11.45**	13.52	13.98	9.48	10.44	**9.05**

the unit circle and, therefore, suited for handling periodicity. Since this cost function clearly outperforms the commonly used mean squared error in [6], we decided in favor of the von Mises cost function in all experiments too.

B. Application to Upper Body Orientation Estimation

In order to examine whether the promising results of the biternion net architecture can be transferred to upper body orientation estimation, we conducted a first set of experiments on our data set. Since our data set comes with both RGB and depth images for all samples, we were able to compare three different types of input: *RGB*, *depth*, and *RGB-D*. However, since both images provide different statistics, a combined input as four-channel image would lack of representational power. To take different statistics into account, we replicated the first layers of the networks (see dashed line in Tab. I) and joined the independent branches right before the first pooling layer. Fusion is done by concatenating the feature maps of both branches and by applying a $1{\times}1$-convolution that combines the features pixel by pixel and halves the number of feature maps. In this way, the subsequent structure remains identical. In addition to different input types, we evaluated both output encodings: *single output* and *biternion output* as well. Thus, a total of 18 (3 input types $\times$ 3 input sizes $\times$ 2 output encodings) different networks are compared in this first set of experiments.

1) Training: All networks were implemented using Tensorflow [33] and trained using eight NVIDIA GTX 1080Ti graphics cards. For parameter optimization, we used stochastic gradient descent (SGD), a fixed momentum of 0.9 and a batch size of 128. As initial learning rates, we used 0.0001, 0.001, 0.01, and 0.02, respectively. During training, the learning rate was further adapted using a polynomial learning rate decay. For data augmentation, we randomly flipped the images horizontally. The final weight configuration was chosen within 800 epochs based on the performance on the validation set. To ensure reporting meaningful results only, we repeated each experiment three times in a row with different random seeds and averaged the results on the test set. For further details on the training procedure and other hyper parameters, we refer to the implementation.

2) Results: Tab. II shows results of the first set of experiments. It is obvious that the biternion output encoding clearly outperforms classic regression using a single output neuron by a large margin. Since this coincides with the findings in [6], we omitted the single output encoding in subsequent

TABLE III: Modified biternion net architecture [6] for different input sizes. Abbreviations are the same as in Tab. I. Except for the last two convolutions, "same"-padding is used.

	square small $n_{in}\times 48 \times 48$	square large $n_{in}\times 96 \times 96$	large $n_{in}\times 126 \times 48$
2×Conv+BN+ReL	$24\times 48 \times 48$	$24\times 96 \times 96$	$24\times 126 \times 48$
Conv+BN+ReL	-	$24\times 96 \times 96$	$24\times 126 \times 48$
Pool(2×2)	$24\times 24 \times 24$	-	-
Pool(3×3)	-	-	$24\times 42 \times 24$
Pool(3×3)	-	$24\times 32 \times 32$	-
2×Conv+BN+ReL	$48\times 24 \times 24$	$48\times 32 \times 32$	$48\times 42 \times 24$
Conv+BN+ReL	-	$48\times 32 \times 32$	$48\times 42 \times 24$
Pool(2×2)	$48\times 12 \times 12$	$48\times 16 \times 16$	-
Pool(3×2)	-	-	$48\times 14 \times 12$
Conv+BN+ReL	$64\times 10 \times 10$	$64\times 14 \times 14$	$64\times 12 \times 10$
Conv+BN+ReL	$64\times 8 \times 8$	$64\times 12 \times 12$	$64\times 10 \times 8$
Pool(2×2)	$64\times 4 \times 4$	$64\times 6 \times 6$	$64\times 5 \times 4$
D(0.2)+FC+ReL	512	512	512
D(0.5)+FC+Act	n_{out}	n_{out}	n_{out}

experiments. Moreover, depth information seem to be better suited for estimating the upper body orientation than color information. Combining both inputs does improve the performance but cannot compete with pure depth information. Therefore, we have not pursued combined inputs any longer. Regarding the input size, the results are very close, with the largest size slightly ahead. As a larger input size also means that the number of computations increases, we did not fix the input size at this time.

C. Modified Biternion Net Architecture and MobileNet v2

While comparing different input types and output encodings, we noticed that the trained networks tend to misestimate the orientation when the person is close to the edges of the image. We assume that this behavior results from applying convolutions without appropriate padding since it is not possible to integrate complex features from these areas without padding. In order to verify this assumption, we replaced all convolutions, except the last two ones close to fully-connected layers, with convolutions with "same" padding, i.e. padding of 1 pixel. Due to this change, the features maps remain larger as well. We tried to compensate this by adding a third pooling layer to all networks. In addition, we removed all bias weights from the convolutional layers since each convolutional layer is followed by a batch normalization. As a final step, since the results for *square small* and *square medium* (see Tab. II) are very close to each other and in favor of a fair comparison to the smallest version of MobileNet v2, we dropped the less efficient architecture for *square medium* but added an architecture for inputs of size $96{\times}96$. In the following, we refer to this new input size as *square large*. Tab. III summarizes all architectural changes. Note that the overall number of parameters for the modified *square small* and *large* networks are still of the same magnitude as for the original biternion net architectures. Thus, the results for both are still comparable.

However, the architectures used so far are relatively small. Therefore, we decided to include MobileNet v2 [7] as a more sophisticated but still suitable architecture for mobile applications in our experiments as well. The MobileNet v2

TABLE IV: Mean absolute error in degrees (averaged over 3 runs) obtained on test set for different architectures and input types. For the MobileNet v2 architecture, only the best result and the result of the network with similar number of weights ($\alpha = 0.5$) are given. For both input types, the best result is printed in bold. The overall best result is underlined.

		square small	square large	large
Depth	Biternion net architecture	6.25	-	5.68
	Mod. biternion net architecture	5.73	5.54	5.44
	MobileNet v2 ($\alpha = 1.0$, Adam)	-	**<u>5.35</u>**	-
	MobileNet v2 ($\alpha = 0.5$, Adam)	-	5.79	-
RGB	Biternion net architecture	17.36	-	13.64
	Mod. biternion net architecture	11.54	10.61	9.87
	MobileNet v2 ($\alpha = 1.0$, Adam)	-	**8.13**	-
	MobileNet v2 ($\alpha = 0.5$, Adam)	-	9.58	-

architecture was designed especially for processing RGB images and, therefore, might improve our results for estimating the upper body orientation using RGB images. The architecture is based on the ResNet architecture [27] but uses inverted residuals with shortcut connections located between the thin bottleneck layers. To further reduce the number of computations, lightweight depthwise convolutions [34] are used in the intermediate expansion layers. For further details on the architecture, we refer to [7]. The MobileNet v2 architecture provides two hyper parameters to adjust its complexity. First, the input size which is fixed to 96×96 (smallest possible size) in our case. Second, the width multiplier that multiplies the number of features maps in the whole architecture by a factor α. We decided to vary the width multiplier in four steps between 0.35 and 1.0 (0.35, 0.5, 0.75, 1.0). Note that for $\alpha = 0.5$, the overall number of parameters in the resulting network is almost equal to the one of the network of our modified biternion net architecture. In order to apply the MobileNet v2 architecture to the task of upper body orientation estimation, we dropped the final fully-connected layer and added the fully-connected stage, used in our modified biternion nets architecture as well.

1) Training: The training procedure is almost equal to the one described in Sec. IV-B.1, except of a small weight decay of 0.00005, which was added to penalize large weights. However, during the first experiment set, we noticed that higher learning rates often lead to better results. Therefore, we included another three higher initial learning rates (0.03, 0.04, 0.05). For the MobileNet v2 architecture, we used the publicly available pretrained weights on the ImageNet data set [35] to initialize the weights in the first stage. For depth information as input, we summed up all three weights for each position and output channel, which is equal to triplicate the depth information and to feed them in as a three-channel input. In addition, since fine tuning the MobileNet v2 architecture with a small number of samples using SGD might be challenging, we also tried the Adam optimizer [36] with initial learning rates of 0.0001, 0.001, 0.01, 0.02, and 0.03, respectively.

2) Results: The results of our second extensive set of experiments are depicted in Fig. 4. Furthermore, important results for each architecture are summarized in Tab. IV.

TABLE V: Mean absolute error in degrees (each averaged over 3 runs) obtained on test set by applying the MobileNetv2 architecture using either SGD or Adam as optimizer. The overall best result is printed in bold.

	Depth				RGB			
α	1.0	0.75	0.5	0.35	1.0	0.75	0.5	0.35
SGD	5.87	5.89	5.94	6.31	10.57	10.16	11.02	11.97
Adam	**5.35**	5.43	5.79	6.38	8.13	8.66	9.58	11.52

Comparing our modified biternion net architecture to the original architecture, it turns out that the modified one leads to better results. Even more, for depth information (see Fig. 4 left), the modified biternion net architecture beats its MobileNet v2 ($\alpha = 0.5$) counterpart with similar number of parameters and can compete with the more complex MobileNet v2 with $\alpha = 1.0$. For RGB images (see Fig. 4 right), the margin between both biternion net architectures becomes even larger, but they do not reach the performance of the MobileNet v2 architecture. Tab. V compares the best networks based on the MobileNet v2 architecture. It turns out that the performance increases while the depth multiplier increases. Furthermore, optimizing using Adams constantly leads to better results. However, initial trials for optimizing the biternion net archtitecture using Adam as well did not lead to promising results.

Similar to previous experiments, depth information clearly outperform RGB by a large margin, even for MobileNet v2, which was designed especially with regard to RGB images. We assume that estimating the upper body orientation from RGB images is just more challenging. The results in Fig. 4 (right) support this assumption as the most complex architectures (MobileNet v2 with $\alpha = 0.75$ or 1.0) lead to the best results. Hence, if available, depth information should be preferred for fast orientation estimation.

The best networks (MobileNet v2 with $\alpha = 1.0$) achieve a mean absolute error of $5.17°$, $5.43°$ and $5.44°$, respectively. However, the MobileNet v2 archtitecture with $\alpha = 1.0$ has more than three times as many parameters and requires three times as long for inferring a single sample as the second best networks (modified biternion net architecture with *large* inputs), which achieve a mean absolute error of $5.51°$, $5.28°$ and $5.52°$, respectively. In favor of fast inference and the opportunity to apply dropout sampling during inference as well (see Sec.V), we decided to stick to the modified biternion net architecture.

D. Best Network Analysis

Since the best configuration for our network has been selected, we turn to a deeper analysis on the test set. In Fig. 5 (left), the performance of our best network is visualized using a confusion matrix. It can be seen that the main diagonal is densely occupied, and that there is no dominating orientation, even though our training set is not completely balanced with respect to the ground truth angles. The mean absolute error as a function of the sensor distance strongly correlates with the distances covered by our data set (Fig. 5 right). Therefore, we conclude that recording further samples covering larger distances could further improve the quality of our approach.

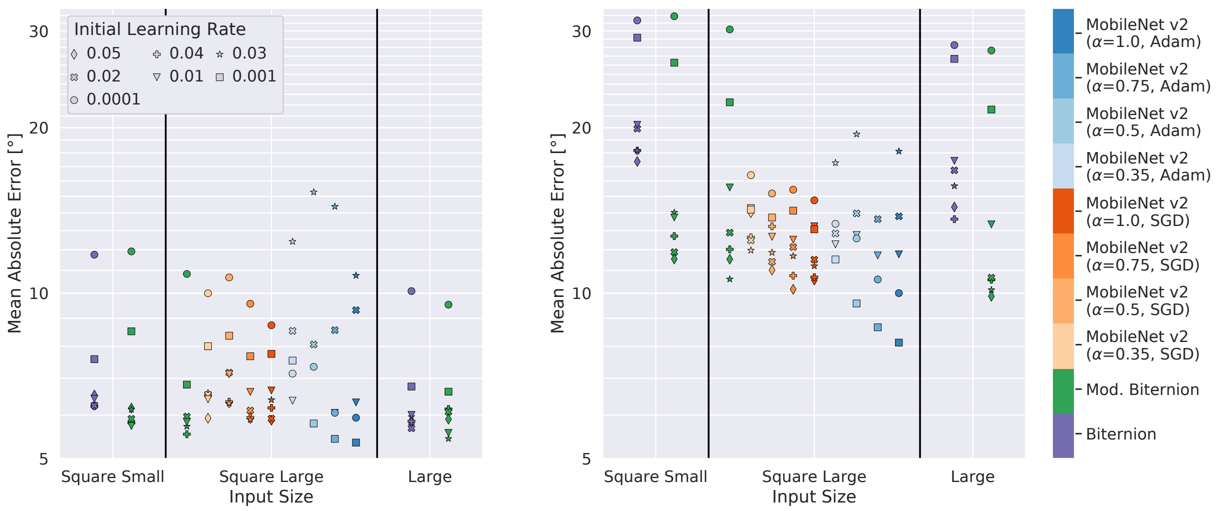


Fig. 4: Mean absolute error in degrees obtained on test set for different input sizes, architectures, and initial learning rates on depth inputs (left) and RGB inputs (right). Each marker represents results averaged over 3 runs.

Furthermore, we analyzed the error per person as shown in Fig. 6 (left). While the mean absolute error for most persons is close to the overall mean, person 13 is an exception. We took a look at the samples that received the largest errors (examples are shown in Fig. 6 right) and found that most of the large errors result from unusual poses. In particular, person 13 is often shown while swinging around a large handbag. We assume that our network does not generalize well on such uncommon appearances.

E. *Inference Time*

Since our robot navigates through a supermarket during opening hours, it is required that the orientation estimation runs in real time. Therefore, we measured the inference time of our network on three different devices suitable for use on a mobile platform: NVIDIA Jetson TX2, NVIDIA AGX Xavier and an Intel Core i7-7700H CPU. To further speed up inference, we deployed three different versions: a pure Tensorflow graph, a frozen Tensorflow graph and an optimized graph using NVIDIA TensorRT [37]. The frame rates for all platforms are shown in Tab. VI. For predicting a single image patch, all platforms enable inference in real time, even the pure CPU implementation (see first column in Tab. VI). Thus, we are able to enhance the capability of our network by applying additional sampling during inference in order to estimate the network's uncertainty as well (see Sec. V-B)).

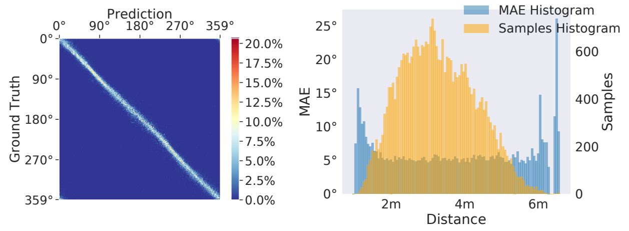


Fig. 5: Left: Confusion matrix with 1° bin division of our best network. Right: Histograms for the mean absolute error (left y-axis) and the amount of samples (right y-axis) as a function of the distance to the sensor (x-axis).

V. Robotic Application

With the findings of our experiments at hand, we deploy our network for orientation estimation to our mobile robot and compare its performance to other suitable approaches. So far, the network was always fed directly with image patches from our data set. These patches are free from background and cropped to persons. However, data captured by the Kinect2 mounted on our robot must be preprocessed in order to become background free and cropped. To do so, we deployed the 3D person detector presented in [5]. This approach is designed to enable real-time detection rates even on a single CPU core of a mobile platform. It operates on point clouds and outputs a point cloud cluster for each detected person. After detection, we back-project each cluster onto the image plane of the Kinect2 and then crop the projection to the encasing bounding box of the human. The resulting image patches are free of background objects and are input for our network. The whole procedure is visualized in Fig. 1 and in the accompanied video to this paper.

A. *Quantitative comparison to other Approaches*

At first, we compare our system to other modern approaches suitable for estimating or extracting the upper body orientation as continuous angle on our test set. In particular, we include the regression approach of [21] based on 3D point clouds and the skeleton estimation methods [9] and

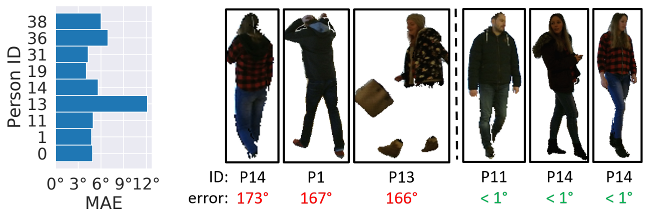


Fig. 6: Left: Mean absolute error per person on our test set. Right: The three samples with the largest and the three ones with the lowest absolute error including person ID.

TABLE VI: Frames per second on different computing devices with respect to different batch sizes. Abbreviations: TF - pure Tensorflow graph, TFF - frozen Tensorflow graph, TRT - graph optimized using NVIDIA TensorRT.

		Single Prediction	Sampling Batchsize 20	40	60	80	100
Jetson TX2	TF	262.9	39.9	20.8	14.1	10.6	8.6
	TFF	361.8	33.4	17.2	11.6	8.7	7.0
	TRT	520.5	69.1	36.2	24.6	18.3	14.6
Jetson Xavier	TF	261.9	116.4	66.2	47.1	35.2	29.1
	TFF	424.9	120.8	65.9	45.3	34.7	28.2
	TRT	675.8	177.5	101.0	69.2	53.4	43.9
i7	TF	222.0	9.6	4.7	3.1	2.3	1.9
	TFF	317.0	10.7	5.3	3.5	2.6	2.1

[10] as references. In contrast to our previous experiments, we now apply the preceding person detector to each frame for patch extraction since the mentioned approaches do a detection in the whole input image as well. As each approach produces some false negatives, we report results on a subset of the test data based on a shared blacklist only. We also ignore false positive detections since we focus on orientation estimation. For the skeleton approaches, we extract orientations by calculating the cross product of the shoulder-to-shoulder and center-hip-to-shoulder-vector and projecting the resulting vector onto the ground plane. Results of this comparison are shown in Tab. VII. Note that [21] is the only reference trained on our data because our data set does not provide skeleton labels. This limits the value of the comparison for [9] and [10]. However, we want to give an insight on how modern skeleton estimation approaches perform on orientation tasks. The results indicate that it is not necessary to rely on expensive skeleton computation if only the upper body orientation is of interest. Estimating orientation directly is much faster and significantly accurate in the context of navigation tasks. Furthermore, deep learning seems to outperform classical approaches like [21]. Tab. VII also shows that switching from perfectly masked image patches to patches from a detector causes a slight decrease

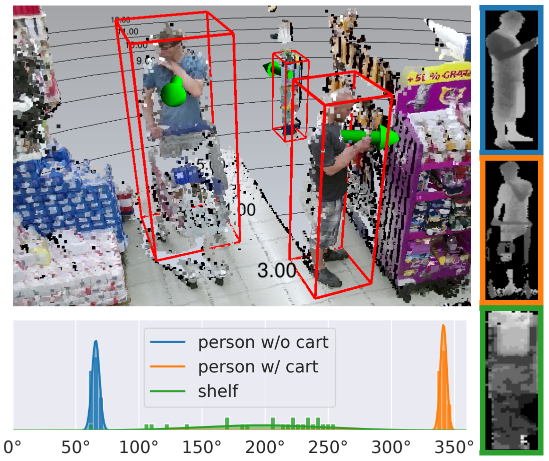


Fig. 7: Left: Point cloud person detections including one false positive and corresponding sampling distributions. Orientation estimates are displayed with green arrows. Right: Depth patches of back-projected point clusters.

TABLE VII: Comparison to other approaches. [9], [10] and our networks run on a NVIDIA AGX Xavier. Runtimes (in frames per second) include detection and orientation estimation. For [21] and our pipeline we used the person detector of [5].

approach	[9]	[10]	[21]	Mod. biternion	MobileNet v2
MAE	23.69	14.69	14.76	6.64	**6.14**
Runtime	0.89	1.65	10.78	**13.43**	13.11

in performance.

B. Sampling during Inference and Qualitative Results

Modeling uncertainty can be useful for certain tasks. One obvious application is to skip uncertain estimations, e.g. in subsequent navigation modules of a robot. Another one is temporal tracking of orientations using a Kalman filter to achieve a more robust and combined estimate.

However, by default most neural networks are not capable to represent uncertainty. In [38] a link between dropout and approximated Bayesian inference is examined. According to this, it is possible to obtain uncertainty information by applying dropout at inference time as well. Luckily our network makes use of dropout before both fully-connected layers comprising most of the weights. Hence, we can easily model uncertainty by enabling dropout during inference and replicating the input patch several times. In order to obtain meaningful uncertainty estimates, we followed the recommendations in [39], [40] and used 100 stochastic forward passes for each patch. Nevertheless, 25 passes already seem to perform similarly. As shown in Tab.VI, we can still enable real-time inference. However, since dropout is applied in the second stage only, the inference speed could further be optimized by separating the two stages of the network and replicating only the input for the second stage.

Fig. 7 shows an example using the sampling approach. For each detection, a normal distribution for the orientation was calculated out of 25 samples. Obviously, the estimation for the non-human object is very uncertain. This shows, that the additional uncertainty information might also be used as an indicator for false positive person detections. Applying sampling to all true and false positive detections obtained by our detector pipeline on the test set proofs this claim, as shown in Fig. 8.

In addition, we have observed qualitatively that our approach provides sufficiently accurate orientation estimations for our navigation tasks up to 10 meters distance, although the training set contains samples up to 6 meters only. Typical failure cases are squatting persons and heavy occlusions in

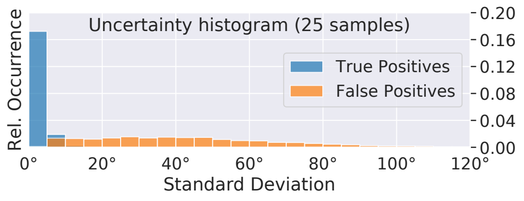


Fig. 8: Relative occurrence of angular standard deviations when sampling over true and false detections on our test set.

the upper body area. Both cases are not covered by our data. We believe that sampling is very helpful in these situations too, but it remains future work to further examine all effects.

VI. Conclusion

We have presented a system for estimating the continuous upper body orientation of persons using deep learning. To determine appropriate network architectures, input types and output encodings, we have conducted a comprehensive series of experiments. Due to the application of lightweight networks and the processing of fixed size image patches, our system runs in real time on a mobile platform, even on the CPU only. Furthermore, the fast runtime enables dropout sampling during inference for detecting false positives and enhancing subsequent tasks. For training and evaluations, we recorded an RGB-D data set with 37 persons. In total, it consists of more than 100,000 samples with precise continuous ground truth labels. Our code, the network weights, and our data set have been made publicly available to the community.

Since typical failure cases on our robot are mostly caused by the preceding person detector, e.g. when detections are missing or extracted patches are over or under segmented, it remains future work to improve this part of the overall system. We see great potential in using multi-task learning, for combining person detection and orientation estimation in a single neural network operating on raw patches directly.

References

[1] H.-M. Gross, *et al.*, "TOOMAS: Interactie shopping guide robots in everyday use – final implementation and experiences from long-term field trials," in *Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 2005–2012.

[2] P. Papadakis and P. Rives, "Binding human spatial interactions with mapping for enhanced mobility in dynamic environments," *Autonomous Robots*, vol. 41, no. 5, pp. 1047–1059, 2017.

[3] X.-T. Truong and T. D. Ngo, "To Approach Humans?: A Unified Framework for Approaching Pose Prediction and Socially Aware Robot Navigation," *IEEE Trans. on Cognitive and Developmental Systems*, vol. 10, no. 3, pp. 557–572, 2017.

[4] A. Krizhevsky, *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[5] B. Lewandowski, *et al.*, "A fast and robust 3d person detector and posture estimator for mobile robotic applications," in *Int. Conf. on Robotics and Automation*, 2019, pp. 4869–4875.

[6] L. Beyer, *et al.*, "Biternion nets: Continuous head pose regression from discrete training labels," in *German Conference on Pattern Recognition*, 2015, pp. 157–168.

[7] M. Sandler, *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[8] Z. Cao, *et al.*, "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 1302–1310.

[9] D. Tome, *et al.*, "Lifting from the deep: Convolutional 3d pose estimation from a single image," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 5689–5698.

[10] C. Zimmermann, *et al.*, "3D Human Pose Estimation in RGBD Images for Robotic Task Learning," in *IEEE Int. Conf. on Robotics and Automation*, 2018, pp. 1986–1992.

[11] T. Schnürer, *et al.*, "Real-time 3d pose estimation from single depth images," in *Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics - Theory and Applications*, 2019, pp. 716–724.

[12] C. Weinrich, *et al.*, "Estimation of Human Upper Body Orientation for Mobile Robotics using an SVM Decision Tree on Monocular Images," in *Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 2147–2152.

[13] L. Fitte-Duval, *et al.*, "Upper body detection and feature set evaluation for body pose classification," in *Int. Conf. on Computer Vision Theory and Applications*, 2015, pp. 439–446.

[14] F. Shinmura, *et al.*, "Estimation of Human Orientation using Coaxial RGB-Depth Images." in *Int. Conf. on Computer Vision Theory and Applications*, 2015, pp. 113–120.

[15] L. Fitte-Duval, *et al.*, "Combination of rgb-d features for head and upper body orientation classification," in *Advanced Concepts for Intelligent Vision Systems*, 2016, pp. 591–603.

[16] J. Choi, *et al.*, "Human body orientation estimation using convolutional neural network," *arXiv preprint arXiv:1609.01984*, 2016.

[17] M. Raza, *et al.*, "Appearance based pedestrians head pose and body orientation estimation using deep learning," *Neurocomputing*, vol. 272, pp. 647–659, 2018.

[18] Y. Kohari, *et al.*, "CNN-based Human Body Orientation Estimation for Robotic Attendant," *Workshop Robot Perception of Humans*, 2018.

[19] B. Ahn, *et al.*, "Real-time head orientation from a monocular camera using deep neural network," in *Asian Conf. on Computer Vision*, 2014, pp. 82–96.

[20] Y. Kawanishi, *et al.*, "Misclassification tolerable learning for robust pedestrian orientation classification," in *Int. Conf. on Pattern Recognition*, 2016, pp. 486–491.

[21] T. Wengefeld, *et al.*, "Real-time person orientation estimation using colored pointclouds," in *to appear: Europ. Conf. on Mobile Robots*, 2019.

[22] M. Simon, *et al.*, "Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds," in *Computer Vision – ECCV 2018 Workshops*, 2019, pp. 197–209.

[23] P. Sudowe, *et al.*, "Person Attribute Recognition with a Jointly-Trained Holistic CNN Model," in *IEEE Int. Conf. on Computer Vision Workshop*, 2015, pp. 329–337.

[24] W. Liu, *et al.*, "Accurate estimation of human body orientation from rgb-d sensors," *IEEE Trans. on Cybernetics*, vol. 43, no. 5, pp. 1442–1452, 2013.

[25] Advanced Realtime Tracking GmbH. (2018) ART DTrack2. [Online]. Available: https://ar-tracking.com/

[26] K. Pentenrieder, *et al.*, "Analysis of tracking accuracy for single-camera square-marker-based tracking," in *Workshop Virtuelle und Erweiterte Realität*. Citeseer, 2006.

[27] K. He, *et al.*, "Identity mappings in deep residual networks," in *Europ. Conf. on Computer Vision*, 2016, pp. 630–645.

[28] S. Xie, *et al.*, "Aggregated residual transformations for deep neural networks," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 5987–5995.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. on Learning Representations*, 2015, pp. 1–14.

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Int. Conf. on Machine Learning*, 2015, pp. 448–456.

[31] N. Srivastava, *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[32] K. Mardia and P. Jupp, *Directional Statistics*, ser. Wiley Series in Probability and Statistics. Wiley, 2009.

[33] M. Abadi, *et al.* (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. [Online]. Available: www.tensorflow.org

[34] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Int. Conf. on Learning Representations*, 2016.

[35] J. Deng, *et al.*, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[36] J. Duchi, *et al.*, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[37] NVIDIA Corp. (2018) NVIDIA TensorRT. [Online]. Available: https://developer.nvidia.com/tensorrt

[38] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Int. Conf. on Machine Learning*, 2016, pp. 1050–1059.

[39] Y. Gal, *et al.*, "Deep bayesian active learning with image data," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[40] D. Seichter, *et al.*, "How to improve deep learning based pavement distress detection while minimizing human effort," in *IEEE Int. Conf. on Automation Science and Engineering*, 2018, pp. 63–70.