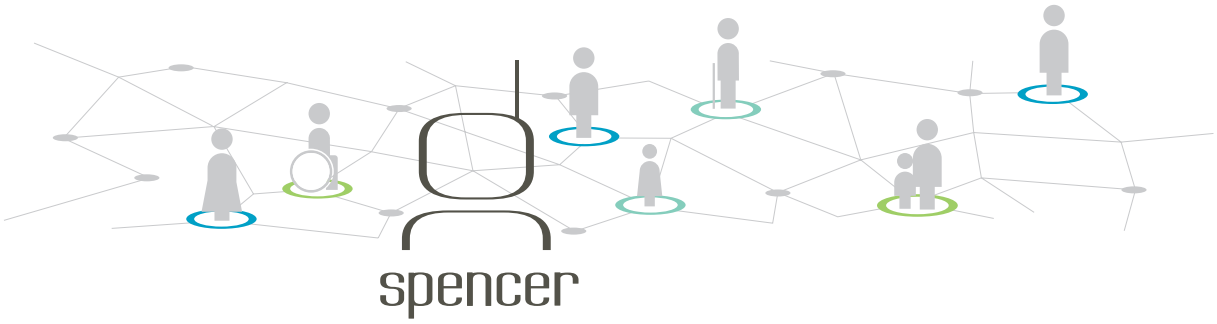


Grant agreement no: FP7-600877

SPENCER:

Social situation-aware perception and action for cognitive robots

Project start: April 1, 2013

Duration: 3 years

DELIVERABLE 3.3

Integrated on-line system for detailed multi-person analysis

Due date: month 34 (January 2016)

Lead contractor organization: RWTH

Dissemination Level: PUBLIC

Contents

Abstract

In this deliverable, we present the basic structure and the individual components of the integrated on-line system for detailed multi-person analysis developed at RWTH. The goal of this system is to perform a fine-grained analysis of persons detected and tracked by the different modules from WP2 with the aim of inferring their rough body posture, their head pose and viewing direction, and their articulated upper-body pose. This information will be of use for the group-level analysis in WP4.

An early iteration of this multi-person analysis system was already sketched in D3.2. In this deliverable, we now present the final version of the different system components, as well as the architecture in which they interact.

1 Introduction

The goal of WP3 is to deliver detailed information about people in the close vicinity of the SPENCER robot in order to enable the robot to interpret people's current actions and interactions. We have already presented a first integrated prototype for this task in D3.2. In this deliverable, we now report on an extended and improved version of this system that includes the final components for the different estimation tasks to be used in the SPENCER deployment.

Figure 1 gives an overview of the data flow in our integrated multi-person analysis system. As the SPENCER robot is moving through its target environment, people in its vicinity will be detected by the RGB-D and laser based detection modules (green) and will then be tracked by the tracker module (blue) from WP2. A planner module (yellow) selects tracks that are visible in the field of view of the RGB-D sensors and for which the different analysis components shall give a detailed analysis (red) of head pose (Sec. 2), rough posture (Sec. 3) and articulated upper body pose (Sec. 4). When selected, each of those modules is passed a task-specific region extracted from the RGB-D video input with fixed offsets relative to the bounding box provided by the tracker. Since each analyzed detection bounding box is part of a person's tracked trajectory, the information from our different analysis modules is directly associated to an individual and can be integrated over time.

Compared to the first prototype version of this system in D3.2, we now have extended and improved components for each of the analysis modules available that have also been optimized for resource efficient real-time operation on the SPENCER robot. Each analysis module can be called through ROS messages for a given detector/trackor bounding box and is then applied independently from the others. In the following sections, we describe each of the modules in detail.

For two of the modules – the head pose estimation approach from Sec. 2 and the articulated upper body pose analysis from Sec. 4 – research papers have been published during Y2 that we append to this deliverable [1, 2]. In addition, Section 4.5 will give an outlook to a recently developed deep learning component for body pose estimation which achieves very promising results and which may eventually replace the current system component in a future iteration of the robot. We are currently preparing an arXiv pre-publication of this approach that will be available in time for the SPENCER Y3 review meeting.

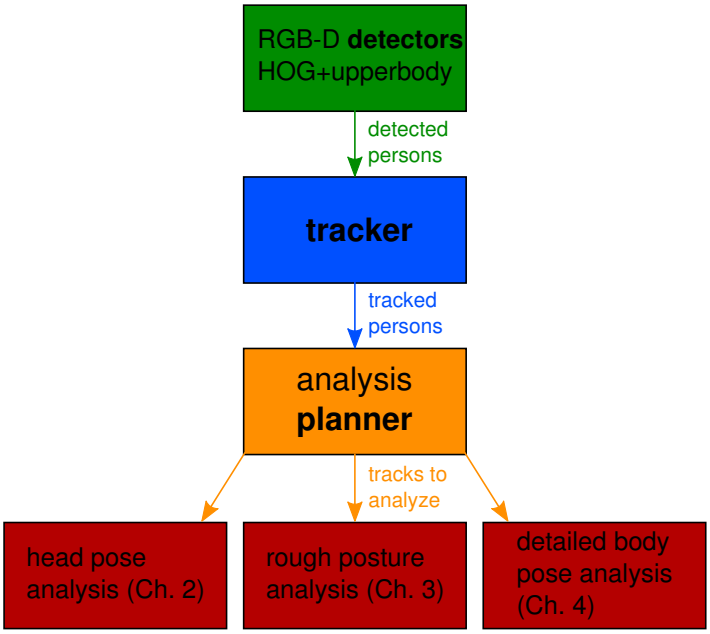


Figure 1: Overview of the data flow for detailed multi-person analysis. After the detection (green) and tracking (blue) pipeline in RGB-D, a set of tracked persons is available. A planner module (yellow) selects tracks for which the different analysis components shall give a detailed analysis (red) of head pose (Sec. 2), rough posture (Sec. 3) and body analysis (Sec. 4). Of course, the analysis can be called for all available tracks, but in case of computational constraints a subset might be selected. Criteria for this selection process involve, *e.g.*, the distance of the tracked person to the robot or whether the person is facing towards or away from the robot.

2 Head Pose Analysis

2.1 Approach

As planned in D3.2, we have now replaced the "WARCO"-based [3] head pose classification module by a novel regression module called "BiternionNet", which we published at GCPR'15 [1]. A BiternionNet is a deep convolutional neural network with a novel output layer specifically tailored to the regression of orientations. This "biternion" layer is a projection onto the two-dimensional unit circle, thus corresponding to the sine and cosine decomposition of an angle. Combined with a corresponding von-Mises optimization criterion, this output layer was shown to be vastly superior to the typically used one-dimensional regression and Euclidean distance criterion, as can be seen in Table 1 (in an evaluation on the TownCentre dataset from [4]).

In [1], we first showed that the deep network architecture used in BiternionNets surpasses the performance of the original WARCO method on every single benchmark dataset in [3] (see Table 2). We then further demonstrated that the biternion output layer makes it possible to learn a continuous regression output (shown in Fig. 2) using only discrete 8-bin annotations, which are easy to create. Finally, as is usual in deep-learning approaches, the implementation supports computation on the GPU, thus freeing the load of all but one CPU core. The prediction-rate is real-time and almost

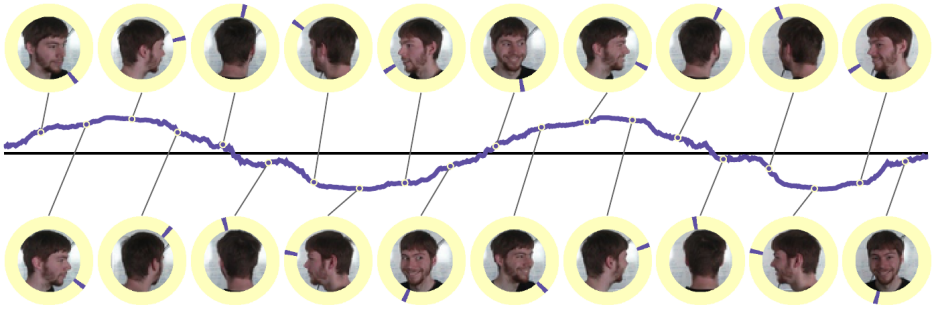


Figure 2: Head pose regressions from our "BiternionNet" for an unseen person turning on the spot. The horizontal axis represents time, the vertical axis the cosine of the predicted orientation.

Method	MAE
Linear Regression	$64.1° \pm 45.0°$
Naive Regression	$38.9° \pm 40.7°$
Von Mises	$29.4° \pm 31.3°$
Biternion	$21.6° \pm 25.2°$
Biternion+Von Mises	$\mathbf{20.8}° \pm \mathbf{24.7}°$
Benfold&Reid [4]	$25.6°$ / $64.9°$

Table 1: Quantitative regression results for the TownCentre dataset [4].

independent of the number of tracked people due to efficiently batched computation.

2.2 Interface

Over the course of multiple integration meetings, it has become apparent that the head pose estimation module would be much more useful if it made predictions based on person tracks, as opposed to upper-body detections as it did so far. The new BiternionNet component takes this into account and gives predictions corresponding to tracks, *i.e.* the messages specified in `perception/spencer_tracking_msgs`.

2.3 Adaptation for Schiphol Scenario

A few minor adaptations were necessary in order to run the BiternionNet approach in the Schiphol deployment environment. Due to the unique lighting conditions at Schiphol, we needed to augment the training set with additional examples. During the fourth integration meeting, we have therefore recorded 18 people in large range of different orientations on-site, resulting in almost 9300 training images. Because the tracker has a tiny lag, we had to crop heads to a size of 80×50 as opposed to the 50×50 as used in [1]. These wider crops include a considerable amount of background, making training with limited data more difficult.

As an initial evaluation, we have trained a BiternionNet on the data collected at Schiphol and looked at its predictions for a person not seen during training. Since the only labels we have are

	HIIT	HOCoffee	HOC	QMUL	
# Samples	12 000/12 007	9522/8595	6860/5021	7603/7618	9813/8725
# Classes	6	6	4	4	4 + 1
Tosato *et al.* [3]	96.5%	81.0%	78.69%	94.25%	91.18%
Lallemand *et al.* [5]	-	-	79.9%	-	-
Our CNN	**98.70%**	**86.99%**	**83.97%**	**95.58%**	**94.30%**

Table 2: Class-average accuracies on the four classification datasets from [3]. The sample counts refer to the provided train/test splits. We obtain state-of-the-art results on all datasets.

8-bin class-labels (see [1]), we discretize the predictions for the purpose of the evaluation and look at the classification error. With this setup, the network reached an average accuracy of 64.5% over four clips of that person. When also accepting the orientation segments neighboring the ground-truth label as correct in order not to penalize class overlap, the BiternionNet reaches a respectable average accuracy of 96.8%. (Note that the BiternionNet is not optimized for classification performance; it is just the only way we are able to measure performance given the available labels). Those results show that the Biternion net already learns a good representation, but that its performance can probably still be further improved through a better adaptation to the Schiphol scenario.

In the remaining time before the final deployment, we will therefore further extend the training set by additional examples collected in Schiphol with the child stroller setup during a previous visit (see D3.1), taking the available track annotations as an indirect cue to predict likely head orientations (similar in spirit to [4]). In addition, we are currently working on a background-subtraction preprocessor to ease the network's learning, in particular with respect to the less accurate alignment.

3 Rough Posture Analysis

In the same way in which BiternionNets replaced the previous head-orientation module, we believe they also lend themselves very well to the estimation of full-body orientation. All that is necessary is a small change in the architecture, since full-body images have a vastly different aspect-ratio than heads. Besides this minor change, we can essentially reuse all components developed as well as all data collected for the head-pose component.

We have verified this idea by comparing BiternionNets to the "WARCO" approach on the HOC dataset from [3], which contains fully-body person detections similar to our application scenario. As Table 2 shows, BiternionNets indeed perform better. We therefore propose to use BiternionNets also for body orientation and coarse posture analysis.

3.1 Adaptation for Schiphol

The main issue with adapting BiternionNets for pose estimation is to collect a sufficient amount of training data under the specific conditions of the application scenario. We are currently in the process of preprocessing and annotating our Schiphol recordings for full-body orientation.

3.2 Interface

We decided to move the prediction onto the output of the tracking system as opposed to detections, for the same reasons mentioned in Subsection 2.2.

4 Detailed Body Pose Analysis

The task of the detailed body pose analysis module is to estimate a given person's articulated body pose (in terms of a set of 3D skeleton joint positions) from depth images.

4.1 Regression Forest Approach

As already described in D3.2, we have implemented a regression framework for pose estimation that is based on regression forests [6]. Briefly stated, this approach uses depth information of all the pixels belonging to a person and predicts a non-parametric distribution over 3D position of each body joint belonging to the upper body of a person. At test time each tree in the regression forest considers the neighborhood around a depth pixel q. At each internal tree node, it evaluates a binary splitting function based on the pixel's neighborhood using the depth comparison features from [7]. Depending on the outcome of this splitting function, evaluation proceeds to the left or right child node. Once a leaf node is reached, votes are cast for possible locations of the all upper body joints according to the vote distributions stored at the node.

This approach provides good pose estimates as long as people are fully visible and well separated from each other, but similar to the original work from [6], it quickly breaks down when scene occlusions occur (which will often be the case in the SPENCER airport scenario). We have therefore extended the pose estimation approach for explicitly handling occlusions. The main idea behind our proposed extension, called Occluder Aware Regression Forests (OARF), is to learn the depth appearances of common occluders along with the depth appearance of body parts, so that we can (a) distinguish those occluders from regular body parts (and thus avoid false positive matches) and (b) use the known occlusion patterns to predict the 3D positions of occluded joints. This extension was also already initially described in D3.2 and has led to a publication at the CVPR'15 ChaLearn Workshop [2], which we append to this deliverable.

Figure 3 shows some qualitative results of OARF on example images from our test set, compared to the joint predictions delivered by the Kinect SDK. We present results using two different versions of OARF: one that has been trained to recognize different categories of occluders ("OARF with semantics", top row) and one that does not differentiate between occluders ("OARF without semantics"). As can be seen from those results, OARF can achieve better pose estimation results than the Kinect SDK in occluded cases. This finding was also verified in quantitative experiments in our appended paper [2].

Altogether, those results are very encouraging. They show that the OARF idea clearly has potential to improve body pose estimation accuracy under occlusion.

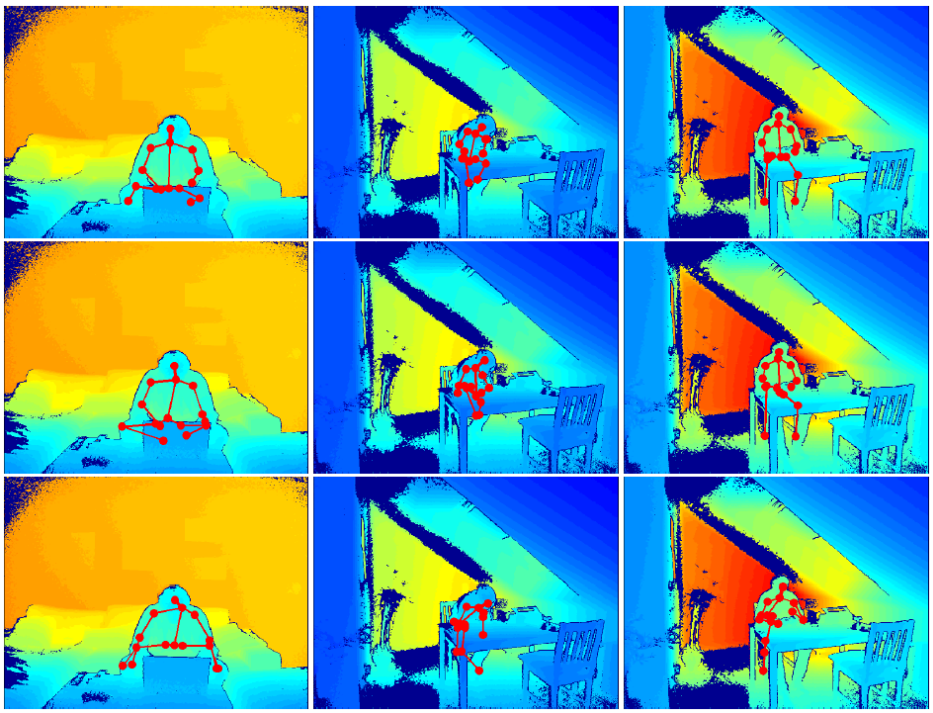


Figure 3: Qualitative results for some sample images taken from our real test set [2]. Top row shows the body joints predicted by OARF with semantics. Middle row shows the body joint predictions from OARF without semantics. Bottom row shows the body joints predicted by the Kinect SDK. The results show that integrating knowledge about occluding objects helps in improving the body joint prediction performance for the occluded joints.

4.2 Random Walk Forests

Still, the regression forest based approach for detailed Body Posture Analysis described in the previous section has two major limitations, particularly with respect to its use in SPENCER project.

First, it requires well segmented persons as a preprocessing step to provide detailed body posture analysis which is not feasible for the scenarios encountered in SPENCER project. In SPENCER, the only available option for person segmentation is the output of the upper body detector, which only provides a bounding box around head and shoulders. This input is very limited, as it does not contain any information about the segmentation of upper body limbs, and thus the regression forest performs poorly for predicting the 3D positions of joints belonging to upper body limbs. Second, our efficient implementation of the approach relies on parallel computation using either a GPU or a multi-core CPU which makes it less feasible for SPENCER, where the hardware resources are very much constrained. For these reasons, we propose to use a simpler approach instead, inspired by a recent publication of [8]. This new approach, based on Random Walk Forests, is extremely fast, runs on a single CPU core, but is still very effective.

A Random Walk Forest consists of regression trees, where each tree is trained to be an expert on a single body joint only. Given a random position on the upper body, each tree outputs a direction

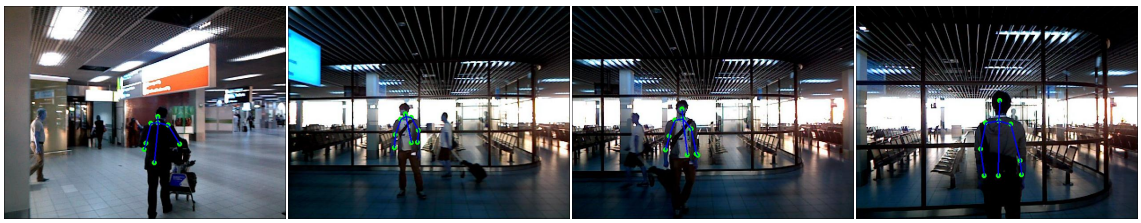


Figure 4: Qualitative results for some frames selected from the Schiphol data.

where a particular joint could be located. Then a step is taken in that direction on the upper body with a certain step size, resulting in a new position on the upper body. This procedure is repeated a for a number of steps. The final joint position is then the expected value of all the positions reached during this procedure. The number of iterations and the step size are hyper-parameters, which are learned by cross validation.

Each tree is trained on the set of training samples $Q = \{(q, o, D, u_j)\}$, each consisting of a sampled pixel q in a training image D, its offset o to joint j and a unit direction vector u_j to the joint j from pixel q. This direction vector is computed as $u_j = (p_j - o)/\|p_j - o\|$ where p_j is the 3D position of joint j. During training of each regression tree, the training pixels q at a split node are partitioned into left and right subsets by using the depth comparison features from [7] and the quality of the split is measured using the information gain criteria and the best split is selected. This procedure is continued until the maximum allowed depth for a tree is reached. In order to insert randomness into direction selection, all the pixels q that have reached a particular leaf node l are further clustered using mean shift clustering. Then during inference a cluster center is chosen randomly.

During inference, a test pixel q is traversed through the tree until it reaches a leaf node l, where a direction vector u_j to joint j is randomly selected. Then by using the learned step size $dist_s$ a new pixel q is chosen that is again traversed through the same tree to get a new direction. This procedure is repeated N times. The final joint position is the expectation of the random walk $\hat{q}$.

4.3 Evaluation on SPENCER data

The property of the Random Walk Forest (RWF) approach that it only requires a single point on the upper body and not already a detailed segmentation including limbs makes it very attractive for use in the SPENCER scenario, where we can use the center of an upper body detection as a starting point.

We evaluated RWF on the SPENCER data recorded at Schiphol Airport. Some qualitative results are shown in Figure 4. The results are produced by only applying RWF on a frame-by-frame basis and without using any limb-length and temporal smoothing constraints. We trained our RWF models by using synthetic data that was generated by the procedure described in D3.2. As we do not have upper body skeleton ground truth annotations available for Schiphol data yet, we cannot report quantitative results for this scenario at this point. Qualitatively, the results look already very promising, though, and we expect that they will further improve through the planned addition of limb length and temporal smoothing constraints.

4.4 Interface

The current ROS node for upper body skeleton subscribes to the following messages :

`/spencer/perception_internal/people_detection/rgbd_rear_top/upper_body_detector/detections`

`/spencer/perception_internal/people_tracking/rgbd_rear_top/tracked_persons`

`/spencer/perception_internal/people_tracking/rgbd_rear_top/tracked_persons_2d`

The goal here is to have a priority mechanism that automatically selects people for which upper body skeletons should be provided by fusing information from incoming messages. For example, in an airport scenario, an input frame may contain many people, but the most important ones are those that are walking towards the robot and are close to the robot. Therefore, the ROS node for upper body skeleton estimation has an implicit perception controller that assigns priority to people based on information received from incoming messages (currently position and orientation of people provided by the people tracker) and that passes the prioritized upper bodies to the upper body skeleton detector to get skeletons.

As output, the upper body skeleton detector provide a list of 3D positions of 9 upper body joints, namely `head`, `neck`, `shoulders`, `elbows` and `wrists`. The upper body ROS node publishes the following message

`/spencer/perception_internal/detailed_upper_body_analysis/rgbd_rear_top/upper_body_skeleton`

4.5 CNN based Skeleton Detector

A limitation of the RWF approach introduced in Section 4.2 is that it uses very weak hand-crafted features that are very fast to compute but require large amounts of training data to generalize well. In recent years, Convolutional Neural Networks (CNNs) have achieved impressive results for different computer vision problems. This is mainly due to their ability to learn hierarchical features that generalize well in many situations.

In order to leverage those advantages, we have started work on a CNN-based body pose estimation approach that is already producing very promising results. While this approach will not be ready for real-time operation on the SPENCER robot until the end of the project, it will provide an excellent basis for follow-up work that will lead to vastly improved visual capabilities for a future robot deployment.

Our approach is based on the Inception deep network architecture by [9], augmented with skip-layer fusion. We set up the network to produce output heat-maps for each joint. Formally, the network takes an input image I and produces a fixed-size $i \times j \times k$ cube as output, where k is the number of joints in the upper body. The network is trained on training data $N = \{I, y\}$, consisting of training images I with heat-maps y. During training, we minimize the mean squared error between ground-truth heat-maps y and predicted heat-maps $\hat{y}$. We make several improvements in the network design and training pipeline that lead to improved performance.

	Average Joint Accuracy	
	Ours	Tompson et al [11]
Joint		
Wrists	74.1%	63.4%
Elbows	78.6%	67.9%
Shoulders	85.4%	79.2%
Hips	85.2%	69.5%
Head	95.1%	90.6%
Knees	81.1%	71.0%
Ankles	77.7%	64.2%
Overall	82.5%	72.0%

Table 3: Per joint detection accuracy of 2D body joint predictions on the LSP dataset [10], measured using the strict evaluation measure from [12] with an error threshold of 0.2% of the upper body size).

In order to assess its performance, we evaluate our Deep CNN based skeleton regressor on the standard benchmark for 2D Human Pose Estimation, the "Leeds Sport" dataset (LSP) [10]. As Table 3 shows, even this early version of our approach already significantly outperforms the current state-of-art results by [11] using the strict *pck* evaluation measure from [12]. Some qualitative results are also shown in Figure 5. As can be seen from those examples, our approach achieves accurate results for a variety of poses and under vastly different lighting conditions (something that will be especially relevant for SPENCER). We are currently preparing an arXiv pre-publication on this work that will be available in time for the SPENCER review meeting, with plans of later submitting it to a vision conference.

5 Conclusion

In this deliverable, we have presented an integrated on-line system for detailed multi-person analysis for use in SPENCER. Our system comprises three component modules for head pose, coarse body posture, and articulated body pose estimation. Integration is achieved through well-defined interfaces with ROS message definitions that allow a perception planner module to pass tracked person bounding boxes to the detailed analysis modules for further processing. Each of the component modules has been evaluated on task-specific datasets and we have verified that each module achieves a sufficient performance level for this integration.

In the remaining weeks until the Y3 review meeting, we will use the data from the latest recording sessions at Amsterdam Schiphol airport in order to create adapted training sets that will further improve performance of the different analysis modules in the Schiphol scenario.

Figure 5: Qualitative upper body pose estimation results for some images from the LSP dataset [10].

References

[1] Lucas Beyer, Alexander Hermans, and Bastian Leibe. Biternion nets: Continuous head pose regression from discrete training labels. In *Pattern Recognition, Proceedings of GCPR 2015*, volume 9358 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 2015.

[2] U.Rafi, J.Gall, and B.Leibe. A Semantic Occlusion Model for Human Pose Estimation from a Single Depth image. In *CVPR15 Chalearn Looking at People Workshop*, 2015.

[3] D. Tosato, M. Spera, M. Cristani, and V. Murino. Characterizing Humans on Riemannian Manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1972–1984, 2013.

[4] Ben Benfold and Ian Reid. Unsupervised Learning of a Scene-Specific Coarse Gaze Estimator. In *International Conference on Computer Vision*, 2011.

[5] Joé Lallemand, Alexandra Ronge, Magdalena Szczot, and Slobodan Ilic. Pedestrian Orientation Estimation. 2014.

[6] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient Regression of General-Activity Human Poses from Depth Images. In *International Conference on Computer Vision*, 2011.

[7] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time Human Pose Recognition in Parts from Single Depth Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[8] H.Y.Jung, S.Lee, Y.S.Heo, and I.D.Yun. Random Tree Walk toward Instantaneous 3D Human Pose Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[9] C.Szegedy, W.Liu, Y.Jia, P.Sermanet, S.Reed, D.Anguelov, D.Erhan, V.Vanhoucke, and A.Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[10] S.Johnson and M.Everingham. Learning effective human pose estimation from inaccurate annotation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[11] J.Tompson, A.Jain, Y.LeCun, and C.Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Neural Information Processing Systems*, 2014.

[12] B. Sapp and B. Taskar. Modec: Multimodal decomposable models for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels

Lucas Beyer, Alexander Hermans, and Bastian Leibe

Visual Computing Institute, RWTH Aachen University

Abstract. While head pose estimation has been studied for some time, continuous head pose estimation is still an open problem. Most approaches either cannot deal with the periodicity of angular data or require very fine-grained regression labels. We introduce biternion nets, a CNN-based approach that can be trained on very coarse regression labels and still estimate fully continuous 360° head poses. We show state-of-the-art results on several publicly available datasets. Finally, we demonstrate how easy it is to record and annotate a new dataset with coarse orientation labels in order to obtain continuous head pose estimates using our biternion nets.

1 Introduction

The estimation of head poses is an important building block for higher-level computer vision systems such as social scene understanding, human-computer interfaces, driver monitoring, and security systems. For many of these tasks, a continuous head pose angle is arguably more useful than few discrete orientation classes as yielded by most current head pose systems [8,34,4].

While many face pose and gaze estimation methods have been covered in the literature, the task of regressing *head* pose is distinctly different in that it also handles people not facing the camera, resulting in poses spanning the full 360° spectrum. Thus, head pose estimators need to be able to cope with the periodicity of angular data, *i.e.* the fact that 361° corresponds to 1° and, for a head pose of 0°, a prediction of 359° is no worse than a prediction of 1°. Face pose and gaze estimators can conveniently sidestep this difficulty by constraining the prediction range to non-periodic intervals such as $[-90°, 90°]$. Another difficulty in learning a head pose regressor lies in obtaining enough training data with accurate regression labels [6,11]. All publicly available datasets, except [5], are either restricted to coarse orientation bins, or to the range of front-facing poses [6,14,2,15,9,10].

A multitude of approaches [25,32] has been proposed which solve only one of the two aforementioned problems: either they cannot cope with periodicity [26,24,34], or they need fine-grained regression data [35,33,16]. Since none of this is satisfactory, we propose a principled approach to solve both problems simultaneously.

Our approach is based on convolutional neural networks (CNNs), for which we propose a novel output layer embedding an angle into two dimensions, coupled

with a fitting cost function. It is able to handle fully periodic, continuous regression while *only requiring coarse, discrete class-labels* as training data, which are easily obtainable from video recordings. We call our approach *biternion nets*. Before demonstrating the effectiveness of the biternion output layer, we validate our CNN architecture on several publicly available datasets and show that it yields state-of-the-art results.

In summary, our contributions are threefold: (1) We present a CNN architecture that outperforms state-of-the art results on several public head pose datasets. (2) We propose a novel combination of output layer and cost function to elegantly solve the problem of periodic orientation regression, which we call *biternion nets*. (3) We show that we can learn *continuous* head-pose regression from discrete training labels. To demonstrate this, we present continuous regression results obtained from a biternion net trained on data recorded and annotated in less than two and a half hours.

2 Related Work

Head pose estimation has been a very active research field for the past 20 years [32,25]. Over time, authors have developed many different methods to approach this problem. The probably most popular direction is the functional mapping of images to a feature space where classifiers or regressors can directly be applied. These mappings range from simple gradient-based features [24,7,21], over covariance features [34], to learned functional mappings [26,33,4]. These approaches often result in a manifold embedding of the images [26,34]. However, if training data is sparse, it is hard to ensure the quality of these manifolds [19]. Another approach is to find facial landmarks, such as eye and mouth locations, and use these to determine the pose of a face [9]. It is also possible to use tracking information to get a good prior for the head pose [7,10]. Here, interactions between the body pose and the head pose can be exploited [5,8]. Several of these techniques have also been used for objects such as cars or chairs [33,28,18].

While some of these approaches work on high resolution images [14,2,10,12], the majority of them is based on low resolution images [26,24,5,34]. With the recent availability of cheap RGB-D sensors, depth information has also been used to improve head pose estimation [12].

The high activity within this field has resulted in a large number of different datasets for head pose estimation [6,14,2,1,15,5,34,9,10], most of which are face pose rather than head pose datasets and often only contain sparse head poses and fairly coarse orientation labels. As we are interested in continuous head pose estimations, most of these datasets are not suitable for our experiments.

Based on the available datasets, most approaches focus on coarse face poses, while only few *head* pose estimation approaches and datasets exist [35,5,34]. Wu and Toyama [35] estimate gradient distributions from 1024 different viewpoints and match new views to the nearest viewpoint to determine the pose. Benfold and Reid [5] use the walking direction obtained from unsupervised people tracking in a video sequence to train a regression forest for the head pose. Tosato *et al.* [34] use covariance features to classify head poses into a small set of orientation bins.

CNNs have also been used for orientation estimation before. Qi [28] fine-tunes a large pre-trained CNN to classify the orientation of chairs using a large amount of rendered chairs with precise labels. However, using CNNs pre-trained on ImageNet for low-resolution head pose estimation makes no sense due to the significantly different filter resolution, type of data, and learning task. Most similar to our approach is the one by Osadchy *et al.* [26], which also uses a CNN for continuous head pose estimation. They learn a face manifold on (non-public) data with regression labels, which enables them to jointly detect and estimate the pose of faces. In contrast to us, they focus on using face pose data to improve face detection and do not address the periodicity problem.

Some approaches also aim at solving the periodicity problem [33,16,18]. However, their approaches are typically based on nearest-neighbor matching or kernel density estimation, meaning that they require dense orientation labels for training. All three of the above approaches use fine grained face datasets [14,1] and it is unclear how well they could perform for head pose estimation.

To the best of our knowledge, only Huang *et al.* [19] aim at learning continuous regressors from a discrete face pose dataset. They learn a mixture of local tangent subspaces that are robust to regression regions with bad coverage in the training set. Their representation is based on HOG features and they use high resolution images. It is questionable whether their approach can deal with head poses, as HOG features are not very expressive for the back of a head. Furthermore, they do not evaluate how continuous their regression really is.

In conclusion, based on existing approaches, the task of continuous periodic head pose estimation is still unsolved. Here our approach comes into play.

3 CNNs for Head Pose Estimation

Throughout this paper, we work in the framework of deep convolutional networks and stochastic, gradient-based optimization. In this section, we present the specific network architecture we use for all experiments, changing only the output layer and cost function to match the task at hand. We then apply it to multiple publicly available datasets, consistently outperforming current state-of-the-art methods on those datasets.

3.1 The Network Architecture

We use a moderately deep, batch-normalized [20], VGG-style network architecture [30] consisting of six convolutional layers with 24, 24, 48, 48, 64 and 64 feature channels, respectively, followed by a single hidden layer of 512 units, and train it for a fixed duration of 50 epochs in all our experiments. For all details about the network and the training procedure, please refer to the supplementary material. We implemented the network in Theano [3] using IPython notebook [27]. All numbers reported within this paper are averages over five runs. While we will show that this architecture already performs very well, it is likely possible to reduce the error even further by using deeper networks with

Table 1: Class-average accuracies on the four classification datasets from [34]. The sample counts refer to the provided train/test splits. We obtain state-of-the-art results on all datasets.

	HIIT	HOCoffee	HOC	QMUL	
# Samples	12 000/12 007	9522/8595	6860/5021	7603/7618	9813/8725
# Classes	6	6	4	4	4 + 1
Tosato *et al.* [34]	96.5%	81.0%	78.69%	94.25%	91.18%
Lallemand *et al.* [21]	-	-	79.9%	-	-
Our CNN	**98.70%**	**86.99%**	**83.97%**	**95.58%**	**94.30%**

more careful regularization and a bag of other well-known tricks [23,13,36,29,17]. We do not further go down that road, since the goal of this section is simply to demonstrate the suitability of CNNs in general, and our architecture in particular, for predicting head poses on low-resolution images.

3.2 Experimental Validation

We use the collection of datasets provided by Tosato *et al.* [34] to validate our approach. First, we show results on those datasets that treat pose estimation as a classification task in Table 1. These datasets contain very rough pose bins, such as `Front`, `Back`, `Left` and `Right`, with the addition of `FrontLeft` and `FrontRight` for HIIT and HOCoffee, and `Background` for the 5-class version of the QMUL dataset.

In this case, the network's output layer is a softmax-layer and the cost being optimized is the negative log-likelihood. While the accuracies obtained by state-of-the-art methods are already high, we show that our CNN architecture achieves a significant improvement as it reduces the error by about a third across all datasets.

We next turn to the datasets with continuous regression labels. Statistics about the datasets are shown in Table 2, together with our results. The IDIAP Head Pose dataset, which stems from a video recording of few people in a meeting room, has a very restricted range of angles; specifically, 94 % of the pan angles lie within the rather narrow, front-facing range of $[-60°, 60°]$. For this experiment, the output of our network is computed by a fully-connected layer with three outputs and the cost function is the mean absolute deviation. This simple approach to pan-tilt-roll regression outperforms the state-of-the art in all three dimensions. Please note that with a linear output layer and the MAD cost function, the network does not learn the pan, tilt and roll angles jointly; they merely share a common feature representation.[1]

The CAVIAR dataset comes in both a clean version containing only fully-visible heads, and an occluded version containing *only* partially-occluded heads. While they do come in the full range of angles, almost 40 % of the training samples lie within $\pm 4°$ of the four canonical orientations. A major downside of

[1] This becomes evident by computing the derivatives of the cost w.r.t. the parameters: the tilt and roll terms are absent from the derivative w.r.t. the pan and vice-versa.

Table 2: A comparison to two regression datasets from [34]. The first number is the mean absolute angular deviation, the second its standard deviation across test-samples. We obtain state-of-the-art results on all datasets.

	IDIAP Head Pose			CAVIAR-c	CAVIAR-o
# Samples	42 304/23 991			10 660/10 665	10 802/10 889
Pose range	pan	tilt	roll	pan	pan
	[-101,101]	[-73,23]	[-46,65]	[0, 360]	[0, 360]
Tosato *et al.* [34]	10.3°±10.6°	4.5°±5.3°	4.3°±3.8°	22.7°±18.4°	35.3°±24.6°
Ba & Odobez [2]	8.7°±9.1°	19.1°±15.4°	9.7°±7.1°	-	-
Our CNN	**5.9°±7.2°**	**2.8°±2.6°**	**3.5°±3.9°**	**19.2°±24.2°**	**25.2°±26.4°**

this dataset is that most images have been upscaled to 50-by-50 pixels from their original size of, on average, 7-by-7 pixels. We still perform the comparison for the sake of completeness, and our network manages to beat the current state-of-the-art on such a difficult dataset.

These experiments show that the network architecture we use forms a solid basis by itself and we can now use it to further investigate continuous, periodic orientation regression.

4 Periodic Orientation Regression

None of the datasets in the previous section really uncover a crucial problem for full head-orientation regression: periodicity. We can demonstrate that this is a real problem by adding 360° to all negative pan values of the IDIAP dataset. With this semantically identical dataset, the exact same (naive) network used in the previous section becomes very unstable and only reaches errors of 12.9°, 4.5° and 5.3° for pan, tilt and roll, respectively.

For memory-based models such as k-NN and kernel-methods, periodicity only plays a role during the voting part of the algorithm, where it can easily be solved by a modulo operation. But this kind of model suffers from the inherent need of fine-grained training data, hence our focus on parametric models.

For parametric models such as CNNs, periodicity may cause problems in two different ways: (1) The cost function to be optimized is unaware of the fact that a prediction of 359° for a ground truth orientation of 0° should incur the same loss as 1°. Unfortunately, simply applying a `mod` operator to the output of the network results in a discontinuous error function that can no longer be optimized robustly. (2) A regression output which results from a matrix-vector product, such as performed in most parametric models, is an inherently linear operation, while we ideally want a circular output.

Our biternion approach solves both of these problems in an elegant way.

4.1 Von Mises Cost Function

The first problem of discontinuity in the cost function can be addressed by turning to the von Mises distribution [22], which is a close approximation to the normal distribution on the unit circle:

$$p_{\mathrm{VM}}(\varphi \mid \mu, \kappa) = \frac{e^{\kappa \cos(\varphi - \mu)}}{2\pi I_0(\kappa)} \ . \tag{1}$$

Equation (1) defines its probability density function, where φ is an angle, μ is the mean angle of the distribution, κ is inversely related to the variance of the approximated Gaussian, and $I_0(\kappa)$ is the modified Bessel function of order 0, which is a constant for fixed κ. Since it leverages the cosine function to avoid any discontinuity, it is well-suited for gradient-based optimization and we can derive the following cost function by inverting and scaling it accordingly:

$$C_{\mathrm{VM}}(\varphi \mid t; \kappa) = 1 - e^{\kappa(\cos(\varphi - t) - 1)}. \tag{2}$$

In the cost formulation, we call t the target value and κ is a simple hyperparameter that controls the tails of the loss function.

4.2 Biternion Representation for Orientation Regression

While the von Mises cost presented above solves the first issue, the fundamental problem of predicting a periodic value using a linear operation persists. Also, $\|\mathbf{y}\| = 1$Inspired by the quaternion representation often found in computer graphics, we propose a natural alternative representation of an angle by the two-dimensional vector consisting of its sine and cosine $\mathbf{y} = (\cos\varphi, \sin\varphi)$, which we call the *biternion representation*. Surprisingly, the only use of a similar encoding we found in the related literature is that by Osadchy *et al.* [26], who also embed angles into a similar, albeit different, higher-dimensional space. Unfortunately, their approach does not solve the periodicity problem since it uses the discontinuous `atan2` function.

The biternion representation immediately suggests the use of the continuous, cyclic cosine cost widely used in the NLP literature [31]:

$$C_{\cos}(\mathbf{y} \mid \mathbf{t}) = 1 - \frac{\mathbf{y} \cdot \mathbf{t}}{\|\mathbf{y}\| \, \|\mathbf{t}\|} \ . \tag{3}$$

Implementing a biternion output-layer in any framework for neural networks is relatively straightforward, since all that is needed is a fully-connected layer and a normalization layer. For clarity, Equation 4 gives the operation performed by a biternion-layer during the forward pass, where $\mathbf{W} \in \mathbb{R}^{n \times 2}$ and $\mathbf{b} \in \mathbb{R}^2$ are the learnable parameters from the fully-connected layer:

$$f_{\mathrm{BT}}(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \frac{\mathbf{W}\mathbf{x} + \mathbf{b}}{\|\mathbf{W}\mathbf{x} + \mathbf{b}\|} \tag{4}$$

The derivative of the normalization, necessary for the backward pass, can then be stated as

$$\partial_{x_i} \frac{\mathbf{x}}{\|\mathbf{x}\|} = \partial_{x_i} \frac{\mathbf{x}}{\sqrt{\sum_j x_j^2}} = \frac{\sum_{j \neq i} x_j^2}{\left(\sum_j x_j^2\right)^{\frac{3}{2}}} = \frac{\sum_{j \neq i} x_j^2}{\|\mathbf{x}\|^3} \ . \tag{5}$$

Notice how (1) the normalization in the biternion layer makes sure the output values are learned *jointly* and (2) the normalization terms in $C_{\cos}$ can subsequently be omitted.

Finally, the ensembling of multiple biternion predictions, as needed by some augmentation techniques, can simply be performed by averaging the vectors, since the average of unit vectors is again a unit vector, a fact also used by Hara *et al.* [16].

Biternions are Restricted Quaternions. We now show that biternions correspond to unit-quaternions restricted to a single reference axis of rotation. Let Q_φ be the quaternion $\left(a_x \sin(\frac{\varphi}{2}), a_y \sin(\frac{\varphi}{2}), a_z \sin(\frac{\varphi}{2}), \cos(\frac{\varphi}{2})\right)$ representing a rotation of φ around the axis $\mathbf{a}$ and Q_θ the quaternion representing a rotation of θ around the same axis. A quaternion representing the immediate rotation from Q_φ to Q_θ can be computed as $\frac{Q_\varphi}{Q_\theta}$, which corresponds to:

$$\begin{pmatrix} -\cos(\frac{\varphi}{2})a_x \sin(\frac{\theta}{2}) + a_x \sin(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) - a_y \sin(\frac{\varphi}{2})a_z \sin(\frac{\theta}{2}) + a_z \sin(\frac{\varphi}{2})a_y \sin(\frac{\theta}{2}) \\ -\cos(\frac{\varphi}{2})a_y \sin(\frac{\theta}{2}) + a_y \sin(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) - a_z \sin(\frac{\varphi}{2})a_x \sin(\frac{\theta}{2}) + a_x \sin(\frac{\varphi}{2})a_z \sin(\frac{\theta}{2}) \\ -\cos(\frac{\varphi}{2})a_z \sin(\frac{\theta}{2}) + a_z \sin(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) - a_x \sin(\frac{\varphi}{2})a_y \sin(\frac{\theta}{2}) + a_y \sin(\frac{\varphi}{2})a_x \sin(\frac{\theta}{2}) \\ \cos(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) + a_x \sin(\frac{\varphi}{2})a_x \sin(\frac{\theta}{2}) + a_y \sin(\frac{\varphi}{2})a_y \sin(\frac{\theta}{2}) + a_z \sin(\frac{\varphi}{2})a_z \sin(\frac{\theta}{2}) \end{pmatrix}$$

Using the fact that $\|\mathbf{a}\| = 1$, the last entry of the quaternion —which encodes the cosine of half the angle represented by the quaternion— simplifies to $\cos(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) + \sin(\frac{\varphi}{2}) \sin(\frac{\theta}{2}) = \cos(\frac{\varphi-\theta}{2})$. The other entries can similarly be simplified, resulting in a quaternion representing a rotation of the angle from φ to θ around the same axis $\mathbf{a}$. This shows that biternions can be seen as quaternions around a fixed reference axis $\mathbf{a}$ and the cosine cost corresponds to the amplitude of the direct rotation between the predicted and the target biternions.

Relationship to the von Mises Cost. By comparing C_{VM} and $C_{\cos}$, it is visible that they do *not* compute the same expression, *i.e.*, the biternion-layer coupled with the cosine cost does *not* optimize the von Mises cost. The von Mises cost for the biternion layer can be written as:

$$C_{\mathrm{VM,BT}}(\mathbf{y} \mid \mathbf{t}) = 1 - e^{\kappa(\mathbf{y} \cdot \mathbf{t} - 1)}. \tag{6}$$

Notice the similarity to Equation 3; the main difference is the presence of e, which "pushes down" the error around the target value, in effect penalizing small mistakes less strongly.

4.3 Experimental Results

In order to investigate the relative usefulness of the von Mises cost and the biternion representation for periodic regression, we now turn to the TownCentre dataset [5]. This dataset contains heads of tracked pedestrians in a shopping district, annotated with head pose regression labels. The prior distribution of the pose angle is shown in the middle of Fig. 1. For all experiments, we train on 7920 heads of 3960 persons and evaluate on 774 heads of 387 random but *different* persons. The results can be seen in Table 3.

As a first baseline, we train a shallow linear regressor on raw pixel values. We then train a deep CNN using a naive regression output and cost, as described in Section 3.2. While the depth of the architecture allows it to perform much better, it is still plagued by the two problems of cyclic regression. Using the von Mises cost solves the first problem in the cost function; this reduces the error by a significant amount, showing that the more appropriate cost function indeed does aid optimization. Following this, we evaluate the performance of a biternion net both with the cosine cost and the von Mises cost. As can be seen, the expressive power of the biternion layer solves both problems encountered in periodic regression and produces the best results.

Table 3: Quantitative regression results for the TownCentre dataset [5].

Method	MAE
Linear Regression	64.1°±45.0°
Naive Regression	38.9°±40.7°
Von Mises	29.4°±31.3°
Biternion	21.6°±25.2°
Biternion+Von Mises	**20.8°±24.7°**
Benfold&Reid [5]	25.6° / 64.9°

It should be noted that we cannot fairly compare to most of the related work for various reasons: the results in [8] have been computed on only 15 persons, which is far from representative for this dataset. Chamveha *et al.* [7] use a tracker and scene-specific orientation priors. Even the numbers from Benfold and Reid [5] are not a fair comparison since they use walking direction as a prior. The first of their numbers in Table 3 is achieved by a regressor which has seen all persons and their walking direction during training[2], while the second of their numbers has not seen any of the persons since it has been trained on a different dataset.

5 Continuous Regression from Discrete Training Labels

We have shown that biternion nets are well-suited to fully-periodic head pose regression. We now turn to the third contribution of this paper, namely the ability to perform continuous head pose regression using only discrete pose labels for training. To simulate discrete pose labels, we discretize the continuous annotations of the TownCentre dataset. By varying the number of discrete bins, we generate multiple datasets on which we train various approaches using only *the centers of the bins* as training labels. We then evaluate the predictions made by these approaches by computing their mean angular deviation w.r.t. the full regression annotations of the test set. All results are reported in Table 4. We first apply two classification-based baselines, followed by all regression-based approaches introduced in Section 4.

In order to train a regressor using discrete pose labels, a first rather simplistic approach commonly found in the literature is to train a classifier which outputs the class center as prediction. For probabilistic classifiers, a natural extension of this approach is to output the argmax of a quadratic interpolation of the class with the highest posterior probability and its neighboring classes. On average, this improves the results by about 2°.

[2] Their setup is justified for their task, but makes a fair comparison impossible.

Table 4: Regression results from different approaches for different discretizations. Here infinity represents no discretization. Note that the Biternion layer handles the discrete labels very well, both with the cosine and the von Mises cost.

Class bins	Class center	Class interpolation	Naive regression	Von Mises	Biternion	Biternion + Von Mises
3	37.2°±32.8°	35.5°±30.4°	45.5°±39.7°	36.6°±34.5°	**32.1°±28.1°**	32.2°±28.8°
4	34.9°±30.5°	31.7°±29.3°	43.0°±40.6°	33.4°±32.2°	27.1°±27.3°	**26.9°±27.4°**
6	26.1°±28.4°	24.1°±27.6°	38.3°±38.5°	31.8°±33.1°	**22.1°±25.5°**	22.7°±26.7°
8	24.5°±28.6°	22.6°±28.0°	40.6°±39.7°	30.2°±32.3°	21.8°±24.9°	**21.3°±25.2°**
10	23.8°±27.5°	21.9°±26.9°	37.6°±38.3°	28.8°±30.8°	**21.4°±24.6°**	21.8°±25.5°
12	23.6°±29.4°	22.2°±28.8°	39.0°±38.2°	29.7°±31.5°	**21.4°±25.3°**	21.8°±25.3°
∞	-	-	38.9°±40.7°	29.4°±31.1°	21.6°±25.2°	**20.8°±24.7°**

CNNs compute a continuous function of their input and, during training, each sample *pulls* the parameters of the CNN slightly into a direction leading to a better prediction of its pose. This intuition suggests that it should be possible for CNNs to learn a continuous mapping from images to pose angles even when only given very rough pose labels. This is shown in the last four columns of Table 4. As can be seen, this idea hardly works at all in the naive regression case and is only somewhat improved by the von Mises cost. Biternion nets, on the other hand, have no difficulty being trained this way and in fact outperform the class-based approaches with any number of realistically annotable classes, whether the cosine or the von Mises cost is used

Unfortunately, looking only at numbers representing an average error over a large amount of images does not reflect the real advantage of biternion nets over the classifier approach. For this reason, we plotted heatmaps of the predictions made by a CNN classifier with quadratic interpolation and the predictions made by a biternion net in Figure 1. These heatmaps clearly show that, while the class-interpolation approach and biternion nets give similar scores, the predictions of the biternion nets are vastly superior because they are more continuous and similar to the distribution of the ground-truth angles.

5.1 Practicality

To show the potential of our approach, we recorded a small dataset using a common smartphone camera and annotated it with eight class labels. For this, we recorded 24 people in our lab and asked them to rotate on the spot. We then manually cropped a square region in the resulting videos containing their head and rescaled it to 50×50 pixels to make it compatible to our network

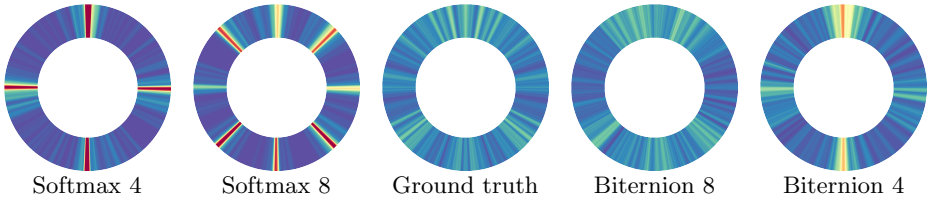


Fig. 1: Prediction distributions for softmax and biternion output layers trained on different discretizations. The classification results include the interpolation.

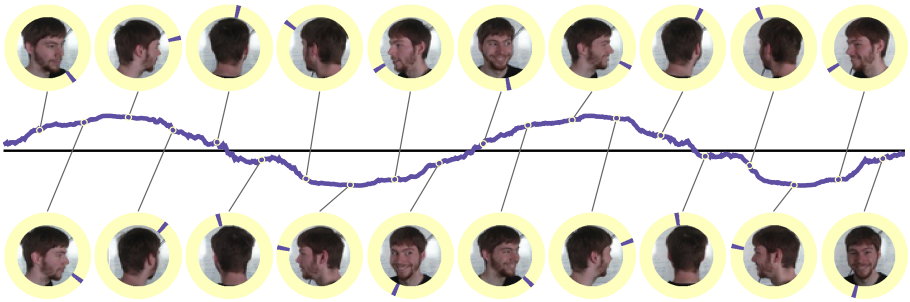


Fig. 2: Qualitative results. The purple line shows the sine of the predicted orientation angle across two full turns. For each head, the purple mark shows the orientation as seen from above. Results are equally spaced and not cherry-picked, more densely sampled results can be seen in the supplementary material.

architecture. In our scenario, the image sequence of a single person can easily be annotated based on temporal constraints. We split up the full annotation task into two annotation runs of four classes. First we annotate `Front`, `Left`, `Back` and `Right`, followed by the same annotation with boundaries shifted by 45°. We select temporal regions in the video through their start and end frames and mark any such region as one class. The resulting pair of annotations can then easily be merged into an eight-class annotation. The whole process, including the cropping of the head regions and the annotation itself, was done by a single person and took no longer than two and a half hours.

We train a biternion net on the resulting dataset except for one person, which we set aside for qualitative evaluation. We only train this network for five epochs since the number of people in this dataset is orders of magnitude smaller than in all previous datasets. We then let the biternion net predict the head pose of the left-out person *for each frame individually*. The result, which can be seen in Figure 2, clearly shows that the network estimates a fairly smooth sinusoidal pose across the two turns the person made, despite having been trained on only eight discrete pose annotations.

6 Conclusion

In this paper, we have introduced biternion nets, a CNN based approach. We have validated our architecture on several public datasets and have shown that our biternion layer is essential for continuous periodic orientation regression. Our obtained results redefine the state of the art on all used datasets. We furthermore show that, using biternion nets, it becomes possible to collect data with discrete and coarse orientation labels, which can be annotated quickly and cheaply, in order to train a continuous and precise head pose regressor. This suggests that fine-grained regression annotations are no longer necessary for continuous orientation estimation. The work in this paper was funded by the EU projects STRANDS (ICT-2011-600623) and SPENCER (ICT-2011-600877). Code is available at `http://github.com/lucasb-eyer/BiternionNet`.

References

1. Aghajanian, J., Prince, S.: Face Pose Estimation in Uncontrolled Environments. In: BMVC (2009)
2. Ba, S.O., Odobez, J.M.: Evaluation of Multiple Cue Head Pose Estimation Algorithms in Natural Environements. In: ICME (2005)
3. Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I.J., Bergeron, A., Bouchard, N., Bengio, Y.: Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop (2012)
4. Baxter, R.H., Leach, M.J., Mukherjee, S.S., Robertson, N.M.: An Adaptive Motion Model for Person Tracking with Instantaneous Head-Pose Features. IEEE Signal Processing Letters 22(5), 578–582 (2015)
5. Benfold, B., Reid, I.: Unsupervised Learning of a Scene-Specific Coarse Gaze Estimator. In: ICCV (2011)
6. Black, Jr., J.A., Gargesha, M., Kahol, K., Kuchi, P., Panchanathan, S.: A Framework for Performance Evaluation of Face Recognition Algorithms . In: Proc. SPIE. vol. 4862, pp. 163–174 (2002)
7. Chamveha, I., Sugano, Y., Sugimura, D., Siriteerakul, T., Okabe, T., Sato, Y., Sugimoto, A.: Head direction estimation from low resolution images with scene adaptation. CVIU 117(10), 1502–1511 (2013)
8. Chen, C., Odobez, J.M.: We are not Contortionists: Coupled Adaptive Learning for Head and Body Orientation Estimation in Surveillance Video. In: CVPR (2012)
9. Dantone, M., Gall, J., Fanelli, G., Van Gool, L.: Real-time Facial Feature Detection using Conditional Regression Forests. In: CVPR (2012)
10. Demirkus, M., Precup, D., Clark, J.J., Arbel, T.: Probabilistic Temporal Head Pose Estimation Using a Hierarchical Graphical Model. In: ECCV (2014)
11. Dollár, P., Welinder, P., Perona, P.: Cascaded Pose Regression. In: CVPR (2010)
12. Fanelli, G., Dantone, M., Gall, J., Fossati, A., Van Gool, L.: Random Forests for Real Time 3D Face Analysis. IJCV 101(3), 437–458 (2013)
13. Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout Networks. In: ICML (2013)
14. Gourier, N., Hall, D., Crowley, J.L.: Estimating Face orientation from Robust Detection of Salient Facial Structures. In: ICPR'04 FG Net Workshop (2004)
15. Gross, R., Matthews, I., Cohn, J., Kanade, T., Baker, S.: Multi-Pie. Image and Vision Computing 28(5), 807–813 (2010)
16. Hara, K., Chellappa, R.: Growing Regression Forests by Classification: Applications to Object Pose Estimation. In: ECCV (2014)
17. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv preprint arXiv:1502.01852 abs/1502.01852 (2015)
18. He, K., Sigal, L., Sclaroff, S.: Parameterizing Object Detectors in the Continuous Pose Space. In: ECCV (2014)
19. Huang, D., Storer, M., De la Torre, F., Bischof, H.: Supervised Local Subspace Learning for Continuous Head Pose Estimation. In: CVPR (2011)
20. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167 (2015)
21. Lallemand, J., Ronge, A., Szczot, M., Ilic, S.: Pedestrian Orientation Estimation. In: GCPR (2014)
22. Mardia, K.V., Jupp, P.E.: Directional Statistics, vol. 494. John Wiley & Sons (2009)

23. Montavon, G., Orr, G.B., Müller, K. (eds.): Neural Networks: Tricks of the Trade - Second Edition. Springer (2012)
24. Murphy-Chutorian, E., Doshi, A., Trivedi, M.M.: Head Pose Estimation for Driver Assistance Systems: A Robust Algorithm and Experimental Evaluation. In: ITSC (2007)
25. Murphy-Chutorian, E., Trivedi, M.M.: Head Pose Estimation in Computer Vision: A Survey. PAMI 31(4), 607–626 (2009)
26. Osadchy, M., Cun, Y.L., Miller, M.L.: Synergistic Face Detection and Pose Estimation with Energy-Based Models. JMLR 8, 1197–1215 (2007)
27. Pérez, F., Granger, B.E.: IPython: a system for interactive scientific computing. Computing in Science and Engineering 9(3), 21–29 (May 2007), http://ipython.org
28. Qi, R.: Learning 3D Object Orientations From Synthetic Images (2015)
29. Saxe, A.M., McClelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In: ICLR (2014)
30. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: ICLR (2015)
31. Singhal, A.: Modern Information Retrieval: A Brief Overview. IEEE Data Eng. Bull. 24(4), 35–43 (2001)
32. Siriteerakul, T.: Advance in Head Pose Estimation from Low Resolution Images: A Review. IJCSI 9(2) (2012)
33. Torki, M., Elgammal, A.: Regression from Local Features for Viewpoint and Pose Estimation. In: ICCV (2011)
34. Tosato, D., Spera, M., Cristani, M., Murino, V.: Characterizing Humans on Riemannian Manifolds. PAMI 35(8), 1972–1984 (2013)
35. Wu, Y., Toyama, K.: Wide-Range, Person- and Illumination-Insensitive Head Orientation Estimation. In: Int. Conf. on Automatic Face and Gesture Recognition (2000)
36. Zeiler, M.D., Rob, F.: Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. In: ICLR (2013)

Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels

Supplementary Material

Lucas Beyer, Alexander Hermans, and Bastian Leibe

Visual Computing Institute, RWTH Aachen University

Abstract. In this supplementary material, we show additional details about both the training procedure and the architecture of our CNN. Furthemore, we show more detailed quantitative and qualitative results.

1 CNN Training

We use Theano [1] as a framework for our implementation. The details of the architecture used throughout the paper can be seen in Table 1. The weights of all convolutional and all but the output's fully-connected layers are initialized using "Xavier"-initialization [2] and all biases are initialized to zero. The weights of both the softmax and the angle output layers are initialized to zero, while those of the biternion output layer are initialized to random standard normal values multiplied by 0.01 to break symmetry. Our implementation of batch-normalization [3] uses a second forward pass through the data after each epoch for collecting exact mini-batch statistics. Its γ weights are initialized to one and its β weights to zero.

For the optimization, we implemented AdaDelta [5] for its stability so we could use the same hyperparameters $\rho = 0.95$ and $\epsilon = 1 \times 10^{-7}$ for all experiments. We only perfromed data augmentation in two ways. First, we horizontally flip all training images and adjust their label accordingly. Second, we use random 46×46 crops during training and average the output of five such crops (center and four corners) during prediction. The size of all our minibatches is 100 and we divide the accumulated parameter gradients by that same number. We ran all experiments five times with random seeds taken from `/dev/urandom`, resetting the optimizer's state between runs, and report all our results as the average of those five runs. Finally, as reported in the main paper, we use early-stopping at epoch 50 for all but the last experiment, for which we stop after the fifth epoch due to the comparatively very small size of the dataset.

Table 1: The CNN architecture used throughout the paper, with two special cases for differently shaped datasets. All Conv layers have a stride of 1 and all MaxPool layers have a stride equal to their size, *i.e.* are non-overlapping.

General		IDIAP		HOC	
Type	Size	Type	Size	Type	Size
Crop size	46×46×3	Input	68×68×3	Input	54×123×3
Conv	24×(3×3×3)	Conv	24×(3×3×3)	Conv	24×(3×3×3)
Batch Norm	24	Batch Norm	24	Batch Norm	24
ReLU		ReLU		ReLU	
Conv	24×(3×3×24)	Conv	24×(3×3×24)	Conv	24×(3×3×24)
Batch Norm	24	Batch Norm	24	Batch Norm	24
ReLU		ReLU		ReLU	
				Conv	24×(3×3×24)
				Batch Norm	24
				ReLU	
MaxPool	2×2	MaxPool	2×2	MaxPool	2×3
Conv	48×(3×3×24)	Conv	48×(3×3×24)	Conv	48×(3×3×24)
Batch Norm	48	Batch Norm	48	Batch Norm	48
ReLU		ReLU		ReLU	
Conv	48×(3×3×48)	Conv	48×(3×3×48)	Conv	48×(3×3×48)
Batch Norm	48	Batch Norm	48	Batch Norm	48
ReLU		ReLU		ReLU	
				Conv	48×(3×3×48)
				Batch Norm	48
				ReLU	
MaxPool	2×2	MaxPool	2×2	MaxPool	3×3
Conv	64×(3×3×48)	Conv	64×(3×3×48)	Conv	64×(3×3×48)
Batch Norm	64	Batch Norm	64	Batch Norm	64
ReLU		ReLU		ReLU	
Conv	64×(3×3×64)	Conv	64×(3×3×64)	Conv	64×(3×3×64)
Batch Norm	64	Batch Norm	64	Batch Norm	64
ReLU		ReLU		ReLU	
		MaxPool	2×2		
Dropout $p = 0.2$		Dropout $p = 0.2$		Dropout $p = 0.2$	
FullyConn	512	FullyConn	512	FullyConn	512
ReLU		ReLU		ReLU	
Dropout $p = 0.5$		Dropout $p = 0.5$		Dropout $p = 0.5$	
Softmax/Angle/Biternion		Softmax/Angle/Biternion		Softmax/Angle/Biternion	

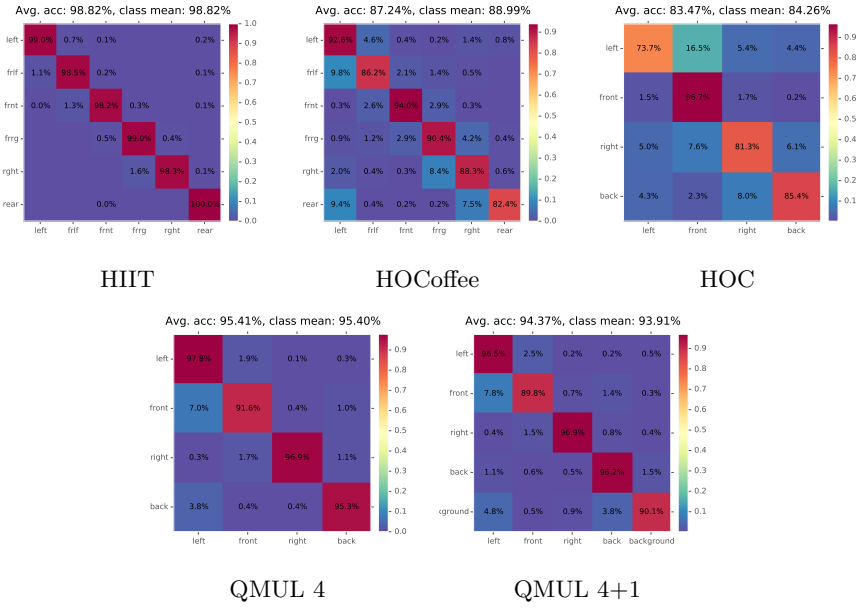


Fig. 1: Confusion matrices of the last run of the classification task.

2 Confusion Matrices of the Classification Tasks

The confusion matrices obtained for the classification task of [4] described in Section 3.2 of the main paper are shown in Fig. 1. The average accuracies may slightly differ from those reported in Table 1 of the main paper because the confusion matrices are those of the last run only, while the numbers in Table 1 of the main paper are the average of five runs.

3 Qualitative Results

Figure 2 shows further qualitative results based on our test person. The blue line in the visualization represents the head pose as seen when looking down onto the person from above it, *i.e.* the line being at the top signifies that the person is looking away from the camera. Due to both anonymity and privacy reasons, further persons will only be published in the final version. None of the pictures are cherry-picked; we show every ninth frame of the full recording. A small mistake of the biternion net is visible in the first three pictures of the second row. Note how the biternion net makes very continuous, stable predictions even though *it works on a frame-by-frame* basis, with no notion of time or order.

References

1. Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I.J., Bergeron, A., Bouchard, N., Bengio, Y.: Theano: new features and speed improvements. Deep

Fig. 2: Qualitative results on our own dataset. The small purple mark indicates the orientation of the person as seen from above, *i.e.* the top corresponds to *back* and the bottom to *front*.

Learning and Unsupervised Feature Learning NIPS 2012 Workshop (2012)
2. Bengio, Y., Glorot, X.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of AISTATS. vol. 9, pp. 249–256 (2010)
3. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167 (2015)
4. Tosato, D., Spera, M., Cristani, M., Murino, V.: Characterizing Humans on Riemannian Manifolds. PAMI 35(8), 1972–1984 (2013)
5. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)

A Semantic Occlusion Model for Human Pose Estimation from a Single Depth Image

Umer Rafi
RWTH Aachen University
Germany
rafi@vision.rwth-aachen.de

Juergen Gall
University of Bonn
Germany
gall@iai.uni-bonn.de

Bastian Leibe
RWTH Aachen University
Germany
leibe@vision.rwth-aachen.de

Abstract

Human pose estimation from depth data has made significant progress in recent years and commercial sensors estimate human poses in real-time. However, state-of-the-art methods fail in many situations when the humans are partially occluded by objects. In this work, we introduce a semantic occlusion model that is incorporated into a regression forest approach for human pose estimation from depth data. The approach exploits the context information of occluding objects like a table to predict the locations of occluded joints. In our experiments on synthetic and real data, we show that our occlusion model increases the joint estimation accuracy and outperforms the commercial Kinect 2 SDK for occluded joints.

1. Introduction

Human pose estimation from depth data has made significant progress in recent years. One success story is the commercially available Kinect system [16], which is based on [20] and provides high-quality body joints predictions in real time. However, the system works under the assumption that the humans can be well segmented. This assumption is valid for gaming application for which the device was developed. Many computer vision applications, however, required human pose estimation in more general environments where objects often occlude some body parts. In this case, the current SDK for Kinect 2 [14] fails to estimate the partially occluded body parts. An example is shown in Figure 1(a) where the joints of the left leg of the person are wrongly estimated.

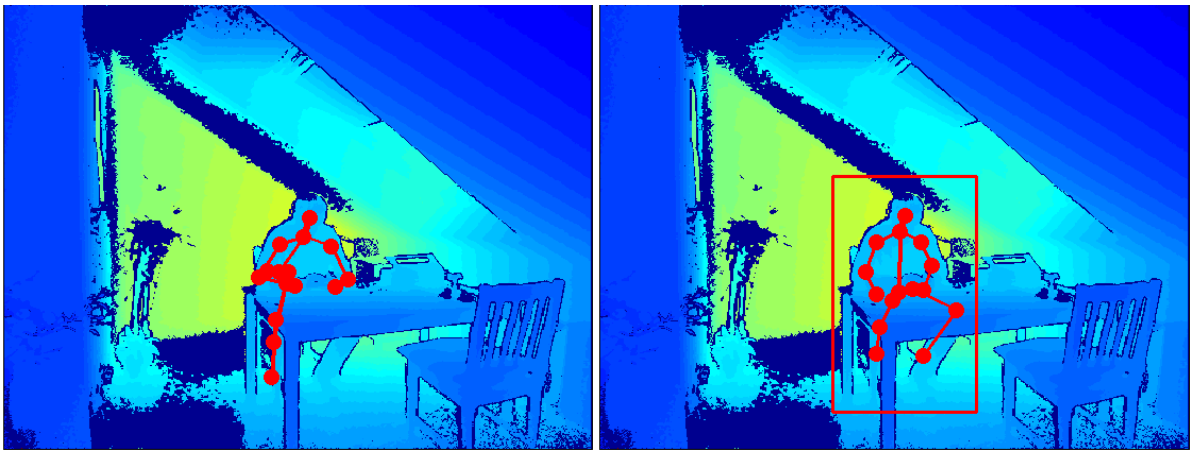
In this work, we address the problem of estimating human pose in the context of occlusions. Since for most applications it is more practical to have always the entire pose and not only the visible joints, we aim to predict the locations of all joints even if they are occluded. To this end, we build on the work from [11], which estimates human pose from depth data using a regression forest. Similar to the SDK, [11] works well for visible body parts but it fails to estimate the joint locations of partially occluded parts since it does not model occlusions. We therefore extend the approach by an occlusion model. Objects, however, not only occlude body parts but they also provide some information about the expected pose. For instance, when a person is sitting at a table as in Figure 1, some joints are occluded but humans can estimate the locations of the occluded joints. The same is true if the hands are occluded by a laptop or monitor. In this case, humans can infer whether the person is using the occluded keyboard and they can predict the locations of the occluded hands.

We therefore introduce a semantic occlusion model that exploits the semantic context of occluding objects to improve human pose estimation from depth data. The model is trained on synthetic data where we use motion capture data and 3D models of furniture. For evaluation, we recorded a dataset of poses with occlusions[1]. The dataset was recorded from 7 subjects in 7 different rooms using the Kinect 2 sensor. In our experiments, we evaluate the impact of synthetic and real training data and show that our approach outperforms [11]. We also compare our approach with the commercial pose estimation system that is part of Kinect 2. Although the commercial system achieves a higher detection rate for visible joints since it uses much more training data and highly engineered post-processing, the detection accuracy for occluded joints of our approach is twice as high as the accuracy of the Kinect 2 SDK.

2. Related Work

3D Human Pose Estimation. Human Pose Estimation is a challenging problem in computer vision. It has applications in gaming, human computer interaction and security scenarios. It has generated a lot of research surveyed in [18]. There are variety of approaches that predict 3D human

[1]The dataset is available at http://www.vision.rwth-aachen.de/data.



(a) (b)

Figure 1: Comparison of Kinect 2 SDK and our approach. The SDK estimates the upper body correctly but the left leg is wrongly estimated due to the occluding table (a). Given the bounding box, our approach exploits the context of the table and predicts the left leg correctly (b).

pose from monocular RGB images [1, 3, 15]. However, a major caveat of using RGB data for inferring 3D pose is that the depth information is not available.

In recent years the availability of fast depth sensors has significantly reduced the depth information problem and further spurred the progress. Researchers have proposed tracking based approaches [13, 17, 8, 9]. These approaches work in real time but operate by tracking 3D pose from frame to frame. They cannot re-initialize quickly and are prone to tracking failures. Recently random forest based approaches [11, 20, 21, 23] have been proposed. They predict the 3D pose in super real time from a single depth image captured by Kinect. These approaches work on monocular depth images and therefore are more robust to tracking failures. Some model based approaches [2, 26] have also been proposed that fit a 3D mesh to image data to predict the 3D pose. However, all these approaches require a well-segmented person to predict the 3D pose and will fail for occluded joints when the person is partially occluded by objects. The closest to our method is approach from [11] that uses regression forest to vote for the 3D pose and can handle self-occlusions. Our approach on the other hand can also handle occlusion from other objects.

Occlusion Handling. Explicit occluder handling has been used in recent years to solve different problems in computer vision. Girshick *et al*. [12] use grammar models with explicit occluder templates to reason about occluded people. The occlusion patterns needs to be specially designed in the grammer. Ghiasi *et al*. [10] automatically learn the different part-level occlusion patterns from data to reason about occlusion in people-people interactions by using flexible mixture of parts [25]. Wang *et al*. [24] use patch based Hough forests to learn object-object occlusions patterns. In facial landmarks localization, recently some regression based approaches have been proposed that also incorporate occlusion handling for localizing occluded facial landmarks [4, 27]. However, these approaches only use the information from non-occluded landmarks in contrast to our approach that also uses the information from occluding objects. Similar to [10, 24] our approach also learns the occlusions from data, however our approach learns occlusions for people-objects interactions in contrast to [10] that learns occlusions for people-people interactions and the occluding objects in our approach are classified purely on appearance at test time and does not incorporate any knowledge about distance from the object they are occluding as in [24].

3. Semantic Occlusion Model

In this work, we propose to integrate additional knowledge about occluding objects into a 3D pose estimation framework from depth data to improve the pose estimation accuracy for occluded joints. To this end, we build on a regression framework for pose estimation that is based on regression forests [11]. We briefly discuss regression forests for pose estimation in Section 3.1. In Section 3.2 we extend the approach for explicitly handling occlusions. In particular, we predict the semantic label of an occluding object at test time and then use it as context to predict the position of an invisible joint.

3.1. Regression Forests for Human Pose Estimation

Random Forests have been used in recent years for various regression tasks, *e.g.*, for regressing the 3D human pose from a single depth image [11, 20, 21, 23], estimating the 2D human pose from a single image [7], or for predicting facial features [6]. In this section, we briefly describe the approach [11], which will be our baseline.

Regression forests belong to the family of random forests and are ensembles of T regression trees. In the context of human pose estimation from a depth image, they take an input pixel q in a depth image D and predict the probability distribution over the locations of all joints in the image. Each tree t in the forest consists of split and leaf nodes. At each split node a binary split function $\phi_\gamma(q, D) \mapsto \{0, 1\}$ is stored, which is parametrized by γ and evaluates a pixel location q in a depth image D. In this work, we use the depth comparison features from [20]:

$$\phi_\gamma(q, D) = \begin{cases} 1 & \text{if } D\left(q + \frac{u}{D(q)}\right) - D\left(q + \frac{v}{D(q)}\right) > \tau \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

where the parameters $\gamma = (u, v, \tau)$ denote the offsets u, v from pixel q, which are scaled by the depth at pixel q to make the features robust to depth changes. The threshold converts the depth difference into a binary value. Depending on the value of $\phi_\gamma(q, D)$, (q, D) is send to the left or right child of the node.

During training each such split function is selected from a randomly generated pool of splitting functions. This set is evaluated on the set of training samples $Q = \{(q, D, c, \{V_j\})\}$, each consisting of a sampled pixel q in a sampled training image D, a class label c for the limb the pixel belongs to, and for each joint j the 3D offset vectors $V_j = q_j - q$ between pixel q and the joint position q_j in the image D.

Each sampled splitting function ϕ, partitions the training data at the current node into the two subsets $Q_0(\phi)$ and $Q_1(\phi)$. The quality of a splitting function is then measured by the information gain:

$$\phi^* = \arg\max_\phi g(\phi), \quad (2)$$

$$g(\phi) = H(Q) - \sum_{s\in\{0,1\}} \frac{|Q_s(\phi)|}{|Q|} H(Q_s(\phi)), \quad (3)$$

$$H(Q) = -\sum_c p(c|Q) \log(p(c|Q)), \quad (4)$$

where $H(Q)$ is the Shannon entropy and $p(c|Q)$ the empirical distribution of the class probabilities computed from the set Q. The training procedure continues recursively until the maximum allowed depth for a tree is reached.

At each leaf node l, the probabilities over 3D offset vectors V_j to each joint j, *i.e.*, $p_j(V|l)$ are computed from the training samples Q arriving at l. To this end, the vectors V_j are clustered by mean-shift with a Gaussian Kernel with bandwidth b and for each joint only the two largest clusters are kept for efficiency. If V_{ljk} is the mode of the k^{th} cluster for joint j at leaf node l, then the probability $p_j(V|l)$ is approximated by

$$p_j(V|l) \propto \sum_{k\in K} w_{ljk} \cdot \exp\left(-\left\|\frac{V - V_{ljk}}{b}\right\|_2^2\right) \quad (5)$$

where the weight of a cluster w_{ljk} is determined by the number of offset vectors that ended in the cluster. Cluster centers with $\|V_{ljk}\| > \lambda_j$ are removed since these correspond often to noise [11].

For pose estimation, pixels q from a depth image D are sampled and pushed through each tree in the forest until they reach a leaf node l. For each pixel q, votes for the absolute location of a joint j are computed by $x_j = q + V_{ljk}$. In addition a confidence value that takes the depth of the pixel q into account is computed by $w_j = w_{ljk} \cdot D^2(q)$. The weighted votes for a joint j are collected for all pixels q and form the set $\mathcal{X}_j = \{(x_j, w_j)\}$. The probability of the location of a joint j in image D is then approximated by

$$p_j(x|D) \propto \sum_{(x_j,w_j)\in\mathcal{X}_j} w_j \cdot \exp\left(-\left\|\frac{x - x_j}{b_j}\right\|_2^2\right) \quad (6)$$

where b_j is the bandwidth of the Gaussian Kernel learned separately for each joint j. As for training, the votes are clustered and only the clusters with the highest summed weights w_j are used to predict the joint location.

3.2. Occlusion Aware Regression Forests (OARF)

In order to handle occlusions, we propose Occlusion Aware Regression Forests (OARF) that build on the regression forest framework described in Section 3.1. They predict additionally the class label of an occluding object at test time and then use this semantic knowledge about the occluding object as context to improve the pose estimation of occluded joints. To this end, we use an extended set of training samples $Q_{ext} = Q \cup Q_{occ}$, where $Q_{occ} = \{(q_{occ}, D, c_{occ}, \{V_{jocc}\})\}$, is a set of occluding object pixels where each pixel q_{occ} is sampled from a training image D, has a class label c_{occ} and a set of of offset vectors $\{V_{jocc}\}$ to each joint of interest j.

During training we use the depth comparison features described in Section 3.1 for selecting a binary split function $\phi_\gamma(q, D)$ at each split node in each tree. To select binary split functions that can distinguish between occluding objects and body parts we minimize the Shannon entropy $H(Q)$ over extended set of class labels $c_{ext} = c \cup c_{occ}$:

$$H(Q) = -\sum_{c_{ext}} p(c_{ext}|Q) \log(p(c_{ext}|Q)), \quad (7)$$

To use the semantic knowledge about occluding object as an additional clue for prediction of occluded joints, we also store at a leaf node l the probabilities over 3D offset vectors V_{jocc} to each joint j, *i.e.*, $p_j(V_{occ}|l)$ that are computed from the training samples Q_{occ} arriving at l by using the mean shift procedure described in Section 3.1. The probability $p_j(V_{occ}|l)$ is approximated by

$$p_j(V_{occ}|l) \propto \sum_{k \in K} w_{ljk} \cdot \exp\left(-\left\|\frac{V_{occ} - V_{ljocck}}{b}\right\|_2^2\right) \quad (8)$$

The inference procedure is similar to standard regression forest inference. At test time pixels q_{occ} from occluding objects in a test image D are also pushed through each tree in the forest until they reach a leaf node l and cast a vote $x_{jocc} = q_{occ} + V_{ljocck}$ with confidence w_{jocc} for each joint j. The weighted votes for each joint j form the set $\mathcal{X}_j = \{(x_j, w_j) \cup (x_{jocc}, w_{jocc})\}$. The final joint position is then predicted by using the mean shift procedure described in Section 3.1.

4. Training Data

Gathering real labeled training data for 3D human pose estimation from depth images is expensive. To overcome this, [20] generated a large synthetic database of depth images of people covering a wide variety of poses together with pixel annotations of body parts. For generating such a database, a large motion capture corpus of general human activities has been recorded. The body poses of the corpus were then retargeted to textured body meshes of varying sizes. Since the dataset is not publicly available, we captured our own dataset.

Synthetic Data. We follow the same procedure. To this end, we use the motion capture data for sitting and standing poses from the publicly available CMU motion capture database [5]. We retarget the poses to 6 textured body meshes using Poser [19], a commercially available animation package, and generate a synthetic dataset of 1'110 depth images of humans in different sitting and standing poses from different viewpoints. For each depth image, we also have a pixel-wise body part labeling and the 3D joint positions. The depth maps and body part labels are shown in Figure 2(a-b). For the occluding objects, we use the publicly available 3D models of tables and laptops from the Sweet Home 3D Furniture Library [22]. We render the object together with the humans as shown in Figure 2(c). The compositions are randomized under the constraints that the tables and feet are on the ground plane, the laptops on the tables and the distance between the humans and the objects is between 3-5 cm. For each composition, we compute the occluded body parts (Figure 2(d)) and the depth and class labels with occluding object classes Figure 2(e-f).

Real Data. We also recorded a dataset using the Kinect 2 sensor. The dataset contains depth images of humans in different sitting and standing poses without occlusions as shown in Figure 2(a). The 3D poses are obtained by the Kinect SDK and we discarded images were the SDK failed. This resulted in 2'552 images. The pixel-wise segmentation of the body parts is obtained by the closest geodesic distance of a surface point to the skeleton as shown in Figure 2(b). The composition with synthetic 3D objects is done as for the synthetic data.

5. Experiments and Results

In this section we describe in detail the experimental settings used to evaluate our method and report quantitative and qualitative results. For comparison, we consider three approaches:

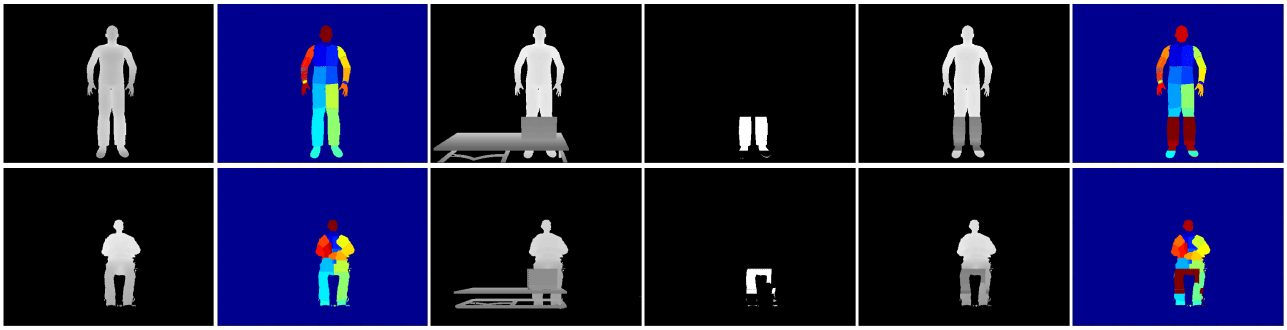
1. The approach [11] described in Section 3.1 is our baseline. It is trained on the training data without occluding objects shown in Figure 2(a-b).

2. The occlusion aware regression forest (OARF W semantics) described in Section 3.2 is trained with the semantic labels of occluding objects shown in Figure 2(e-f).

3. To show the impact of the semantic information of the occluding objects, we also train an OARF by assigning all occluding objects a single label and not the labels of the object classes (OARF W/O semantics).

Training. We train all forests with the same parameters. For each training depth image, we sample 1000 pixel locations. The other parameters are used as in [11], *i.e.*, the regression forests consist of 3 trees with maximum depth 20. For each splitting node, we sample 2000 depth comparison features and use $b = 0.05m$.

Testing. For testing our approach, we use synthetic and real data. The real dataset is recorded in different indoor environments, offices and living rooms, and consists of seven sequences of different subjects in different sitting and standing poses. From each sequence, we select a set of unique poses thus providing us with a total of 1000 images. We split the 1000 images into test and validation set with 800 and 200 images, respectively. While b_j were set as proposed in [11], we observed that the values for λ_j proposed in [11] are not optimal for our baseline. We therefore estimate them on the validation set. The synthetic test set consists of 440 images of 2 subjects. The subjects, occluding objects and the poses in both test sets are different from the ones in the training set. We report quantitative results on real and synthetic test sets for the 15 body joint positions of our skeleton.



(a) (b) (c) (d) (e) (f)

Figure 2: Procedure for generating depth images with pixel level ground truth body parts labels and occluding objects masks with class labels. Synthetic data (top row) and real data (bottom row): (a) Rendered or captured and segmented depth image. (b) Body part labels at pixel level. (c) Depth data with object. (d) Occluded body parts. (e) Segmented depth map with occluding objects. (f) Combined labels of body parts and occluding objects.

Method	Mean Average Precision
Baseline	44%
OARF W/O semantics	48%
OARF W semantics	54%

Table 1: Mean average precision of the 3D body joints predictions on the synthetic test set by using the evaluation measure from [20] with a distance threshold of 10 cm. The results show that integrating semantic knowledge about occluding objects provides a significant improvement over the baseline.

Synthetic Test Data. For quantitative evaluation on synthetic data, we use the evaluation measure from [20] with a distance threshold of 10 cm and report the mean average precision over the 3D body joints predictions in Table 1. Integrating the additional knowledge about occluding objects without object class labels alone provides 4% improvement over the baseline. When we also integrate semantic knowledge about occluding objects then this provides a significant improvement of 10% over the baseline. This shows that occlusion handling is beneficial for pose estimation, but also that the semantic context of occluding objects contains important information about joint locations.

Real Test Data. For real test data, it is difficult to get 3D ground truth body joint positions. For the quantitative evaluation, we therefore manually labeled the 2D body joint positions. Since manual annotations of the 2D positions of body joints, especially occluded joints, in depth images are sometimes noisy, we used the mean annotations of three different annotators as ground truth. For the quantitative evaluation on the test data, we use the evaluation measure from [7] with an error threshold of 0.1 of upper body size to report average detection accuracy over 2D body joint predictions.

Setting	Average Detection Accuracy(%)		
	Occluded Joints	Non Occluded Joints	All Joints
Synthetic Data			
Baseline	22.17	50.51	44.54
OARF W/O semantics	24.36	52.57	46.62
OARF W semantics	25.42	52.43	46.73
Real Data			
Baseline	25.42	46.21	41.82
OARF W/O semantics	28.26	49.72	45.19
OARF W semantics	31.12	51.69	47.94
Real + Synthetic Data			
Baseline	28.62	55.02	49.45
OARF W/O semantics	32.60	55.50	50.66
OARF W semantics	35.77	56.01	51.72
Kinect SDK	18.13	66.36	56.94

Table 2: Average detection accuracy of 2D body joint predictions on the real test set measured by the evaluation measure from [7] with an error threshold of 0.1 of upper body size.

In Table 2, the results for the baseline, OARF with and without semantics are reported. We also evaluate the impact of the synthetic and real training data. The results show that the synthetic training data is not as good as the real training data. However, if we combine real and synthetic data we get another boost of performance. For all three training settings, the numbers support the results of the synthetic test

set. The baseline is improved by occlusion handling and semantic occlusion handling achieves the best result of the three methods. The semantic occlusion model mainly improves the accuracy of occluded joints, but there is also a slight improvement of non-occluded joints. Without occlusion handling, objects close to joints introduce some noisy votes that can result in wrong estimates. The occlusion handling reduces this effect. We also compared our results to the Kinect SDK. The Kinect SDK achieves a higher overall accuracy since it is trained on much more training data and the SDK includes additional post-processing, which is not part of our baseline. However, our approach achieves a much better accuracy for the occluded joints.

Table 3 presents detailed quantitative results for 11 body joints that are occluded in the real test set. The results are reported for OARF trained on real and synthetic data and for the Kinect SDK. The results show that for most joints our model achieves a better accuracy than the Kinect SDK and adding semantic knowledge provides further improvements. There are only three joints, namely the right elbow and the ankles, with a low accuracy. This can be explained by the training data that contains only few examples where these joints are occluded.

Qualitative Results. We show some qualitative results on our real test set in Figure 3 and a few failure cases in Figure 4. The top row shows the body joints predicted by OARF with the semantic occlusion model, the middle row shows the body joints predicted by OARF without the semantic occlusion model and the bottom row shows the body joints predicted by the Kinect SDK.

6. Conclusion

In this paper, we have proposed an approach that integrates additional knowledge about occluding objects into an existing 3D pose estimation framework from depth data. We have shown that occluding objects not only need to be detected to avoid noisy estimates, but also that the semantic information of occluding objects is a valuable source for predicting occluded joints. Although our experiments already indicate the potential of the approach and outperform a commercial SDK already for occluded joints, the overall performance can still be boosted by increasing the variety of objects and poses in the training data.

Acknowledgment: The work in this paper was funded by the EU projects STRANDS (ICT-2011-600623) and SPENCER (ICT-2011-600877). Juergen Gall was supported by the DFG Emmy Noether program (GA 1927/1-1).

Joint	Per Joint Detection Accuracy(%)		
	OARF W semantics	OARF W/O semantics	Kinect SDK
Spine	77.51	73.9	42.57
Left Elbow	76.47	73.53	47.06
Right Elbow	17.5	10.53	71.93
Left Hand	47.59	36.55	28.28
Right Hand	59.38	39.58	18.23
Left Hip	50.53	48.76	10.60
Right Hip	75.42	71.25	22.08
Left Knee	31.64	31.64	22.60
Right Knee	27.27	29.09	12.73
Left Ankle	3.87	4.9	5.15
Right Ankle	5.44	5.07	7.88
Average	35.77	32.60	18.13

Table 3: Per joint detection accuracy of 11 occluded body joints in our real test set by using the evaluation measure from [7] with an error threshold of 0.1 of upper body size. The results are reported for OARF with and without semantics and for the Kinect SDK.

References

[1] A. Agarwal and B. Triggs. Recovering 3d Human Pose from Monocular Image. *PAMI*, 28(1):44–58, 2006. 2

[2] A. Baak, M. Muller, G. Bharaj, H. P. Seidel, and C. Theobal. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *ICCV*, 2011. 2

[3] L. Bo, C. Sminchisescu, A. Kanaujia, and D. Metaxas. Fast Algorithms for Large Scale Conditional 3d Prediction. In *CVPR*, 2008. 2

[4] X. P. Burgos-Artizzu, P. Perona, and P. Dollar. Robust face landmark estimation under occlusion. In *ICCV*, 2013. 2

[5] CMU Mocap. http://mocap.cs.cmu.edu/. 4

[6] M. Dantone, J. Gall, G. Fanelli, and L. van Gool. Real-time Facial Feature Detection using Conditional Regression Forests. In *CVPR*, 2012. 3

[7] M. Dantone, J. Gall, C. Leistner, and L. van Gool. Human Pose Estimation using Body Parts Dependent Joint Regressors. In *CVPR*, 2013. 3, 5, 6

[8] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. In *CVPR*, 2010. 2

[9] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real Time Human Pose Tracking from Range Data. In *ECCV*, 2012. 2

[10] G. Ghiasi, Y. Yang, D. Ramanan, and C. C. Fowlkes. Parsing Occluded People. In *CVPR*, 2014. 2

[11] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient Regression of General-Activity Human Poses from Depth Images. In *ICCV*, 2011. 1, 2, 3, 4

[12] R. B. Girshick, P. F. Felzenszwalb, and D. A. Mcallester. Object Detection with Grammar Models. In *NIPS*, 2011. 2

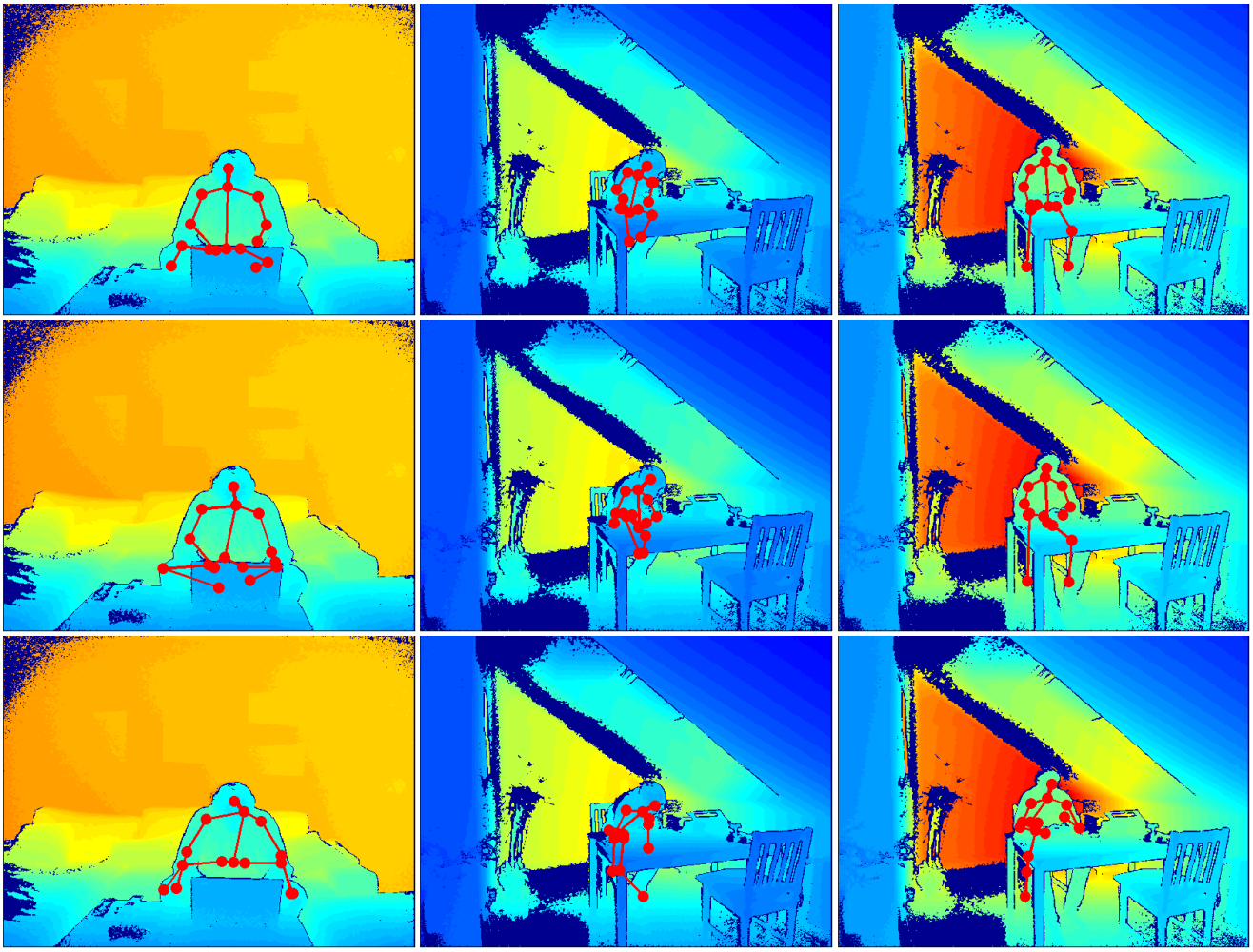


Figure 3: Qualitative results for some sample images taken from our real test set. Top row shows the body joints predicted by OARF with semantics. Middle row shows the body joint predictions from OARF without semantics. Bottom row shows the body joints predicted by the Kinect SDK. The results show that integrating knowledge about occluding objects helps in improving the body joint prediction performance for the occluded joints.

[13] D. Grest, J. Woetzel, and R. Koch. Nonlinear Body Pose Estimation from Depth Images. In *DAGM*, 2005. 2

[14] Kinect For Windows SDK 2.0. http://www.microsoft.com/en-us/kinectforwindows/develop/. 1

[15] I. Kostrikov and J. Gall. Depth Sweep Regression Forests for Estimating 3D Human Pose from Images. In *BMVC*, 2014. 2

[16] Microsoft Corp.Redmond WA. Kinect for Xbox 360. 1

[17] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. Real-Time Identification and Localization of Body Parts from Depth Images. In *ICRA*, 2010. 2

[18] R. Poppe. Vision-based Human Motion Analysis. *CVIU*, 108(1-2):4–18, 2007. 1

[19] Poser. http://my.smithmicro.com/. 4

[20] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time Human Pose Recognition in Parts from Single Depth Images. In *CVPR*, 2011. 1, 2, 3, 4, 5

[21] M. Sun, P. Kohli, and J. Shotton. Conditional Regression Forests for Human Pose Estimation. In *CVPR*, 2012. 2, 3

[22] Sweet Home 3D. www.sweethome3d.com. 4

[23] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The Vitruvian Manifold: Inferring Dense Correspondences for One-Shot Human Pose Estimation. In *CVPR*, 2012. 2, 3

[24] T. Wang, X. He, and N. Barnes. Learning Structured Hough Voting for Joint Object Detection and Occlusion Reasoning. In *CVPR*, 2013. 2

[25] Y. Yang and D. Ramanan. Articulated Pose Estimation using Flexible Mixtures of Parts. In *CVPR*, 2011. 2

[26] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefey. Accurate 3D pose estimation from a single depth image. In *ICCV*, 2011. 2

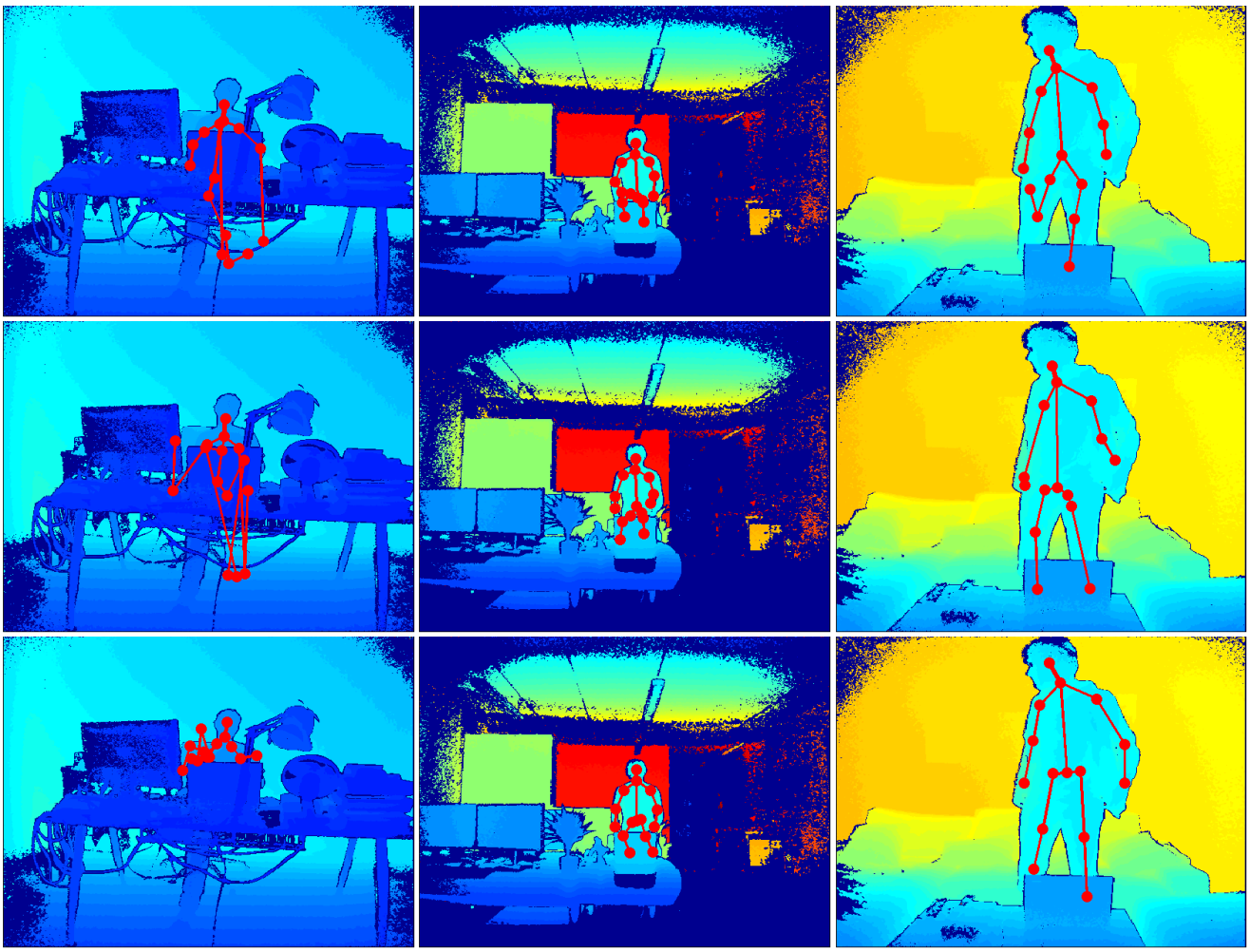


Figure 4: Example failure cases for occluded joints from our Real Test Set. OARF with semantics (top row), OARF without semantics (middle row) and Kinect SDK (bottom row).

[27] X. Yu, Z. Lin, J. Brandt, and D. N. Metaxas. Consensus of Regression for Occlusion-Robust Facial Feature Localization. In *ECCV*, 2014. 2