Visuelle Perzeption für Mensch-Maschine Schnittstellen

Vorlesung, WS 2009

Prof. Dr. Rainer Stiefelhagen
Dr. Edgar Seemann

Interactive Systems Laboratories
Universität Karlsruhe (TH)

http://cvhci.ira.uka.de/teaching/visionhci
rainer.stiefelhagen@kit.edu
seemann@pedestrian-detection.com
Face Detection 2/2

Dr. Rainer Stiefelhagen
Roadmap

- Color based face detection
- An Ellipsoid head model
- **Artificial Neural Networks**
- Feature-based classifier cascades (Viola & Jones)
Neural Network Based Face Detection
A Simple Neuron Model

Different outputs possible, depending on activation function:
Neural Network Based Face Detection Topologies

- Fully connected
- Feed-forward
- Recurrent
Neural Network Based Face Detection

Parameters

Adjustable Parameters are
- Connection weights
  - are to be learned
- Activation function (fixed)
- Number of layers (fixed)
- Number of neurons per layer (fixed)
Neural Network Based Face Detection
Training (1)

Learning the connectionist weights is achieved by descending the gradient of the resulting output error $E$: $\Delta W = -\eta \cdot \nabla E(W)$

(⇒ Backpropagation Algorithm)
Neural Network Based Face Detection Training (2)

Standard Error Backpropagation (1):
Given sample patterns \((p_1, t_1), \ldots, (p_n, t_n)\) with specified target outputs (groundtruth) \(t_p\) for each pattern \(p\)

- Error function for a single pattern \(p\) accumulating over all output neurons \(j\):
  \[
  E = \frac{1}{2} \sum_j (t_{pj} - \text{out}_{pj})^2
  \]

- Deriving from \(\Delta W = -\eta \cdot \nabla E(W)\) weights are adjusted as
  \[
  \Delta w_{ij} = \sum_p -\eta \cdot \frac{\partial E_p}{\partial w_{ij}}
  \]
Neural Network Based Face Detection
Training (3)

Standard Error Backpropagation (2):

\[ \Delta w_{ij} = \sum_p -\eta \frac{\partial E_p}{\partial w_{ij}} \]

\[ -\frac{\partial E_p}{\partial w_{ij}} = -\frac{\partial E_p}{\partial a_{pj}} \frac{\partial a_{pj}}{\partial w_{ij}} = -\frac{\partial E_p}{\partial w_{ij}} \frac{\partial}{\partial a} \sum_{i=1}^n w_{ij} \text{in}_j = -\frac{\partial E_p}{\partial a_{pj}} \text{in}_j = -\frac{\partial E_p}{\partial \text{out}_{pj}} \frac{\partial \text{out}_{pj}}{\partial a_{pj}} \text{in}_j \]

Assume \( \Psi(a) \) to be \( \Psi(a) = \frac{1}{1 + e^{-a}} \Rightarrow \frac{\partial \Psi(a)}{\partial a} = \Psi(a)(1 - \Psi(a)) \)

2) \( \frac{\partial \text{out}_{pj}}{\partial a_{pj}} = \frac{\partial \Psi(a_{pj})}{\partial a_{pj}} = \Psi(a_{pj})(1 - \Psi(a_{pj})) \)

1) \[ -\frac{\partial E_p}{\partial \text{out}_{pj}} = (t_{pj} - \text{out}_{pj}), \text{if } j \text{ is output unit} \]

\[ -\frac{\partial E_p}{\partial \text{out}_{pj}} = \sum_k -\frac{\partial E_p}{\partial a_{pk}} w_{jk}, \text{if } j \text{ is hidden unit} \]
Neural Network Based Face Detection

Idea for image classification: remodel human visual perception
Neural Network Based Face Detection

General approach for detecting (upright, frontal) faces with NN:

- Network receives as input a 20x20 pixel region of an image
- output ranges from -1 (no face present) to +1 (face present)
- the neural network „face-filter“ is applied at every location in the image
- to detect faces with different sizes, the input image is repeatedly scaled down

Neural Network Based Face Detection

Network Topology

ANN Topology:
- 20x20 pixel input retina
- 4 types of receptive hidden fields
- One real-valued output
Neural Network Based Face Detection
System Overview

Figure 1: The basic algorithm used for face detection.
Neural Network Based Face Detection
Trainingsset

- **Training Set:**
  - 1050 normalized face images
  - 15 face images generated by rotating and scaling original face images
  - 1000 randomly chosen non-face images

Face Samples

Non-Face Samples
Neural Network Based Face Detection

Preprocessing:
- correct for different lighting conditions (overall brightness, shadows)
- rescale images to fixed size
- Often: extract relevant features (edges, FFT, wavelets, PCA, DCT, …)

Original window:

Best fit linear function:

Lighting corrected window:
(linear function subtracted)

Histogram equalized window:
Histogram equalization

- Defines a mapping of gray levels $p$ into gray levels $q$ such that the distribution of $q$ is close to being uniform
- Stretches contrast (expands the range of gray levels)
- Transforms different input images so that they have similar intensity distributions (thus reducing the effect of different illumination)
- Fast algorithm exists,
  - Transformation can be defined in terms of the cumulative histogram
Histogram Equalization (Algorithm)

- The probability of an occurrence of a pixel of level $i$ in the image is $p(x_i)$:
  \[ p(x_i) = \frac{n_i}{n}, i \in 0, ..., L - 1 \]

- define $c$ as the *cumulative distribution function*:
  \[ c(i) = \sum_{j=0}^{i} p(x_j) \]

- create a transformation of the form
  \[ y_i = T(x_i) = c(i) \]
  \[ y_i \in [0,1] \]
  \[ y'_i = y_i \cdot (\max - \min) + \min \]
  \[ y'_i \in [\min, \max] \]

$L$: number of gray levels, $n_i$: number of occurrences of gray level $i$
Unequalized image

Equalized image

Corresponding Histograms
Neural Network Based Face Detection
Training

Training Procedure:

1. randomly choose 1000 non-face images
2. train network to produce 1 for faces, -1 for non-faces
3. run network on images containing no faces. Collect subimages in which network incorrectly identifies a face (output > 0)
4. select up to 250 of these „false positives“ at random and add them to the training set as negative examples
Neural Network Based Face Filter

- Output of ANN defines a filter for faces
- Search
  - Scan input image with search window, apply ANN to search window
  - Input image needs to be rescaled in order to detect faces with different size
- Output needs to be post-processed
  - Noise removal
  - Merging overlapping detections
- Speed up can be achieved
  - Increase step size
  - Make ANN more flexible to translation
  - Hierarchical, pyramidal search
Neural Network Based Face Detection

Results on 130 images containing 507 faces (83 Million subregions examined):
- 92.5% detection rate at false detection rate of 1/96402 (862 false detections)
- 77.9% detection rate at false detection rate of 1/41Mio. (2 false detections)

Speed:
- best system, using two networks: 383 seconds per image
- tuned for speed (77% detection rate): 7.2 seconds per image
Neural Network Based Face Detection
Results
Neural Network Based Face Detection Results
Roadmap

- Color based face detection
- Contour shaping
- Artificial Neural Networks
- Feature-based classifier cascades (Viola & Jones)
Feature-based Face Detection

- Detection based on features, not pixels
  - Features can encode domain knowledge that is difficult to learn
  - Faster than a pixel-based system
  - Scale independent
Feature-based Face Detection

- Robust realtime detection of general objects consisting of
  - Features (for human faces)
  - Integral Image (to compute features)
  - Weak Classifier Cascade
  - Train the Cascade
Feature-based Face Detection:
Features (1)
Feature-based Face Detection:
Features (2)

Features applied on grayscaled subwindows of fixed size (e.g. 24x24 pixels)

- Over 180,000 rectangle features possible for each subwindow!
  (Different positions, sizes)
- Computing each feature’s value independently takes way too long!!
Feature-based Face Detection: Integral Image

\[ ii(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \]

\( ii(x,y) \) is the integral image
\( i(x,y) \) is the original image

computing in two recurrences:

\[ s(x, y) = s(x, y - 1) + i(x, y) \]

\[ ii(x, y) = ii(x - 1, y) + s(x, y) \]
Feature-based Face Detection: Feature Evaluation (1)

\[ \begin{align*}
\text{area}(A) &= i_{1} \\
\text{area}(A + B) &= i_{2} \\
\text{area}(A + C) &= i_{3} \\
\text{area}(A + B + C + D) &= i_{4} \\
\Rightarrow \text{area}(D) &= i_{4} + i_{1} - (i_{2} + i_{3}) \\
\end{align*} \]

(4 array references)
### Feature-based Face Detection: Feature Evaluation (2)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Array references</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangle</td>
<td>4</td>
</tr>
<tr>
<td>Type I</td>
<td>6</td>
</tr>
<tr>
<td>Type II</td>
<td>8</td>
</tr>
<tr>
<td>Type III</td>
<td>9</td>
</tr>
</tbody>
</table>
Feature-based Face Detection: Feature Evaluation (3)

- Computing one feature can now be computed very fast
  - Computing all features however still takes too much time!
- Still too many features to compute in real time
- Idea: Only learn significant features!
Feature-based Face Detection: AdaBoost (1)

- AdaBoost is used to boost classification performance of a simple learning algorithm (e.g. simple perceptron)
- Variant of AdaBoost is used to select features + train the classifier
- It combines a collection of weak classifier to form a stronger one
Feature-based Face Detection: AdaBoost (2)

- A weak classifier $h$ is a classifier with accuracy only slightly better than chance

- Boosting: combine a number of weak classifiers so that the ensemble is arbitrarily accurate
  - Allows the use of simple (weak) classifiers without the loss of accuracy
Feature-based Face Detection: AdaBoost (3)

- Combine weak classifiers $h_i \in \{-1, 1\}$ linearly to build a strong classifier $H$:

$$H = \text{sign}\sum_{i=1}^{N} w_i h_i$$
Feature-based Face Detection: AdaBoost (4)

Viola & Jones approach:

- Weak classifier $h_j(x)$ consists of a feature $f_j$, a threshold $\theta_j$, and a parity $p_j$

$$h_j(x) = \begin{cases} 
1 & \text{if } p_j f_j(x) < p_j \theta_j \\
0 & \text{else}
\end{cases}$$
Feature-based Face Detection: AdaBoost (5)

- AdaBoost…
  - searches over the set of possible weak classifiers and selects those with the lowest classification error
  - applies a greedy feature selection process
  - is efficient for selecting a small number of “good” features with significant variety
Feature-based Face Detection: AdaBoost (6)

- Given examples $x_i$ with annotations $(x_i, y_i)$ where $y_i \in \{-1, 1\}$
- Initialize weights: $w_{1,i} = \frac{1}{2m} \cdot \frac{1}{2l}$
- For $T=1..t$:
  - Normalize weights: $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
  - For each feature $j$, train a classifier $h_j$, restricted to using a single feature
    - That is, adjust threshold $\theta_j$ so $h_j$ classifies best between the weighted positive and negative samples
  - Choose classifier $h_t$ with lowest error $\varepsilon_t$
  - Update weights: $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$
    where $e_i=0$ if sample $x_i$ is classified correctly, $e_i=1$ otherwise and
    $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$
- Finally, the strong classifier then is:

$$H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \text{ where } \alpha_t = \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases}$$

38
Feature-based Face Detection: AdaBoost (7)

Updating weights forces subsequent classifiers to focus on misclassified samples instead of the same general sampleset as classifiers before did.
Feature-based Face Detection: Classifier Cascade

- Not all subwindows show faces but we need to apply our strong classifier on all subwindows still
- Idea: Build cascade of classifiers
  - Each subsequent classifier is more 'precise'
    - First layer detects all positive samples, but includes many false positives
    - Second layer focuses on false positives of first layer and is more complex
  - ...
- Fast rejection of non-face subwindows whereas face images will be passed through the cascade until the very end
Classifier Cascade

All Sub-windows

1 → T → 2 → T → 3 → T → Further Processing

Reject Sub-window

(auswendig lernen 😊)
Feature-based Face Detection: Cascade Training

- Reduce false positive rate & decrease detection rate with each subsequent layer
- For each stage, a target is set for minimum reduction and decrease. Target is achieved by adding more features to corresponding layer
- Adding layers until final target of false detections is met
- Each subsequent layer becomes more complex and include more features
Training the Cascade

Example: 10 stage cascade

Target detection rate: $D = 0.9$
Target false positive rate: $F = 10^{-5}$

\[
F = \prod_{i=1}^{10} f_i \quad \quad \quad D = \prod_{i=1}^{10} d_i
\]

⇒ Each stage:
- detection rate $d = 0.99$
- false positive rate $f = 0.3$
Feature-based Face Detection: Cascade Training

Define Target Rates \( F \) and \( D \) of final detector;
Assign \( f_i \) and \( d_i \) to each cascade layer;
while( \( F \) not reached ){
    add new layer;
    \( j = 1 \);
    while( \( f \) not reached ) {
        train the classifier with \( j \) features;
        adjust threshold to meet \( d \);
        \( j++ \);
    }
    delete detected negatives;
}
Feature-based Face Detection: Results (1)

- Training set:
  - 4916 hand labeled face images
  - 9544 non-face images
- Each layer was trained with all face images and 10,000 non-face subwindows (24x24 pixels)
  - For subsequent layers, this included previous false positives too
- 38 layers as target
Feature-based Face Detection: Results (2)

- Final number of features used in total: 6061
  - First layer: 1 feature
  - Second layer: 10 features
  - Fifth layer: 50 features
  - ... (38 layers)
- Pentium 3, 700 MHz: total scan of 384x288 pixels in about 0.067 sec. (15 times faster than previous approaches!)
- Large majority of subwindows are rejected in first or second layer
Feature-based Face Detection: Results (3)

- Evaluated on MIT & CMU frontal face database
  - 103 images of 507 frontal faces (labeled)
  - Number of scanned subwindows: 75,081,800
Feature-based Face Detection: Results (4)
Feature-based Face Detection: Results (5)
Conclusion

- Approach that is 15 times faster than any previous approach
- Can also be applied to detect general objects
- Learning techique and cascades have impact on a variety of other tasks
- Comparable detection rates
- Easily detects multiple faces per images
- Robust against illumination variances
References

