1. Introduction

Currently, the actual task of creating video analytics systems that allow in few seconds detecting people on the street, in supermarkets, airports and railway stations. Detection of people on photos and videos is a difficult task, because the detection system should be resistant to changes in brightness and shape, the orientation of an object relative to the camera. People can often occur among other people or in interior. Therefore, it is better to use the upper part of the human (head-shoulder) to detect people in difficult conditions.

In recent years, researchers have developed many methods of head-shoulder detection. The main differences between the methods are due to feature selection for pattern recognition and classifier type. As features for pattern recognition researchers often choose the HOG and LBP features. Shape features are used in\(^\text{3-5}\). In\(^\text{3}\) a method was proposed based on the enhanced contour of the head (En-Contour) and the HOG and LBP features. In\(^\text{4}\) to describe the forms edgelets – short line segments, curves were used. In\(^\text{5}\), the authors proposed an approach based on the omega-shape features. In\(^\text{5}\), the authors used the Aggregated channel Features for the detection of the upper body. As can be seen from the literature review, most of the work uses the handcrafted features. For classification, researchers often use methods of boosting and support vector machine\(^\text{8}\). A fundamentally different approach is to use the deep neural network models (deep neural network autoencoder and deep convolutional networks\(^\text{9}\)) for automatic feature extraction.

The benefit of applying deep neural network models is the automatic extraction of informative features, the ability to use GPU as in the training process and recognition process. The disadvantage of deep neural network models for the problem of detecting people in the video stream is relatively slow on the CPU (which can be a constraint in the application of this approach on mobile and embedded platforms). In contrast to the deep neural network models, the best performance...
can be achieved when using handcrafted features. For example, to calculate edgelets only some of the pixels of the image are used, which leads to a significant increase in performance in recognition process.

In this article a method of human upper body detection is proposed, based on the use of HOG features and features extracted using deep neural network of autoencoders.

2. Dataset

In this work, the positive examples were extracted from the TUD Multiple View Pedestrian database, in which the people are represented by 8 views (Figure 1). The negative images that do not contain people were taken from the Nicta Dataset. We also used positive and negative examples collected from online surveillance cameras. Images collected from online cameras were marked manually, which provided us with the exact location of Head-shoulders objects. The dataset was divided into training (50%), test (25%) and validation (25%).

3. Concept Headings

The input of algorithm is an RGB image. First a preprocessing of the input image is performed, which is to convert RGB image into image gray level and then to scale. Then preprocessed image is sampled into parts (each part is described by a rectangle). After that, each image fragment is detected using two cascades (Figure 2): the HOG cascade and the cascade based on automatically extractable features. The main task of HOG-cascade is a quick filtering of the large number of false hypotheses.

The main objective of the cascade based on automatically extractable features is refining the solutions obtained by using the HOG-cascade.

3.1 HOG-Cascade

Training HOG-cascade was performed using the module opencv_traincascade.exe, that is part of OpenCV Computer Vision Library. Training of the cascade was carried out with different parameters. Training settings were varied to obtain the best Recall, FPPI. The main parameters and their description are shown in Table 1.

![Figure 1. Head-shoulder images extracted from the dataset TUD_MultiView_pedestrian.](image1)

![Figure 2. Block diagram of the head-shoulder detection.](image2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>The boundaries of parameter variations</th>
</tr>
</thead>
<tbody>
<tr>
<td>nStages</td>
<td>The number of cascade stages</td>
<td>10…20</td>
</tr>
<tr>
<td>StageRecall</td>
<td>Target Recall for stage</td>
<td>0.97…0.999</td>
</tr>
<tr>
<td>Depth</td>
<td>The maximum depth of the tree classification stage</td>
<td>1…4</td>
</tr>
<tr>
<td>nWeaks</td>
<td>The maximum number of weak classifiers stage</td>
<td>30…1000</td>
</tr>
<tr>
<td>PosRatio</td>
<td>The ratio of positive examples used for training</td>
<td>0.5…0.95</td>
</tr>
<tr>
<td>PosNegRatio</td>
<td>The ratio of positives and negatives</td>
<td>0.3…3.33</td>
</tr>
</tbody>
</table>

Detection quality was estimated using the Recall and FPPI (false positives per image). Recall is calculated according to the equation:

$$Recall = \frac{TP}{TP + FN}$$  \hspace{1cm} (1)
where \( TP \) is the number of true positives detected; \( NP \) is the total number of positives.

FFPI metric is calculated according to the Equation:

\[
FFPI = \frac{N_{sp}}{N}
\]

where \( N_{sp} \) – the total number of objects detected on the image, on which there are no positives, \( N \) – the total number of negative images (on which there are no positives).

Figure 3 presents the results of testing the quality of HOG-cascades, training with different parameters on the training data set. Figure 4 presents the results of testing the quality of HOG-cascades on the test data set.

As can be seen from Figures 3, 4, the low level of FPPI can be achieved only by decreasing the Recall. Therefore, as a result of many experiments, optimal parameters for training the HOG-cascade have been found, which are presented in Table 2.

### Table 2. The optimal parameters for training the HOG-cascade

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nStages</td>
<td>16</td>
</tr>
<tr>
<td>StageRecall</td>
<td>0.997</td>
</tr>
<tr>
<td>Depth</td>
<td>2</td>
</tr>
<tr>
<td>nWeaks</td>
<td>400</td>
</tr>
<tr>
<td>PosRatio</td>
<td>0.89</td>
</tr>
<tr>
<td>PosNegRatio</td>
<td>1.0</td>
</tr>
</tbody>
</table>

The results of testing the quality of the HOG-cascade, trained with the optimal parameter values are presented in Table 3.

### Table 3. Quality HOG-cascade trained at optimal parameters

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>FPPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>0.95</td>
<td>4.4</td>
</tr>
<tr>
<td>Test set</td>
<td>0.90</td>
<td>4.5</td>
</tr>
</tbody>
</table>

The performance of the HOG-cascade for video 640x480 is 6 FPS on the CPU Intel Core i5-3570.

### 3.2 Cascade based on Automatically Extracted Features

Cascade based on automatically extracted features are trained on a set of negative and positive examples recognized by the HOG-cascade as positive. As the HOG-cascade already filters out false hypotheses well, after filtering examples HOG-cascade number of negatives is about 5 times less than the number of positive examples.

To build a balanced dataset, a set of negatives was expanded by generating examples based on the available negatives. The size of the training set of examples was approximately 50K of examples for each class.

The work of the cascade is based on recognizing automatically extracted features. In the process of detection features have been extracted by using deep neural network autoencoder. Then the extracted features are recognized using the binary classifier (see Figure 5).
Head-Shoulder Detection using Deep Autoencoders

For training the deep autoencoder, patches of size 6x6 are extracted from negative and positive images (in experiments we used fragments of various sizes, the 6x6 size was chosen so as to achieve the best performance of the cascade). The total number of extracted patches is 750K. Next, the extracted patches are normalized using the equation:

$$y_i = \frac{x_i - \bar{x}}{\sigma}, i = 0...N - 1,$$

where $y_i$ is the i-th pixel in the normalized patch; $x_i$ - the i-th pixel in the source patch; $\bar{x}$ - the average value of brightness of the pixels of the source patch; $\sigma$ - the standard deviation of the pixels of the source patch; $N$ - the total number of pixels of the patch.

After normalization of patches the ZCA Whitening transformation was applied.

Thus, training the deep autoencoder to extract features is based on the normalized and whitened patches.

3.3 Deep Autoencoders Training

In this work, deep autoencoder is implemented using auto-associative neural networks of type multilayer perceptron (MLP). Work auto-associative neural network with one hidden layer (Figure 6.) can be expressed using the equations:

$$\overrightarrow{y}(\overrightarrow{x}, \overrightarrow{w}) = \varphi(D\overrightarrow{h})$$

$$\overrightarrow{h} = \varphi(H\overrightarrow{x})$$

where $\overrightarrow{x} \in \mathbb{R}^N$ is the input vector of the neural network; $\overrightarrow{y} \in \mathbb{R}^N$ is the output vector of the neural network (the input and output vectors for an auto-associative neural network have the same dimension); $\overrightarrow{w}$ is the vector of the weights of the auto-associative neural network, which consists of the coefficients of the matrices of encoding $H \in \mathbb{R}^{p \times N}$ and decoding $D \in \mathbb{R}^{p \times N}$, layers; $h$ is the output vector of the encoding layer of an auto-associative neural network; $\varphi$ is the activation function of the neurons in the encoding and decoding layers. In this research work for encoding and decoding layer is used the same activation function, which is a symmetric sigmoid function.

$$\varphi(x) = \frac{1}{1 + \exp(-x)} - \frac{1}{2}$$

Figure 5. Block diagram of cascade based on automatically extracted features.

Figure 6. Autoencoder with one encoding layer $h$. Weights vector of the autoencoder is a vector $\overrightarrow{h}(\overrightarrow{x}, \overrightarrow{H})$ which is the solution of the optimization problem using the regularization procedure according to the equations:

$$\overrightarrow{w} = \arg \min \left\{ \sum_i \|\overrightarrow{y}(\overrightarrow{x}_i, \overrightarrow{w}) - \overrightarrow{x}_i\| + L_1(\lambda, \overrightarrow{w}) \right\},$$

$$L_1(\lambda, \overrightarrow{w}) = \lambda\|\overrightarrow{w}\|$$

where $\overrightarrow{w}$ is the desired vector of weights of the autoencoder; $\{\overrightarrow{x}_i\}$ - examples of the training dataset ($\overrightarrow{x}_i \in \mathbb{R}^N$ is an input vector which represents an image object of the specified class); $\sum_i \|\overrightarrow{y}(\overrightarrow{x}_i, \overrightarrow{w}) - \overrightarrow{x}_i\|$ is the error function, which represents the sum of squared differences between the output of the autoencoder and the corresponding training input; $L_1(\lambda, \overrightarrow{w}) = \lambda\|\overrightarrow{w}\|$ is the component for the regularization.
of the complexity of the autoencoder (complexity regularization is performed using the method of weight decay\(^{(14)}\)).

Autoencoder with one hidden layer allow to extract low-level features of objects which may not be enough to achieve a high level of quality recognition. To achieve the best quality of the recognition was trained deep autoencoder based on auto-associative multilayer neural network(Figure 7). Training of the deep autoencoder was performed iteratively using the method, which is often called “Deep learning”\(^{(15,16)}\).

![Figure 7. Deep multilayer autoencoder with encoding layer h.](image)

After training the deep autoencoder, the outputs of its hidden layer are used to patch representation. Each neuron of the hidden layer divides the feature space into two half-spaces. Therefore, each patch with a threshold rule can be assigned to one of two half-spaces and are encoded using 1 and 0. Thus, the set of outputs of neurons of the hidden layer generates a binary sequence that is used as a code patch. As a result of this encoding is that similar patches generate similar code. As a result of experiments it was found that the optimal structure of the deep autoencoder is a structure 36-24-16-8-16-24-36, the optimal number of encoding bits is 8. Thus, the number of codewords that can be represented by the patch image is 256.

### 3.4 The Calculation of the Outputs of the Deep Autoencoder

A fast calculation of the outputs of the deep autoencoder for many examples is performed using the operations of matrix multiplication (which takes the major computing time when calculating the output) from layer to layer. Suppose you have a matrix of inputs \(X_{m \times n}\), the matrix of weight coefficients of the layer \(W_{k \times m}\), and the biases layer \(B_{1 \times k}\):

\[
\begin{bmatrix}
  x_{11} & x_{12} & \cdots & x_{1m} \\
  x_{21} & x_{22} & \cdots & x_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{n1} & x_{n2} & \cdots & x_{nm}
\end{bmatrix}
\begin{bmatrix}
  w_{11} & w_{12} & \cdots & w_{1m} \\
  w_{21} & w_{22} & \cdots & w_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
  w_{k1} & w_{k2} & \cdots & w_{km}
\end{bmatrix}
+\begin{bmatrix}
  b_{11} \\
  b_{12} \\
  \vdots \\
  b_{1k}
\end{bmatrix}
\]

where \(n\) is the number of examples for calculation, \(m\) is the number of inputs of the layer, \(k\) is the number of neurons (output) layer.

It is necessary to compute the matrix of outputs of the layer \(Y_{m \times k}\):

\[
\begin{bmatrix}
  y_{11} & y_{12} & \cdots & y_{1k} \\
  y_{21} & y_{22} & \cdots & y_{2k} \\
  \vdots & \vdots & \ddots & \vdots \\
  y_{n1} & y_{n2} & \cdots & y_{nk}
\end{bmatrix}
\]

Then the outputs of the layer are calculated in two stages according to the Equations:

\[
O = XW^T, \quad y_{ij} = f(o_{ij} + b) \tag{6}
\]

Where \(f(x)\) – the specified activation function; \(o_{ij}\) – output of \(j\)-th neuron of the current layer for the \(i\)-th example, \(b\) – the bias for the \(j\)-th neuron of the current layer, \(y_{ij}\) the outputs of the current layer.

The operation of matrix multiplication can be efficiently parallelized on modern multi-core processors using software and hardware technologies of the Intel MKL,\(^{(17)}\) OpenBLAS,\(^{(18)}\) NVIDIA CUDA.\(^{(19)}\)

### 3.5 Feature Extraction with Deep Neural Network Autoencoder

After training the deep autoencoder, the features for all negative and all positive images were extracted. The block diagram of feature extraction with deep autoencoder is
Head-Shoulder Detection using Deep Autoencoders

shown in Figure 8. First, the input image is resized to 32x32. Then for the image patches of size 6x6 were extracted and normalized. The total number of patches is 27x27. Next, each patch is encoded by a deep autoencoder, and computes the error of reconstruction of the patch. This creates two matrices of size 27x27 (matrix codes and the error matrix reconstruction). Error of the reconstruction is computed using the L2 metric.

The next step is to use the MAX pooling operation with a block size of 2x2 and step 1. The use of MAX Pooling operation allows increasing the stability of local shifts and rotations, which are typical of natural images. The Max pooling operation is performed based on the matrix reconstruction error. For each 2x2 block is selected the code, for which the reconstruction error is the smallest one. The result is a matrix code size 26x26, which is used as features of the image (overall dimension of the feature vector is 676).

Figure 8. Block diagram of feature extraction by deep autoencoder.

Figure 9 shows examples of images of head-shoulder and the corresponding coded images.

Figure 9. Examples of images of head-shoulder and the corresponding coded images.

3.6 Binary Classification

After extracting features for all images from the training set, a binary classifier was trained. A binary classifier is the ensemble of discrete Naive-Bayes classifiers. The ensemble of discrete Naive-Bayes classifiers is trained using the Adaboost algorithm. The choice in favor of a discrete Naive-Bayes classifier was done due to its high performance in detecting vectors of features of high dimensionality. This is because in the process of classification a discrete Naive-Bayes classifier can only be used for practical implementation in the operation of addition. The decision rule in this case is determined by the Equation:

\[
\sum_i \log p(x_i^+ | c = 1) + \log p(c = 1) > \sum_i \log p(x_i^- | c = 0) + \log p(c = 0)
\]  

Where \( \log = p(c = 1) \) – the logarithm of the prior probability of positive class; \( \log = p(c = 0) \) – the logarithm of the prior probability of negative class,
\( \log p(x_i | c = 1) \) and \( \log p(x_i | c = 1) \) are the logarithms of the likelihood which is the probability of the predictor given the class.

When using balanced datasets in which the number of positives equals the number of negatives expression 7 can be rewritten in the form:
\[
\sum \log p(x_i | c = 1) > \sum \log p(x_i | c = 0)
\]  
(8)

The logarithms of likelihood \( \log p(x_i | c = 1) \) and \( \log p(x_i | c = 0) \) can be pre-calculated, so the process of classification uses only operations of adding and fetching from memory.

Figure 10, 11 show the dependence of the accuracy of the ensemble from the number of weak classifiers for training and testing dataset.

Figure 10. The dependence of the accuracy of the ensemble from the number of weak classifiers on the training dataset.

The recognition time of one image by using a cascade based on automatically extracted features is 1.8 milliseconds (~1.5 milliseconds is feature extraction and 0.3 milliseconds is classification) at CPU Intel Core i5-3570.

3. Results

The quality testing results of the developed two cascaded detector on the validation set are shown in Table 4. For comparison, the same data were performed testing the detector of the upper part of the pedestrians from the OpenCV Library.

Figure 11. The dependence of the accuracy of the ensemble from the number of weak classifiers on the testing dataset.

Figure 12. Examples of the developed detector (red rectangles) and the detector of OpenCV (blue rectangles).
As can be seen from Table 4, we have developed a detector that outperforms the detector from the OpenCV, both in quality and performance.

Table 4. The results of testing the quality of the developed detector and the OpenCV on the validation dataset

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>FPPI</th>
<th>Performance at CPU Intel Core i5-3570, FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developed detector</td>
<td>0.85</td>
<td>0.5</td>
<td>5.5</td>
</tr>
<tr>
<td>OpenCV</td>
<td>0.34</td>
<td>5.3</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Figure 12 shows examples of the developed detector and the detector of OpenCV for real video. Red rectangles display the results of the developed detector; blue rectangles show the results of the OpenCV detector.

4. Discussion

Our study showed that the combined use of HOG-features and automatically extracted features allows accomplishing high quality and performance in the problem of head-shoulder detection. High performance is achieved by applying HOG-cascade, which filters a significant part of false hypotheses. High quality is attained by using a cascade based on automatically extracted features.

5. Conclusion

The results show that the proposed method is comparable or slightly outperforms the state-of-the-art methods. The developed approach allows achieving a high level of Recall (85%), but still can cause difficulties in practical application, due to a relatively high level of false positives (FPPI: 0.5).

When using a detector to real video, it would be possible to reduce the level of false positives by using a motion mask or filter based on the model of the scene. But we also see the ability to improve the quality by increasing the number of training examples, and using the adaptation of the detector for installed cameras, as each specific video camera class of negatives can be narrower than the class of negatives used for training a General detector. On the other hand, the variation of types of head-shoulders may be below, therefore, can be achieved a higher level of Recall.

6. Acknowledgement

The authors thank Ministry of Education and Science of the Russian Federation for supporting this research. Funding Information. The research was financed by the Ministry of Education and Science of the Russian Federation under the Grant Contract dated 24 November, 2014, (grant number 14.579.21.0070, contract unique identifier RFMEFI57914X0070), the grant was assigned for the research into "2D- and 3D-data fusion and processing software for object detection and recognition".

7. Author’s Contributions

Authors contributed to this research as follows:

**Pavel Vyacheslavovich Skribtsov**: Developed regularization procedure and autoencoder’s training algorithm, organized collection of training and validation datasets.

**Pavel Aleksandrovich Kazantsev**: Designed research plan and algorithm flow diagram, developed detection algorithm, carried out validation and tests.

**Aleksey Vladimirovich Dolgopolov**: Developed classification algorithm, implementation algorithms, optimization computation.

8. References

2. Zeng C, Ma H. Robust head-shoulder detection by PCA based multilevel hog-lbp detector for people counting. Proc Inter Conf Pattern Recognition (ICPR); 2010.
7. Fitte-Duval L, Mekonnen A, Lerasle F. Upper body detection and feature set evaluation for body pose classification. Proceedings of the 10th International Conference on


