Hough Networks for Head Pose Estimation and Facial Feature Localization

Gernot Riegler
riegler@icg.tugraz.at
David Ferstl
ferstl@icg.tugraz.at
Matthias Rüther
ruether@icg.tugraz.at
Horst Bischof
bischof@icg.tugraz.at

Institute for Computer Graphics and Vision
Graz University of Technology
Austria

Abstract

We present *Hough Networks (HNs)*, a novel method that combines the idea of Hough Forests (HFs) [12] with Convolutional Neural Networks (CNNs) [18]. Similar to HFs we perform a simultaneous classification and regression on densely extracted image patches. But instead of a Random Forest (RF) we utilize a CNN which is able to learn higher-order feature representations and does not rely on any handcrafted features. Applying a CNN on a patch level has the advantage of reasoning about more image details and additionally allows to segment the image into foreground and background. Furthermore, the structure of a CNN supports efficient inference of patches extracted from a regular grid. We evaluate HNs on two computer vision tasks: head pose estimation and facial feature localization. Our method achieves at least state-of-the-art performance without sacrificing versatility which allows extension to many other applications.

1 Introduction

Head pose estimation and facial feature localization are keys to advanced human computer interaction systems and human behavior analysis. Due to their relevance, both tasks have gained a lot of attention in the computer vision community [1, 2, 6, 10, 11, 21, 23, 30, 32].

Recent state-of-the-art methods like [2, 6, 11] report impressive results and are real-time capable. However, most of the available approaches utilize handcrafted features. In contrast, our method is motivated by the success of Deep Neural Networks (DNNs) in recent years on a variety of tasks, such as image classification and localization [17, 22, 25], human pose estimation [27], and face recognition [26]. Impressive results are enabled by the high capacity of DNNs, the ability to learn higher-order feature representations and the capability to train them in reasonable time through the computational power of modern Graphics Processing Units (GPUs).

On the other hand, the principle of Hough Forests (HFs) [12, 13] proved to be very successful in many computer vision applications, such as head pose estimation [10, 11],

© 2014. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

human pose estimation[14], or facial feature localization [5, 6]. All these methods train a Random Forest (RF) for joint classification and regression. For inference, patches are densely sampled from the image and the model classifies for each patch if it belongs to the foreground or the background. Additionally, for foreground patches the model performs a regression of vectors to vote in a Hough space.

We propose a novel method that combines the idea of HFs with the representative power of Convolutional Neural Networks (CNNs), *Hough Networks (HNs)*. Identical to HFs, we perform simultaneously classification and regression on image patches, where the patches are large enough to cover some context, but small enough to gather image details. However, in contrast to HFs, HNs do not rely on handcrafted features and are instead able to learn them directly from annotated training images. Further, the structure of a CNN also allows for fast inference, as we can apply the model on the whole image instead on every overlapping patch independently.

2 Related Work

Head pose estimation, facial landmark localization and neural networks are very active fields. Since a complete literature review is beyond the scope of this paper, we analyze the approaches that are most related to our method.

Head Pose Estimation Several works in the literature address the problem of estimating the pose of the human head [19] and existing methods can be divided by the type of input data (2D images, depth data, or both). In general, approaches that use 2D images are sensitive to illumination and the lack of distinctive features. Due to this and because cheap depth sensors have already entered the consumer market, we focus on works that use depth data as input.

Breitenstein et al. [1] presented a method based on first computing hypotheses of nose locations from high-resolution depth images and than minimizing a error function between these hypotheses and reference pose images. The method achieves real-time performance, if implemented on a GPU. The approach of Fanelli et al. [9] utilizes a Random Regression Forest (RRF) for this estimation task. It also uses high resolution depth data as input, but in contrast to [1] the method of [9] has not the drawback that the nose has to be visible in any frame. The approach was later adopted to depth data from consumer depth cameras [10, 11]. In addition, they trained the RF to discriminate between head and background patches and jointly regress the head center location and the head pose. Schulter et al. [21] recently improved the method by using an Alternating Regression Forest (ARF). In contrast to the RRF, an ARF minimizes a global loss to obtain better generalization.

Facial Feature Localization The localization of facial feature from 2D images is a well-studied problem. Early, holistic approaches include Active Contour Models [16], Active Shape Models [3] and Active Appearance Models [4]. In general, these methods suffer from poor generalization properties.

Most of recent work tackles the problem in a regression framework by learning one or several real-valued functions, which predict the location of different facial features. Valstar et al. [28] used Support Vector Regression and Markov Random Fields to constrain the search space. Yang and Patras [31] utilized a Structured-Output Regression Forest that learns at each leaf node a regression function and simultaneously an interdependency model between parts in a star-graph. In the work of Dollár et al. [7], they developed a cascade of regressors,

e.g. random ferns. The first stage learns a rough estimate of facial feature locations, whereas subsequent stages learn the difference between these estimates and the true locations to refine the results. Burgos-Artizzu et al. [2] improved the previous method to better handle occlusion and shape variations by additional learning point occlusions and introducing shape-indexed features. The work proposed by Dantone et al. [6] uses a Conditional Regression Forest. A first trained RRF predicts the head pose and depending on the result another RRF computes the facial feature locations from densely extracted image patches.

Neural Networks Image Processing with neural networks goes back at least to the 1980s and the introduction of the back-propagation algorithm [8]. LeCun and Bengio [18] developed CNNs that can learn local features and higher-order feature representations via consecutive convolutions with an arbitrary number of filter kernels and sub-sampling.

In the recent years, deep network architectures have become popular due to the huge amount of training data available and the ability to train the networks more efficient on GPUs. Krizhevsky et al. [17] obtained impressive results on the Large Scale Visual Recognition Challenge 2012. Szegedy et al. [25] presented a method for object detection based on the deep architecture and used it for object localization by regressing bounding boxes.

CNNs have also been already applied to pose estimation problems and facial feature localization. Toshev and Szegedy [27] trained a cascade of DNNs to estimate human joint locations in color images. Each stage of the network gets a higher resolution crop of the image as input to refine the results. The approach of Sun et al. [23] for facial feature localization is similar, except that they learn in each stage an ensemble of networks.

In contrast to the methods of [23, 27], our approach does not rely that the face was previously detected by another method. The proposed method is similar to the ideas of [6, 10, 11] as we perform a joint classification and regression on image patches, but instead of a RF we utilize a CNN.

3 Hough Networks

Our method is inspired by the HFs [12, 13], also named Discriminative Random Regression Forest [10]. First, overlapping patches are densely extracted from the image along a regular grid. Then, a binary classifier computes the patch probability for belonging to foreground or background (*e.g*. to a face, or not). Simultaneously, for each foreground patch several real-valued numbers are predicted, *e.g*. head center and pose, or facial feature locations.

Notation In this work we will use the following notation: We have a supervised learning problem and assume a training dataset $\{(x_s, \mathbf{t}_s)\}_{s=1}^{S}$ with S samples. An image patch $x_s \in \mathbb{R}^{M\times N\times K}$ is of size $M \times N$ and has K channels. A target vector $\mathbf{t}_s = (t_{s,c}, \mathbf{t}_{s,r})^T$ contains the classification information that discriminates between foreground and background $t_{s,c} \in \{0,1\}$ along with regression values $\mathbf{t}_{s,r} \in \mathbb{R}^R$. We denote $\mathbf{y}(x,\theta) \in \mathbb{R}^{1+R}$ as the classification and regression function with model parameters θ. With a slight abuse of notation we define $\mathbf{y}_s = (y_{s,c}, \mathbf{y}_{s,r})^T = \mathbf{y}(x_s, \theta)$.

Architecture In our approach, the classification and regression function is based on a CNN. The CNN consists of 6 layers, where each layer performs a non-linear transformation of the data from the previous layer. The architecture of our CNN is visualized in Figure

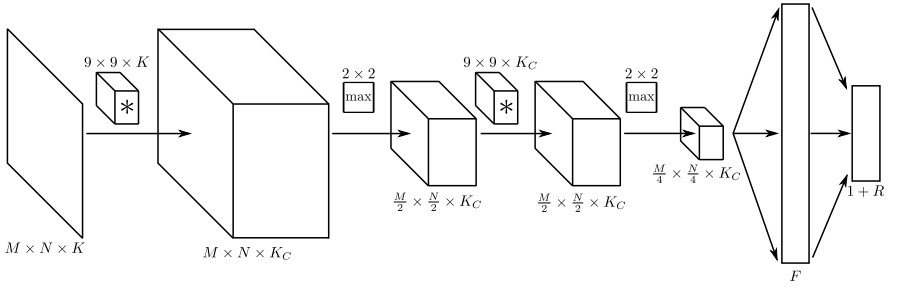


Figure 1: An illustration of the CNN we use for classification and regression. The first layer takes the image of dimension $M \times N \times K$ as input and computes a convolution with K_C kernels of size 9×9, followed by a local response normalization. The next layer performs max-pooling in a 2×2 non-overlapping neighborhood. Layer three and four perform a convolution with K_C kernels of size 9×9 and a 2×2 max-pooling, respectively. As last layers of our architecture we implement a fully-connected layer with F neurons, followed by the output-layer with $1 + R$ neurons.

1 and is motivated by the work of [17]. The input of the network is an image patch of size $M \times N \times K$, whereas we have normalized the image by subtracting the mean and divided by the standard deviation of all training images. As first layer we use a convolutional layer with K_C kernels of size 9×9 and a positive bias term. This results in K_C feature maps on which we apply local response normalization as suggested in [17]. This aids generalization and can be seen as a kind of brightness normalization. In the second layer we utilize a non-overlapping max-pooling with neighborhood size of 2×2. The third layer performs again a convolution with the same number and size of kernels as the previous one. After another max-pooling layer, we have a fully-connected layer with F neurons and an output layer with $1 + R$ neurons.

Both, the convolutional layers and the fully-connected layers are linear transformations followed by a non-linear one. We use a Rectified Linear Unit defined as $\max(0, a)$ as non-linearity in our network. The output layer consists of one sigmoid activation function for classification y_c and of R linear activation functions for the regression $\mathbf{y}_r$.

3.1 Training

The biggest difference to existing CNN approaches is the simultaneous classification and regression. For that reason we minimize a distinctive objective function. For classification, we utilize the cross-entropy error. For a sample s it reads:

$$E_{s,c}(\theta) = -\left(t_{s,c} \ln\left(y_{s,c}\right) + \left(1 - t_{s,c}\right) \ln\left(1 - y_{s,c}\right)\right). \tag{1}$$

In contrast, for the regression components we use the L_2 loss that minimizes the distance between target and predicted regression values. Again, for a sample s we have:

$$E_{s,r}(\theta) = \frac{1}{2} \|\mathbf{y}_{s,r} - \mathbf{t}_{s,r}\|^2. \tag{2}$$

These two error functions are combined in a weighted sum as follows

$$E_s(\theta) = \lambda_c E_{s,c} + \lambda_r E_{s,r}, \tag{3}$$

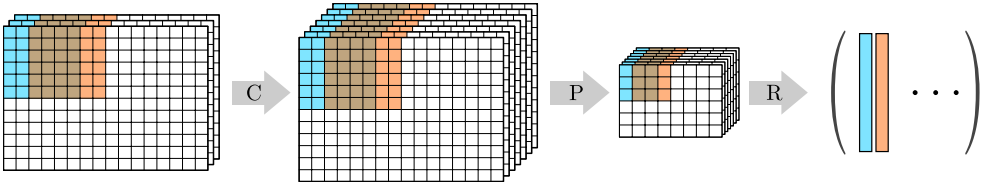


Figure 2: The structure of a CNN allows for fast inference in HNs. Instead of densely extracting overlapping patches from the input image and evaluate the network for each patch separately, we can obtain the same results with a single evaluation of the CNN. If we use the whole image as input, the convolutional (C) and pooling (P) operations are overlapping for neighboring patches (blue and orange depicts two 6×6 patches and brown their overlap). Only before the first fully-connected layer we have to reshape (R) the values of the feature maps into a matrix, where each column-vector represents one patch.

where λ_c and λ_r are weighting coefficients of the individual error functions and relate to increased or decreased delta values in the back-propagation algorithm. They can be used to steer between classification, or regression during learning.

Finally, we learn the model parameters θ by optimizing

$$\arg\min_{\theta} \sum_{i=1}^{I} E_s(\theta) \tag{4}$$

using the back-propagation algorithm and Nesterov's Accelerated Gradient [20] in a stochastic setting:

$$\hat{\theta}^{\tau+1} = \mu\hat{\theta}^{\tau} - \eta\nabla E(\theta^{\tau} + \mu\hat{\theta}^{\tau}) \tag{5}$$

$$\theta^{\tau+1} = \theta^{\tau} + \hat{\theta}^{\tau+1}. \tag{6}$$

where η is the learning rate and μ is the momentum coefficient. Sutskever et al. [24] showed that this scheme accelerates learning compared to normal stochastic gradient descent.

The objective function in Equation 3 allows that the values in the single target vectors can be missing (*e.g.* a facial feature is occluded). In such cases we set the gradient values of the involved weights (which only effects connection to the output layer) to zero. We especially utilize this fact, if a patch does not belong to the foreground. In case of a background patch we back-propagate only the error values of the class information and the other gradient values are set to zero.

3.2 Inference

The straight-forward approach for inference in our HN is to densely extract overlapping patches from the input image and evaluate the network for each patch separately. However, this implies many redundant operations, as the structure of CNNs allow for a more efficient inference.

In a CNN neighboring output units share common input units (see Figure 2). For a convolutional layer we only change the number of feature maps from the input of the layer to the output. Hence, a convolutional layer does not change the neighborhood relation of patches in the image. For pooling layers we have to be more careful, because they change

the size of the patches. However, if the patch size is a multiple of the pooling size, the patch neighborhood can be related. For example, let's assume two patches which are x pixels apart. Having a pooling size of y, after pooling the patches are $\frac{x}{y}$ pixels apart.

For a fast inference, we can present the whole image as input to the CNN. Before the first fully-connected layer, we rearrange the intermediate result into a matrix, where each patch is represented as one column-vector. We note that our approach is similar to [22], with the exception that instead of replacing the fully connected layers by convolution operations with kernels of size 1×1, we reshape the values.

Finally, for each patch we obtain a classification and a regression $\mathbf{y}_{(i,j)} = \left(y_{(i,j),c}, \mathbf{y}_{(i,j),r}\right)^T$, where (i, j) denotes the center of the patch. We only consider regression values of patches, where the foreground probability $y_{(i,j),c}$ is above an application dependent threshold.

4 Evaluation

In this Section we demonstrate the performance of our proposed HNs on two challenging computer vision tasks. The first task deals with the head pose estimation from consumer depth cameras (Section 4.1) and the second task demonstrates the applicability to facial feature localization in color images (Section 4.2). In both areas derivatives of HFs have shown great performance and we demonstrate that our approach improves on those results without the need of handcrafted features.

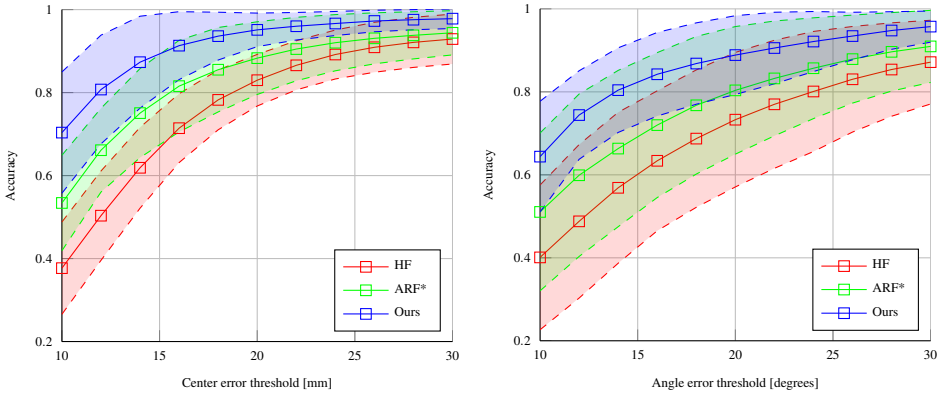
4.1 3D Head Pose Estimation

In the first task, we evaluate the performance of HNs for head pose estimation in 3D. We follow the experimental setup of Fanelli et al. [10, 11]. Given a depth image we train a patch-wise classification and regression model that estimates for a previously unseen image the head center in 3D and pose in Euler angles.

Experimental Setup For training we randomly extract patches of size 100×100 pixels from the depth images ($K = 1$). For each patch we store if it belongs to the head ($t_{s,c} = 1$) or to the background ($t_{s,c} = 0$). Further, for the regression part we extract the offset vector between the patch center and the head center as well as the head pose given in Euler angle ($\mathbf{t}_{s,r} \in \mathbb{R}^6$) for head patches. For the two convolutional layers, we train $K_C = 48$ filter kernels and use $F = 2048$ neurons in the fully-connected layer. The learning rate η is set to 10^{-2} and is exponentially decaying with a rate of 0.998. For the momentum coefficient μ we follow the strategy of Sutskever et al. [24]. Further, we keep the weighting coefficients of the error functions (λ_c, λ_r) equal to 1.0 throughout training. To aid generalization we apply a small weight decay of 0.0005. An additional usage of Dropout [15] did not lead to a better performance, but increased the time needed for training. Hence, we do not utilize it in the evaluation.

During testing we apply our HN on the whole depth image as explained in Section 3.2. This is equivalent to densely extracting patches on a regular grid from the image and a stride of 4. A patch is only allowed to vote for a head center and pose if the foreground probability is large enough, $y_{(i,j),c} > 0.99$. Using the mean-shift variant of [10, 11] we find a single mode that defines the resulting head center and pose.

For the head pose estimation experiments we employ the public available Biwi Kinect Head Pose Database [11]. The database contains 24 sequences of 20 persons, where every



(a) (b)

Figure 3: Accuracy for the head center estimation error (a) and the angle error (b) of the HF [10, 11], ARF* [21] and our approach. The curves visualize the fraction of correct estimates over an increasing success threshold. We evaluate our approach on five different splits, where for one split we randomly select all sequences from two persons as testset and the remaining sequences as trainset. The solid lines represent the mean over those five runs, whereas the shaded areas visualize the standard deviation.

sequence consists of ~600 depth images with corresponding annotation (head center in mm and head pose as rotation matrix). For each depth frame exists also a corresponding color image that is not utilized in this evaluation. We compare our method to the original HFs based approach [10, 11] and with the combination of Alternating Decision Forests and Alternating Regression Forests (ARF*) [21]. For those two methods we use the parameter setting and implementation provided by [21].

Results Following [10, 11, 21], we present the results of our method as the percentage of correctly predicted head centers and poses over an increasing success threshold. We experienced that especially the Forest based methods obtain significantly varying results for different splits of the database. Therefore, we conduct five runs, where in each run we randomly select all sequences of two persons as testset and use the remaining sequences as trainset. For a visualization of the mean error of the estimates and the corresponding standard deviation over those five runs see Figure 3.

We observe that our proposed method clearly outperforms the HF and the ARF* in terms of head center error and head pose error. For a success threshold of 20mm HNs locate 95.11% of the head centers correctly, whereas the ARF* and the HF achieve only 88.30% and 82.99%, respectively. In case of the head pose estimation the error of our method is even less if compared to HFs and ARFs*. For a success threshold of 20° our method achieves 88.86% compared to 80.37% (ARF*) and 73.29% (HF). Further, the smaller standard deviation indicates that the result of our approach does not as heavily depend on the split as both Forest based methods.

In Table 1 we summarize the raw errors of the six single regression variables, the accumulated errors of the head center and pose estimation, and the fraction of missed head detections. Again, HNs achieve similar or better results in all single variables, as well as in

Method	X (mm)	Y (mm)	Z (mm)	Center (mm)	Yaw (°)	Pitch (°)	Roll (°)	Angle (°)	Missed
HF [10, 11]	6.87 ± 6.67	7.35 ± 5.63	4.69 ± 3.39	12.82 ± 6.81	5.71 ± 6.07	9.73 ± 8.86	5.85 ± 5.18	14.26 ± 10.02	0.05
ARF* [21]	5.47 ± 5.55	6.24 ± 6.08	4.12 ± 3.13	10.76 ± 6.91	5.46 ± 5.76	7.78 ± 7.94	4.99 ± **4.43**	12.22 ± 9.04	0.03
Ours	**3.80** ± **4.48**	**4.55** ± **4.03**	**3.70** ± **3.03**	**8.13** ± **5.30**	**3.84** ± **3.73**	**6.68** ± **6.61**	**4.33** ± 4.93	**9.77** ± **8.04**	**0.01**

Table 1: Mean and standard deviation of the regression errors of HF [10, 11], ARF* [21] and our approach. The position of the head center is given in mm in 3D space and the Euler angle in degree. A detection is missed if the estimated head center is 50mm away from the ground-truth annotation.

the accumulated ones. Further, the fraction of missed detections is smaller when compared with the HF and the ARF*.

4.2 Facial Feature Localization

In this second task, we evaluate our method for the task of facial feature localization. We randomly extract patches of size 40×40 pixels from the 2D training images ($K = 3$). For each patch we store the information if it belongs to the head ($t_{s,c} = 1$) or not ($t_{s,c} = 0$). Additionally, for the regression values we include the offset vectors from the patch center to the L facial feature locations ($\mathbf{t}_{s,r} \in \mathbb{R}^{2L}$). The other training relevant parameters in this experiment are kept the same as in Section 4.1.

For testing we apply the trained HN on the whole image, which is equal to separately inferring the patches extracted from the image with a stride of 4. Further, a patch is only allowed to vote for a facial feature, if the foreground probability is above a threshold $\mathbf{y}_j^{(1)} > 0.99$. With mean-shift we cluster each facial feature point, whereas the foreground probability is used as an additional weight in combination with the length of the voting vector as in [6].

Experimental Setup In this experiment we use the Labeled Faces in the Wild (LFW) dataset [6, 29]. LFW contains facial images of 5,749 persons, where 1,680 have more than one image in the dataset. The images vary in appearance (lighting conditions, resolution and quality) and the persons have different poses, gender, race and occlusions. For 13,233 images annotations of 10 facial features and a discrete head pose (left profile, left, frontal, right, right profile) exist. We incorporate the 5 discrete head poses via a binary encoding $\mathbf{t}_{s,h} \in \{0,1\}^5$ into the target values by extending $\mathbf{t}_s = \left(t_{s,c}, \mathbf{t}_{s,h}, \mathbf{t}_{s,r}\right)^T$ and expand our objective function. Hence, we utilize the cross-entropy error $E_{s,h}(\theta)$ and add it to the final objective function $E_s(\theta)$ as weighted term as follows:

$$E_{s,h}(\theta) = -\sum_{i=1}^{5} \mathbf{t}_{s,h}^{(i)} \ln \mathbf{y}_{s,h}^{(i)} \tag{7}$$

$$E_s(\theta) = \lambda_c E_{s,c} + \lambda_r E_{s,r} + \lambda_h E_{s,h}. \tag{8}$$

Results We use the evaluation scheme proposed in [6] and perform a ten-fold cross validation. For measuring the accuracy, we compute the distance between the estimated facial feature location and the ground truth annotation as fraction of the inter-ocular distance. The results of our approach, of Conditional Random Forestss (CRFs) [6] and of Robust Cascaded Pose Regressions (RCPRs) are presented in Table 4(a).

LFW

Method	F. F. Error.	H. P. Error.
CRF [6]	12.0	27.85
RCPR [2]	5.3	-
Ours	6.3	21.83
Human	4.5	-

(a)

COFW

Method	F. F. Error.
RCPR [2]	8.4
Ours	14.1
Human	5.6

(b)

(c)

Figure 4: Performance of HNs compared to CRFs [6], RCPRs [2] and human performance on the LFW dataset (a). The facial feature errors are measured as percentage of the inter-ocular distance an the head pose error states the percentage of falsely classified head poses. Without any retraining HNs perform also reasonable on the challenging COFW dataset [2]. We visualize 9 of 48 filter kernels of the first layer in (c).

Our method, as others, reach almost human performance, but compared to [6] and [2] we only need to train one model that is able to jointly predict head pose and facial feature locations. For the estimation of the head pose we gain even higher values than the CRF and in contrast to the RCPR we do not rely on an additional face detector.

4.3 Remarks

The above results demonstrate the versatility of our method. Figure 4(c) shows some of the filters of the first convolutional layer after training on the LFW dataset. The network learned reasonable color blob and gradient filters that do not depend on a specific dataset. If we apply the HN of Section 4.2 on the COFW dataset [2] we obtain an average error of 14.1% without any retraining (see Table 4(b)). Im comparison, RCPR [2] achieves an average error of 8.4% trained on the COFW dataset and also learning the occlusion state of the features[1].

As we have implemented our method utilizing a GPU, there is at the moment no fair comparison with other methods in terms of runtime possible. Our implementation allows for 10 to 15 frames per second (fps) depending on the image size. If we crop the images as in [2, 6, 10, 11] we gain up to 20 fps.

5 Conclusion

We presented *Hough Networks (HNs)*, a novel approach that combines the principle of Hough Forests (HFs) with the Convolutional Neural Network (CNN) model. The method performs a simultaneous classification and regression on image patches. In contrast to Random Forests (RFs), a CNN is able to learn higher-order feature representation and does not rely on handcrafted features. Further, its architecture allows for fast inference if the patches are densely extracted along a regular grid. We evaluated our method on two computer vision tasks: head pose estimation and facial landmark localization. In both cases we achieved state-of-the-art or better results. In future, we plan to investigate the applicability of HNs to body pose and articulated hand posture estimation.

[1] We only evaluated on the 10 facial features that the LFW and COFW dataset have in common.

Acknowledgments

This work was supported by *Infineon Technologies Austria AG* and the Austrian Research Promotion Agency (FFG) under the *FIT-IT Bridge* program, project #838513 (TOFUSION).

References

[1] Michael D. Breitenstein, Daniel Küttel, Thibaut Weise, Luc J. Van Gool, and Hanspeter Pfister. Real-time face pose estimation from single range images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[2] Xavier P. Burgos-Artizzu, Pietro Perona, and Piotr Dollár. Robust face landmark estimation under occlusion. In *International Conference on Computer Vision*, pages 1513–1520, 2013.

[3] Timothy F. Cootes and Christopher J. Taylor. Active shape models - 'smart snakes'. In *British Machine Vision Conference*, pages 1–10, 1992.

[4] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[5] Timothy F. Cootes, Mircea C. Ionita, Claudia Lindner, and Patrick Sauer. Robust and accurate shape model fitting using random forest regression voting. In *European Conference on Computer Vision*, pages 278–291, 2012.

[6] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc J. Van Gool. Real-time facial feature detection using conditional regression forests. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2578–2585, 2012.

[7] Piotr Dollár, Peter Welinder, and Pietro Perona. Cascaded pose regression. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1078–1085, 2010.

[8] Michael Egmont-Petersen, Dick de Ridder, and Heinz Handels. Image processing with neural networks - a review. *Pattern Recognition*, 35(10):2279–2301, 2002.

[9] Gabriele Fanelli, Juergen Gall, and Luc J. Van Gool. Real time head pose estimation with random regression forests. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 617–624, 2011.

[10] Gabriele Fanelli, Thibaut Weise, Juergen Gall, and Luc J. Van Gool. Real time head pose estimation from consumer depth cameras. In *German Association for Pattern Recognition*, pages 101–110, 2011.

[11] Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc J. Van Gool. Random forests for real time 3d face analysis. *International Journal of Computer Vision*, 101(3):437–458, 2013.

[12] Juergen Gall and Victor S. Lempitsky. Class-specific hough forests for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1022–1029, 2009.

[13] Juergen Gall, Angela Yao, Nima Razavi, Luc J. Van Gool, and Victor S. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202, 2011.

[14] Ross B. Girshick, Jamie Shotton, Pushmeet Kohli, Antonio Criminisi, and Andrew W. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *International Conference on Computer Vision*, pages 415–422, 2011.

[15] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[16] Michael Kass, Andrew P. Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.

[18] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995.

[19] Erik Murphy-Chutorian and Mohan M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31 (4):607–626, 2009.

[20] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[21] Samuel Schulter, Christian Leistner, Paul Wohlhart, Peter M. Roth, and Horst Bischof. Alternating regression forests for object detection and pose estimation. In *International Conference on Computer Vision*, pages 417–424, 2013.

[22] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations*, 2014.

[23] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3476–3483, 2013.

[24] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

[25] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013.

[26] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[27] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[28] Michel François Valstar, Brais Martínez, Xavier Binefa, and Maja Pantic. Facial point detection using boosted regression and graph models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2729–2736, 2010.

[29] Hai Wang, Bong-Nam Kang, and Daijin Kim. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, 2007.

[30] Yan Yan, Elisa Ricci, Subramanian Ramanathan, Oswald Lanz, and Nicu Sebe. No matter where you are: Flexible graph-guided multi-task learning for multi-view head pose classification under target motion. In *International Conference on Computer Vision*, pages 1177–1184, 2013.

[31] Heng Yang and Ioannis Patras. Face parts localization using structured-output regression forests. In *Asian Conference on Computer Vision*, pages 667–679, 2012.

[32] Feng Zhou, Jonathan Brandt, and Zhe Lin. Exemplar-based graph matching for robust facial landmark localization. In *International Conference on Computer Vision*, pages 1025–1032, 2013.