Face detection and tracking in video sequences using the modified census transformation

Christian Küblbeck *, Andreas Ernst

Department of Electronic Imaging, Fraunhofer Institute for Integrated Circuits, Am Wolfsmantel 33, 91058 Erlangen, Germany

Received 7 December 2004; received in revised form 8 July 2005; accepted 23 August 2005

Abstract

We present the combination of an illumination invariant approach to face detection combined with a tracking mechanism used for improving speed and accuracy of the system. We introduce illumination invariant local structure features for object detection. For an efficient computation we propose a modified census transform, which enhances the original work of Zabih and Woodfill [19] [Ramin Zabih, John Woodfill. A non-parametric approach to visual correspondence. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1996.]. The tracking is performed by means of continuous detection. We show that the advent of new rapid detection algorithms may change the need for traditional tracking. Furthermore the mentioned problems have a natural solution within the presented tracking by continuous detection approach. The only assumption on the object to track is its maximal speed in the image plane, which can be set very generously. From this assumption we derive three conditions for a valid state sequence in time. To estimate the optimal state of a tracked face from the detection results a Kalman filter is used. This leads to an instant smoothing of the face trajectory. It can be shown experimentally that smoothing the face trajectories leads to a significant reduction of false detections compared to the static detector without the presented tracking extension. We further show how to exploit the highly redundant information in a natural video sequence to speed-up the execution of the static detector by a temporal scanning procedure which we call ‘slicing’. A demo program showing the outcomes of our work can be found in the internet under http://www.iis.fraunhofer.de/bv/biometrie/ for download.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Face detection; Face tracking; Census transformation; Kalman filter; Illumination invariance

1. Overview

The paper arranged in three main parts. The first part deals with the face detection mechanism on each single frame. It deals with the problem of illumination invariant detection of faces. Often compensation with a simple lighting model followed by histogram equalization is performed like first proposed by Sung [16] in the context of face detection. Many other detection approaches adopt this strategy with minor modifications, see Refs. [13,14,18]. Schneiderman [15] and Viola [17] apply a simpler normalization to zero mean and unit variance on the analysis window. Illumination compensation demands more computational power than the classification of an image patch itself. For example Yang [18] presented the simple linear SNoW classifier for detection, which operates directly on pixel intensities. It would require only to accumulate the outcome of 400 lookup operations to classify an image patch of size 20×20 while alone the histogram equalization on the same patch takes a higher effort. Therefore, we advocate the use of inherently illumination invariant image features for object detection, which convey only structural object information. We propose the new feature set local structure features computed from a 3×3 pixel neighborhood using a modified version of the census transform [19]. We use this feature set with a four-stage classifier cascade. Each stage classifier is a linear classifier, which consists of a set of lookup-tables of feature weights. Detection is carried out by scanning all possible analysis windows of size 22×22 by the classifier. In order to find faces of various size the image is repeatedly downscaled with a scaling factor of \( S_{\text{mra}} = 1.25 \). This is done until the scaled image is smaller than the sub window size. For typical images of size 388×284 pixels and a sub window size of 22×22 we obtain 10 downscaled images, which constitute an image pyramid. To further speed up processing the pyramid is scanned using a grid search, which we introduced with our former work [8].
The second part shows how to use tracking mechanisms to increase speed as well as accuracy of the face detector. In literature many methods are described that can be used to determine the new object location. One simple method is to take the pixel block covered by the object at the current location and perform block-matching to locate the block in a new image [1]. In face-tracking mostly more elaborate model-based approaches are used. Edwards [5] for example uses a 2D active appearance model for tracking, while DeCarlo [3] proposes a 3D approach to also handle out-of-plane rotation. Similar work was done by Dornaika [4] or La Cascia [2], but based on a different feature set.

Pure tracking approaches as the ones mentioned above have two major deficiencies which often limit their application in real-world scenarios, namely:

- Initialization problem, i.e. how does the tracker know when a new object enters the scene and thus a new track has to be started.
- ‘Lost track’ problem, i.e. can we decide if the tracker lost track because the object has left the scene or due to a tracking error.
- In Ref. [10] the problem is for example termed ‘Entry and Exit’ of interesting objects from the scene and its solution is identified as being one of the key issues in fully automated tracking systems. Many approaches therefore need manual segmentation of the object in the first frame [12] or use color segmentation or image difference energy based methods [10], which fail in case of a moving camera. If tracking breaks down due to occlusion or tracking errors the algorithms are widely unable to recover. Most algorithms also have no criterion to decide whether the tracked object exits the scene, or at least do not give any experimental evidence that they are able to detect such events. With the advent of a new class of rapid detection algorithms like those proposed by Viola [17] and Li [11] or that developed in this work, see also [7], the motivation for tracking may change fundamentally. Now it is possible to treat each frame of a video sequence as static image and perform a full search for desired objects. As those algorithms have good detection performance and low error rates the ‘Entry and Exit’-problem is solved implicitly. In this work we suggest a tracking by continuous detection approach which means, that each new frame is processed by a real-time static face detector, which has state of the art detection properties and error rates [7]. The outcome of this detector is combined in an association step, using a prior motion model. The main difference to existing approaches is that the motion model is not used to restrict the search area of the detector but only to connect detections from different frames.

In the third part we show the outcome of our experiments on three videosequences. We compare the recognition rates between static detection and detection using the tracking feature.

2. Feature generation

2.1. Local structure features

The features used in this work are defined as structure kernels of size $3 \times 3$ which summarize the local spatial image structure. Within the kernel structure information is coded as binary information {0,1} and the resulting binary patterns can represent oriented edges, line segments, junctions, ridges, saddle points, etc. Fig. 1 shows some examples of this structure kernels. On a local $3 \times 3$ lattice there exist $2^9=512$ such kernels. Actually there are only $2^9-1$ reasonable kernels out of the $2^9$ possible because the kernel with all elements 0 and that with all 1 convey the same information (all pixels are equal) and so one is redundant and thus is excluded. At each location only the best matching kernel is used for description. The whole procedure can be thought of as non-linear filtering where the output image is assigned the best fitting kernel index at each location. In the next section we show the index of the best kernel can be obtained by a modified version of the census transform (Fig. 2).
2.2. The census transform

The census transform (CT) is a non-parametric local transform, which was first proposed by Zabih and Woodfill [19]. It is defined as an ordered set of comparisons of pixel intensities in a local neighborhood representing which pixels have lesser intensity than the center. In general the size of the local neighborhood is not restricted, but in this work we always assume a 3 × 3 surrounding as motivated in the last section. Let \(\mathcal{N}(x)\) define a local spatial neighborhood of the pixel at \(x\) so that \(x \in \mathcal{N}(x)\). The CT then generates a bit string representing which pixels in \(\mathcal{N}(x)\) have an intensity lower than \(I(x)\). With the assumption that pixel intensities are always zero or positive the formal definition of the process is like follows: Let a comparison function \(\zeta(I(x), I(x'))\) be 1 if \(I(x) < I(x')\) and let \(\otimes\) denote the concatenation operation, the census transform at \(x\) is defined as

\[
C(x) = \otimes_{y \in \mathcal{N}} \zeta(I(x), I(y)). \tag{1}
\]

Note that \(C(x)\) is not an intensity or similarity coefficient as with linear transforms. Moreover all bits of \(C(x)\) have the same significance level. Thus \(C(x)\) may be interpreted as the index of a structure kernel defined on \(\mathcal{N}(x)\) with the center set to zero. In this interpretation the pixels in the kernel represent the outcome of the single census comparison \(\zeta(\cdot)\) at the corresponding location in the neighborhood. This interpretation links the census transform to the local structure kernels introduced in the last section. The census transform in its original form does not always result in the best describing kernel like shown below, because not all kernels are computable.

2.3. The modified census transform

Using the original formulation of the census transform given in Eq. (1) we can only compute a subset of \(2^8 = 256\) of all the 511 structure kernels defined on a 3 × 3 neighborhood. This is because the value of the center pixel of the kernel is fixed to 0 by the choice of the comparison function \(\zeta(\cdot)\) as described above. In order to obtain all the structure kernel we switch the basis of the comparison, so that we also obtain a result for the center pixel. Let \(\mathcal{N}(x)\) be a local spatial neighborhood of the pixel at \(x\) so that \(\mathcal{N}(x) = \mathcal{N}(x) \cup x\). The intensity mean on this neighborhood is denoted by \(\bar{I}(x)\). With this we now reformulate Eq. (1) and write the modified census transform as

\[
\Gamma(x) = \otimes_{y \in \mathcal{N}} \zeta(\bar{I}(x), I(y)). \tag{2}
\]

With this transform we are able to determine all of the 511 structure kernels defined on a 3 × 3 neighborhood. To illustrate the modified transform consider a region of an image with the following pixel intensities in the first column. The structure kernels assigned by the original transform and the modified transform are displayed in column 2 and 3. The transforms are given for the center pixel of the image patch.

From this example we can see that the original census transform will not capture the local image structure correctly in some cases while the modified transform assigns the right kernel.

2.4. Illumination invariance of the structure kernels

Ideally image features for detection and recognition systems should only depend on the structure of the interesting object. In a simple model of image formation the image intensity \(I(x)\) is regarded as the product of object reflectance \(R(x)\) and the illuminance \(L(x)\) at each point \(x=(x,y)\). Additionally the camera influence can be modeled by a gain factor \(g\) and a bias term \(b\), which are assumed to be constant on the image plane. Thus a simple image formation model is

\[
I(x) = gL(x)R(x) + b. \tag{3}
\]

Robust detection systems should only be based on the object structure, which is conveyed by its reflectance properties. But without any knowledge or assumptions on the illumination field \(gL(x)\) determining the object’s reflection field \(R(x)\) is an ill posed problem.

A popular assumption on \(L(x)\) is that it varies only smoothly in \(x\). The proposed local structure features used in this work are also based on the assumption that \(gL(x)\) is spatially smooth. With this assumption we consider the lighting parameters to be constant in a small 3 × 3 neighborhood, \(L(x) = L\).

As the application of constant illumination and gain defines a linear and thus monotonic transformation on the reflectance \(R(x)\). A monotonic transform preserves the order with respect to its arguments and thus does not change the intensity order in the neighborhood. This means that the census transform, which relies on the local intensity order is unaffected. The same applies for the modified census transform. In Fig. 3 an illustrative example is shown. The census transform is visualized as index image where the kernel index determines the pixel intensity.

3. Training and classification

3.1. Sequence of classifiers

The face detector analyzes image patches \(W\) of size 22 × 22 pixel. In the current setup for frontal faces each window has to be classified either as face or background. For a fast detection system this decision should be computable in an efficient manner. The whole classification procedure consists of a sequence of tests performed on the analysis window. The window may be rejected as background after each stage. In the current system we use four such stages like displayed in Fig. 4. The first stage has the lowest complexity (the results of only 20 lookup-table operations have to be accumulated) but is able to reject more than 99% of all windows as background while retaining almost all of the face locations. A similar approach was recently described by Viola [17], but there some thirty stages were necessary and especially the early stages are less
powerful than in this method. Let $H_j(\Gamma)$ be the classifier of the $j$th stage, which classifies the current analysis window, represented by the modified census features $\Gamma$, by

$$H_j(\Gamma) = \sum_{x \in W^j} h_x(\Gamma(x)), \quad (4)$$

where $x$ denotes the location within the analysis window and $W^j \subseteq W$ is the set of pixel-locations with an associated pixel-classifier $h_x$. The pixel-classifier $h_x$ also called elementary classifier consists of a lookup table of length 511, which is the number of the possible kernel indices of the modified census transform. The lookup table holds a weight for each kernel index. A response from an elementary classifier is the weight addressed by the kernel index. A training of a stage classifier is done using a version of the boosting algorithm described in Ref. [6]. In boosting a number of weak classifiers are combined to form a final strong classifier. The goodness of a weak classifier is measured by its error $\epsilon_t$ on the training set. A stage classifier consists of a set of lookup tables $\{h_x : x \in W^j\}$ for the positions $x = (x, y)$ chosen by the algorithm. Each lookup table holds a weight for each kernel index $\gamma$, with $0 \leq \gamma \leq 511$. Table 1 gives the pseudo code for the AdaBoost following the notation in Ref. [6].

One so-called weak classifier $w_x$ for a single pixel position is generated in every boosting round like described in Table 2. For the construction of a weak classifier we first count the kernel index statistics at each position with respect to the boosting weights $D_t(i)$ of the training data. The resulting weighted histograms $g^0_i$ and $g^1_i$ determine whether a single feature should be associated with the face or non-face class. If it is more likely to show up in the face class the weak classifier $w_x$ is assigned 0 at position $\gamma$, else 1. Finally the single feature trained using the Winnow update rule, which is detailed in Section 3.3.

3.2. Training of a stage classifier

The training of a stage classifier is done using a version of the boosting algorithm described in Ref. [6]. In boosting a number of weak classifiers are combined to form a final strong classifier. The goodness of a weak classifier is measured by its error $\epsilon_t$ on the training set. A stage classifier consists of a set of lookup tables $\{h_x : x \in W^j\}$ for the positions $x = (x, y)$ chosen by the algorithm. Each lookup table holds a weight for each kernel index $\gamma$, with $0 \leq \gamma \leq 511$. Table 1 gives the pseudo code for the AdaBoost following the notation in Ref. [6].

One so-called weak classifier $w_x$ for a single pixel position is generated in every boosting round like described in Table 2. For the construction of a weak classifier we first count the kernel index statistics at each position with respect to the boosting weights $D_t(i)$ of the training data. The resulting weighted histograms $g^0_i$ and $g^1_i$ determine whether a single feature should be associated with the face or non-face class. If it is more likely to show up in the face class the weak classifier $w_x$ is assigned 0 at position $\gamma$, else 1. Finally the single feature
Initialize 1500 boosting cycles. The final pixel classifier \( h_x \) of the final strong classifier is based on the final face model:

\[
H_j(x) \leq T_j,
\]

where \( T_j \) is the score threshold at stage \( j \). The score threshold \( T_j \) is tuned so that it maximizes the detection rate on a test database different from the training set.

### 3.3. Training of the last stage

In fourth stage we use a different training algorithm. It produces the same kind of lookup-tables, but it uses all of the given pixel locations. The decision function for this classifier is

\[
H(\Gamma) = H_1(\Gamma) - H_2(\Gamma) = \sum_{x \in \Gamma} h_1^x(\Gamma(x)) - \sum_{x \in \Gamma} h_2^x(\Gamma(x)).
\]

where \( h_1^x \) is the weight for the face class and \( h_2^x \) those of the background-class at location \( x \). The pixel-classifier in this case consists of the difference of the two class-specific weight tables. As the summation is performed in the same domain the pixel-weights can be summed up at training time, \( h_x = h_1^x - h_2^x \).

Two separate weight-tables are only necessary for training as we shall see. The final decision is made by applying a threshold to \( H(\Gamma) \), see Eq. (5), which is determined to achieve a given error or detection rate on a database different from the training set. The pixel classifier \( h_1 \) has the same structure as the boosted pixel classifiers (Fig. 5) described in the last section. The main direction is the fundamentally different training procedure, which leads to a different error distribution. The two sets of weight-tables \( \{h_1^x\} \) and \( \{h_2^x\} \) are trained using an iterative procedure. Initially all weights are set to zero. If a certain weight is addressed for the first time during training it is set to a start-value of 1. The adaptation of the weights is mistake-driven, i.e. if the current pattern is misclassified only the weights addressed by the pattern are changed. The change applies immediately (online update policy). The weights are only updated if the current training pattern is misclassified.

The weight update is done with the Winnov Update Rule [18] which is a multiplicative update rule. There are three training parameter namely a threshold \( T_0 \), a promotion parameter \( \alpha > 1 \) and a demotion parameter \( 0 < \beta < 1 \). For training we fixed the threshold to \( T_0 = 128 \) in all of our experiments. Given \( (\Gamma_1, c_1), \ldots, (\Gamma_m, c_m) \) where \( c_i = 0 \) for \( \Gamma_i \in F \) and \( c_i = 1 \) for \( \Gamma_i \in B \) where \( F \) and \( B \) is the class of faces and non-faces.

**Table 1**

<table>
<thead>
<tr>
<th>Boosting the sparse local structure net</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Given ((\Gamma_1, c_1), \ldots, (\Gamma_m, c_m)) where ( c_i = 0 ) for ( \Gamma_i \in F ) and ( c_i = 1 ) for ( \Gamma_i \in B )</td>
</tr>
<tr>
<td>- Initialize ( D_1(i) = (1/2) b(i/2n) ) for ( c_i = 0, 1 ) where ( n ) is the number of faces and non-faces</td>
</tr>
</tbody>
</table>
| - For \( r = 1, \ldots, T \)
|   Generate weak classifier \( w_i \) at position \( x_i \) with error \( \epsilon_i \) and distribution \( D_r \) as shown in Table 2 |
|   Choose \( a_i = (1/2)(\ln(1 - \epsilon_i) - \epsilon_i) \) |
|   Update the distribution: \( D_{r+1}(i) = D_r(i) \times \left \{ \begin{array}{ll} c^{a_i} & \text{if} \ w_i(\Gamma_i(x)) = c_i \\ c^0 & \text{if} \ w_i(\Gamma_i(x)) \neq c_i \end{array} \right \} \)
|   where \( Z_i \) is chosen so that \( D_{r+1} \) is a distribution |
| - The resulting elementary classifier of a single feature position \( x \) can be obtained by a combination of the appropriate weak classifiers: \( h_x(x) = \sum_{i=1}^{T} a_i w_i(x) \) |
| - The final strong classifier is based on the final face model: \( H(\Gamma) = \sum_{x} h_x(\Gamma(x)) \) |

**Table 2**

<table>
<thead>
<tr>
<th>Training of a weak structure feature classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Generate tables of local weighted kernel indices from faces and non-faces: ( \hat{g}^k(x, y) = \sum_{\Gamma \in \Gamma} D(r)(\Gamma(x) = y)H_i = 0 ), ( \hat{g}^k(x, y) = \sum_{\Gamma \in \Gamma} D(r)(\Gamma(x) = y)H_i = 1 ),</td>
</tr>
<tr>
<td>- Where ( H_i ) is the indicator function that takes 1 if the argument is true and 0 otherwise</td>
</tr>
</tbody>
</table>
| - Calculate error \( \epsilon_i \) for each look-up table \( (x) \): \( \epsilon_i(x) = \min \{ \hat{g}^k(x, y), g^k(x, y) \} \)
| - Select the best position \( x_i \) of loop \( r \):
|   \( x_i = \begin{cases} \hat{g}^k(x, y) = \min \{ \epsilon_i(x) \} & \text{if} |W_r| < n \\ \hat{g}^k(x, y) = \min \{ \epsilon_i(x) \} & \text{else} \end{cases} \)
|   Whereas \( n \) is the maximal number of positions allowed and \( W_r \) is the set of locations already chosen till loop \( r \), thus \( W_{r+1} = \{(x) \cup W_r \} \) and \( W_r = \{\} \)
| - Create look-up table for the weak classifier of loop \( r \) and position \( x_i \):
|   \( w_i(x) = \begin{cases} 0 & \text{if} \ g_i^k(x, y) > g_i^k(x, y) \\ 1 & \text{else} \end{cases} \)

Fig. 5. Illustration of a lookup-table of an elementary classifier \( h_i(x) \).
is the class of faces and $B$ that of non-faces. If $\mathbf{H}(\Gamma) \leq T_0$, which means that the pattern is rejected from the face class but its true label is $c_i=0$ then the involved weights are increased
$$\forall \mathbf{x} \in W, \quad h_i^t(\Gamma_i(\mathbf{x})) \leftarrow \alpha h_i^t(\Gamma_i(\mathbf{x})).$$
If $c_i \neq 0$ and $\mathbf{H}(\Gamma) > T_0$, all involved weights are decreased,
$$\forall \mathbf{x} \in W, \quad h_i^t(\Gamma_i(\mathbf{x})) \leftarrow \beta h_i^t(\Gamma_i(\mathbf{x})).$$
The same applies for the non-face class. The training procedure is repeated iteratively on a training set until no more weight changes occur. The update parameters have been set to $\alpha = 1.01$ and $\beta = 0.99$ for this work.

4. Tracking by means of continuous detection

4.1. Slicing-a temporal scanning procedure

In order to find faces of different size the static detector searches repeatedly downscaled versions of the original grey image. This results in an image pyramid with a constant level ratio $S_p$ between adjacent resolution levels. In this work the level ratio is fixed to $S_p = 1.25$. If we neglect border effects the number of image windows $N_w$ that must be examined by the face detector may be approximated by the number of pixels in the pyramid.

$$N_w \approx W \times H \times \sum_{l=0}^{\infty} S_p^{-2l} = W \times H \times \frac{1}{1-S_p^{-2}},$$

where $W \times H$ is the input image size and $l$ iterates over all pyramid levels. For $S_p = 1.25$ we get a number of $N_w \approx 2.78 \times W \times H$ windows to process.

The main idea now is that it should be sufficient to scan the whole scene in all resolutions for new objects only a few times a second (3–6 in practice). That suffices because in most practical applications it is very unlikely for a face to show up only for fractions of a second in a video stream.

Assume now that we are working at a frame rate $f = 25 \text{s}^{-1}$. In order to scan the whole image five times a second for new faces in all resolutions it suffices to analyze only one fifth of all possible locations of the image pyramid at a time step $k$. This leads to the idea of slicing, where we partition the pyramid into parts of almost equal size. In this work a single part is called a slice. Fig. 6 shows how the image pyramid is partitioned into five different slices (distinguished by color). To formally introduce the partitioning into five slices we can rewrite Eq. (9) like

$$N_w \approx W \times H \left(0.5 + 0.5 + S_p^{-2} + \sum_{l=2}^{\infty} S_p^{-2l} + \sum_{l=4}^{\infty} S_p^{-2l}\right),$$

where each addend corresponds to the pixel fraction of one slice.

If we fix $S_p = 1.25$ the sum is $N_w \approx W \times H \times (0.5 + 0.5 + 0.64 + 0.67 + 0.47) = W \times H \times 2.78$. As each addend represents the size of a slice and all addends are approximately the same size we can see that the pyramid is almost partitioned equally.

The slices are processed sequentially, another slice at the arrival of a new frame. So each slice is processed every five frames. This means if a face is to detect in slice five and currently slice one is processed the maximal time to detection is defined as $T_{\text{detect}} = 5f = 200 \text{ms}$ if $f = 25 \text{s}^{-1}$. In practice this interval is hardly to perceive\footnote{See our demo at http://www.iis.fraunhofer.de/hv/biometrie/} and it gets even smaller if the frame rate increases.

As a tracked face must be updated in each time step a local neighborhood of a tracked face is also considered regardless which slice is currently processed. This means detection is also performed in a local surrounding and neighboring pyramid levels of a tracked face. In practice this leads to smaller speedup than the theoretically achievable factor five. In practical applications we observed a speedup factor from two to three depending on the number of faces in the scene.

4.2. Data association and trajectory filtering

In this work each face is tracked as an independent entity, that means no interdependencies are modeled. For tracking each target face is assigned a state $\mathbf{x}_k \in \mathcal{X}$ vector, which contains position and size as well as the derivatives up to the second derivative at discrete time $k$. $\mathcal{X}$ represents the space of admissible states. A static detection is treated as an observation or measurement, which is used for updating the target-state.

Before we can update a target state we must assign a measurement from the static detector to a tracked target face. Finding the right face from a number of nearby static detections requires an association strategy. Data association in this case addresses the problem of retaining a tracked face’s identity throughout the sequence. In principle this should be solved using a face identification method, which can guarantee that the
face used for update stems from the same person than the tracked target.

In this work we use a simpler method to retain identity of a tracked face based on spatial relationships. During consecutive frames, association is maintained by using a minimum distance criterion where detected faces are associated to the closest target position from the previous frame. This has shown to work quite well in sparsely populated scenes, but it cannot determine a faces identity and thus may fail in crowded scenes.

Once the association is done we update the target-state with the current observation using a Kalman filter. This leads to an instantaneous smoothing of the estimated state sequence and thus gives smooth object trajectories.

Based on an observation sequence from the last $N$ frames we can also decide if the tracked target represents a valid face. This helps to further reduce the false detect rate of the detector.

4.2.1. The state update

In the following we assume to have a persistent measurement for each object, as the static detector delivers exactly one measurement for a distinct face. The case where the object has left the scene or the static detector missed a face is treated in the next section.

The object state $x_k$ contains the spatial position $(x_1, x_2)$ in the image plane as well as eye-distance $e$ which represents the face size in the image plane, as well as their velocities and accelerations,

$$x_k = (x_1, \dot{x}_1, x_2, \dot{x}_2, e, \dot{e})^T. \quad (11)$$

A state $x_k$ is updated if it can be assigned a valid measurement at discrete time $k$. A measurement comes from the set of detections of the static detector in the current image. A measurement is defined by a measurement vector

$$z_k = (x_1, x_2, e)^T. \quad (12)$$

From a set of measurements in a neighborhood of the current state we choose the one that minimizes the following cost function

$$c = ||x^S - x^M|| + ||e^S - e^M||, \quad ||x^S - x^M|| < \Delta x,$$

$$||e^S - e^M|| < \Delta e,$$  

where $x=(x_1, x_2)$ is the position, $^M$ denotes measurements and $^S$ tracking states. The state neighborhood is defined by a spatial update-radius $\Delta x = 20$ pixel and a size update radius $\Delta e = 10$ pixel with respect to the pyramid level where the face size of the target face equals the analysis window size of the static detector.

In order to estimate the current state form measurements we apply a linear Kalman filter. The Kalman filter estimates a time-varying state of a system, which is described by the dynamic equation

$$x(k+1) = Ax(k) + s(k) \quad (14)$$

where $s(k)$ is a zero mean, Gaussian distributed sequence with covariance $S=\delta[ss^T]$. This error also includes the non-foreseeable motion changes of a person in front of the camera. The state transition matrix $A$ has the following structure:

$$A = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \quad (15)$$

The time constant $T$ represents the time between successive video frames. The state $x$ is related to the observation $z$ by the measurement equation

$$z(k) = Hx(k) + r(k), \quad (16)$$

where $r(k)$ is the measurement noise of the static detector which is also assumed to be Gaussian with zero mean and covariance $R=\delta[r^T]$. This includes all position and size inaccuracies of the static detector. The structure of the measurement matrix $H$ is

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \quad (17)$$

We operate the Kalman filter in steady state, i.e. the density propagation has converged. In this case the whole filtering procedure consists of the two steps update and prediction, which are carried out alternately.

The update of the state $x(k)$ at time $k$ is performed by combining the measurement $z(k)$ with the state prediction $\hat{x}(k)$ of the state at $k$ form the state at $k-1$, which is written as

$$x(k) = \hat{x}(k) + K(z(k) - H\hat{x}(k)), \quad (18)$$

where $K$ is the Kalman gain. The Kalman gain is dependent on the two error co variances $S$ and $R$ and is constant for the steady state mode. For details on the Kalman filter the reader is referred to [9]. The prediction is carried out once the state is updated. The state in the next time step $k+1$ is predicted from the current state using the transition equation

$$\hat{x}(k+1) = Ax(k). \quad (19)$$

4.2.2. Space-time-conditions

The dynamic of a person moving in a natural manner in front of a camera is restricted with respect to velocity and acceleration. Therefore, one can assume that the appearance time of a face within the stream is not arbitrarily short. Taking this into account we formulate conditions that a valid face state sequence must fulfill. In order to be able to put restrictions on the space-time behavior of a tracked object we store also $n$ previous state vectors. This state sequence is denoted by $X_k = \{x_k, x_{k-1}, \ldots, x_{k-n+1}\}$. If an object was newly initialized, the state sequence is of length one and it grows at each time step until it reaches the maximal length $n$. Additionally the state vector is augmented by a variable $m$ which indicates if the state was updated by a valid measurement ($m=1$). We now define
two operations on the state sequence. The first one delivers the number of states, which are currently in the sequence, it is denoted by $\mathcal{H}(X)$. The second one returns the number of states, which are updated by a measurement $z$, it is denoted by $U(X)$.

4.2.2.1. Validity. A tracked object is regarded as valid, if two conditions are fulfilled. The length of the state sequence $\mathcal{H}(X)$ must be above a threshold $K$ and the number of predicted states $U(X)$ must be below a threshold $U$. If the validity condition is violated the object is deleted.

4.2.2.2. Initialization. A new state is initialized and added to the tracker from free measurements, i.e. measurements, which could not he used to update an already existing state due to the update radius constraint, see Section 4.2.1. A object remains in initialization as long as $\mathcal{H}(X) \leq K$. In this phase $U(X)$ must be zero else the object is deleted. This rule defines a minimum appearance interval for a valid face object, thus a object gets valid due to the aforementioned conditions if it could be tracked successfully over at least $K$ frames.

4.2.2.3. Extrapolation. If there are no measurements available for a tracked object a time $k$ this may be caused by objects that have left the scene or (in some rare cases) by an event when the static detector missed to detect a face within the image. In order to reduce the sensitivity to detection failures we can extrapolate the system state by using the predicted state Eq. (19) from the previous time step as new state in the current time step substituting Eq. (18) with $\mathbf{x}(k) = \mathbf{\hat{x}}(k)$ if no valid measurement can be found when starting from Eq. (13)

5. Experiments and results

5.1. Data set

The performance of the proposed tracking system is measured using three video sequences showing a different number of persons moving in front of a camera. All se are recorded at $f=25 \text{s}^{-1}$ and are of size $384 \times 288$. The combined sequence length is 73.9 s. Table 3 summarizes the properties of the datasets.

5.2. Results

The static detector has considerable jitter in position and size. This jitter is filtered out by the optimal state estimation of the Kalman filter. Fig. 7 exemplifies this phenomenon from the x-position of a face over a period of 200 frames. The dashed line represents the measurements from the static detector, while the solid line shows the x position from the optimally estimated state. The Kalman filtered state shows almost no jitter in static phases, and no delay when fast movement occurs. However in phases of rapid change the jitter is not attenuated that much, but this is tolerable for fast moving faces, as far as our experience goes.

The static detector produces two kinds of errors, namely to miss a true face position and to detect faces at a non-face position. The first error is represented by the detection rate while the second one is called number of false accepts. The same applies for the proposed tracking system. Table 4 presents the detection capabilities of the static detector compared to the proposed method of this paper on the datasets of Table 3.

One can see that the static detector has a slightly higher detection rate. This is due to the initialization condition formulated in Section 4.2.2 and the maximal time to detection due to slicing, see Section 4.1. These effects may lead to a delay of approximately 280 ms for a visible face to be initialized as valid tracked face. But the proposed tracking extension to the static detector leads to a reduction of the false accept rate by a factor 6, as can be seen from Table 4.

6. Summary and conclusion

We propose a real-time face detection and tracking system for frontal faces. The detection is based on local structure

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Test-sequences for tracking experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>#Frames</td>
</tr>
<tr>
<td>V–I</td>
<td>467</td>
</tr>
<tr>
<td>V–II</td>
<td>603</td>
</tr>
<tr>
<td>V–III</td>
<td>776</td>
</tr>
<tr>
<td>Total</td>
<td>1846</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Detection results for frontal faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>Static detection</td>
</tr>
<tr>
<td></td>
<td>Detection (%)</td>
</tr>
<tr>
<td>V–I</td>
<td>84.5</td>
</tr>
<tr>
<td>V–II</td>
<td>97.4</td>
</tr>
<tr>
<td>V–III</td>
<td>97.7</td>
</tr>
<tr>
<td>Total</td>
<td>94.3</td>
</tr>
</tbody>
</table>

One can see that the detection rates are almost equal while the number of false detects is reduced with the tracking mode.
features, which are computed with the modified census transform both introduced here. For classification we propose a four-stage classifier cascade consisting of simple linear classifiers. Each classifier consists of a set of lookup-tables of feature weights, ranging from 20 in the first stage to all 484 in the last. Though evaluating only 20 features of an analysis window the first stage is able to correctly reject over 99% of all background locations. Together with a coarse-to-fine grid search introduced in our former work this leads to an efficient real-time detector. This ‘pure’ detector is combined with a new ‘tracking by continuous detection approach’ which leads to a new view on the tracking problem, omitting two of the main problems of traditional tracking. The current version is able to track upright human faces which are parallel to the camera in real-time. The number simultaneously tracked faces is not restricted.

We showed that this approach has several advantages over scanning a sequence with the static detector on each frame independently. Though the used static detector is very fast we could show that in video the computational load can be further reduced by the introduction of a temporal scanning strategy for the image pyramid. The detection and tracking capability was demonstrated on sequences with 1846 frames in total where a detection rate of more than 90% was reached. On this database the proposed trajectory smoothing leads to a significant reduction of the false accepts by a factor of 6.

The outcome of the described system is a face detection system that can be operated in video mode and thus can significantly decreasing computation time at high detection rates and very few false positives.

A demo program demonstrating the capability of the proposed method can be downloaded from the internet under http://www.iis.fraunhofer.de/bv/biometrie/.

7. Dedication

We would like to dedicate this paper to Dr Bernhard Fröba, who died in August 2004. He was the father of this research topic in our institute. We think, that it would have been his desire to continue his work and also to write this paper.

References


