Face Tracking using Optical Flow

Development of a Real-Time AdaBoost Cascade Face Tracker[1]

Andreas Ranftl, Fernando Alonso-Fernandez, and Stefan Karlsson

Embedded and Intelligent Systems Research
Halmstad University
Kristian IV:s väg 3, 301 18 Halmstad, Sweden
andran13@student.hh.se, {stefan.karlsson, fernando.alonso-fernandez}@hh.se

Abstract. In this paper a novel face tracking approach is presented where optical flow information is incorporated into the Viola-Jones face detection algorithm. In the original algorithm from Viola and Jones face detection is static as information from previous frames is not considered. In contrast to the Viola-Jones face detector and also to other known dynamic enhancements, the proposed face tracker preserves information about near-positives. The algorithm builds a likelihood map from the intermediate results of the Viola-Jones algorithm which is extrapolated using optical flow. The objects get extracted from the likelihood map using image segmentation techniques. All steps can be computed very efficiently in real-time. The tracker is verified on the Boston Head Tracking Database showing that the proposed algorithm outperforms the standard Viola-Jones face detector.

1 Introduction

Viola and Jones introduced a real-time face detector in 2001 [VJ01]. Face detection is performed by applying a classifier on several windows within the image. The windows vary in location and size in order to determine scale and position of the face rather exactly [VJ01].

The Viola-Jones method has neither a way of incorporating temporal constraints nor combining evidence from previous frames to aid the inference. In short, it is a fully static algorithm. Apart from lacking temporal consistency, the cascade classifier also lacks a way to save information about near-positives. Not only may face positions behave erratically, but if a face becomes temporarily distorted so that the very last part of the cascade fails, the detection fails abruptly. This paper investigates a way to extend the Viola-Jones cascade classifier to achieve a likelihood map that is suited for a form of belief propagation over time. For example, it is possible that a face is detected after several likelihood map refreshes even if it does not pass all stages of the cascaded classifier.

[1] This paper follows a master thesis which was written within the double degree master program in Embedded and Intelligent Systems of Salzburg University of Applied Sciences, Austria and Halmstad University, Sweden.

The real-time optical flow enhanced AdaBoost cascade face tracker aims at calling the Viola-Jones algorithm at every 20th frame. In the frames in between, face detection is done by processing the likelihood map, which is interpolated with optical flow information.

2 Face Detection Performed by the Viola-Jones Algorithm

The algorithm developed by Viola and Jones is based on a cascade of classifiers using Haar-like features, built up in an AdaBoost-based training process by both extracting features from face images and non-face images. The algorithm achieves real-time performance through the cascade structure of the classifiers. Each window constitutes a hypothesis, which gets discarded as soon as a stage is not passed. Each classifier is designed to cancel the evaluation of windows which contain no faces as soon as possible. If a window passes all classification stages it is considered to contain a face [VJ01]. The method of Viola and Jones was improved by Lienhart et al. in 2002 by introducing diagonal Haar-like feature sets [LKP03]. Multi-Block Local Binary Patterns (MB-LBP) are the currently used type of features for classification and they are also used in this work [ZH07]. The Viola-Jones algorithm does not preserve information about near positives. Furthermore, it does not consider previously obtained information.

3 Optical Flow Enhanced AdaBoost Cascade Face Tracker

When detecting faces within an image sequence with the Viola-Jones algorithm every frame is handled separately. This means that the detection process, by shifting different sized windows over the entire image and evaluating them, is done on every single frame. As there is no temporal information taken into consideration, the resulting face bounding boxes appear to be unstable. The boxes slightly change in size and position although the face does not move, and on occasion the tracking is lost altogether for a few intermediate frames. In order to overcome these problems the proposed algorithm works with a likelihood map that saves information about near positives as well as previously computed data.

3.1 Basic Ideas and Flow Chart of the Proposed Face Tracker

In order to track the faces, the algorithm follows the flow chart shown in **Fig. 1**. In the initialization phase the program opens the video input, loads the cascade classifiers and builds the initial likelihood map by utilizing the modified Viola-Jones algorithm. A likelihood map is used due to the fact that it offers the possibility to recognize an undetected face from the Viola-Jones algorithm after several refreshes.
Within the algorithm frame per frame is processed and optical flow computation is done. The current likelihood map gets then warped with the results from optical flow. The employment of optical flow prevents a face from being lost, as long as it has passed a high number of classification stages when establishing or refreshing the likelihood map. Additionally, the size of the tracked face area does not change erratically.

If the frame obtained from the input is a refresh frame, a modified version of the Viola-Jones algorithm is applied to it. There a temporary likelihood map is built which influences the warped likelihood map by recursive filtering. For the purpose of extracting a face from the current frame, a binary likelihood map is created and then segmenting is done by finding edges within the binary image.

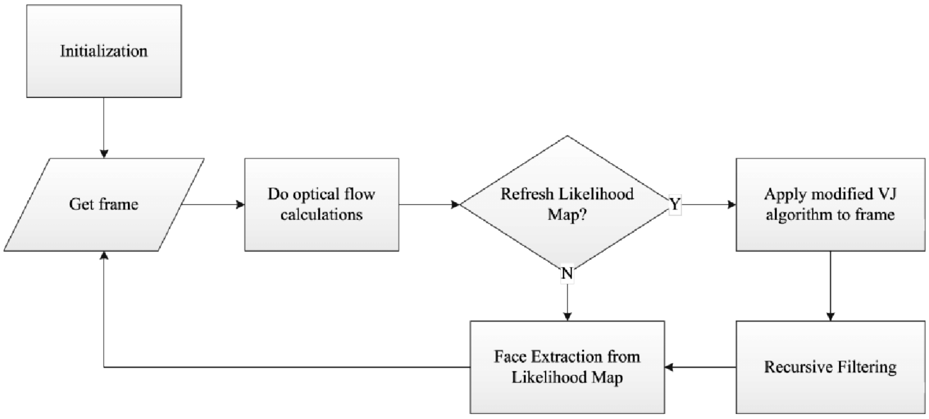
Initialization
Get frame
Do optical flow calculations
Refresh Likelihood Map?
Y
N
Apply modified VJ algorithm to frame
Recursive Filtering
Face Extraction from Likelihood Map

Fig. 1. Flow chart of the developed algorithm.

3.2 Likelihood Map Setup

The likelihood map is built from the intermediate results of the Viola-Jones algorithm. In particular, the number of stages of the classification cascade passed by each detection window is used to form it. This means that the likelihood provides for every pixel a value that indicates the probability of being a face located at the respective position in the original frame.
There are different ways of setting up a single 2-D likelihood map from the Viola-Jones results. It is important to ensure that the possible maximum energy of each detection window scale used in the Viola-Jones algorithm is the same. Otherwise different scales would be weighted differently in the resulting likelihood map.
In order to utilize previously obtained information, a new likelihood map $\mathbf{L_t}$ at time *t* is formed by recursive filtering, see (1). $\mathbf{L}$ describes the current likelihood map obtained from the Viola-Jones algorithm and $\mathbf{L_i}$ is the interpolated likelihood map at same time *t*.

$$\mathbf{L_t} = (1 - \alpha) * \mathbf{L} + \alpha * \mathbf{L_i} \tag{1}$$

The computation time of building the likelihood map can be lowered by taking advantage of a reject level threshold. This means that only windows, which pass a certain number of stages of the cascaded classifier, contribute to the likelihood map.

Window Center Orientated Likelihood Map
One way of building a likelihood map is to add the result of each detection window to the pixel in the likelihood map corresponding to the center pixel of the respective

window, see **Fig. 2** for a 3 * 3 pixels example window. By doing so, information about the size of the face is lost. The likelihood value for an odd sized window is calculated by multiplying the number of passed stages by 4. Otherwise the influence on the likelihood map of even and odd sized windows would not be equal.
As a window with even sized height and width has no single center pixel, the 2 * 2 pixels center is set to the rejection level value, see **Fig. 2** for an example.

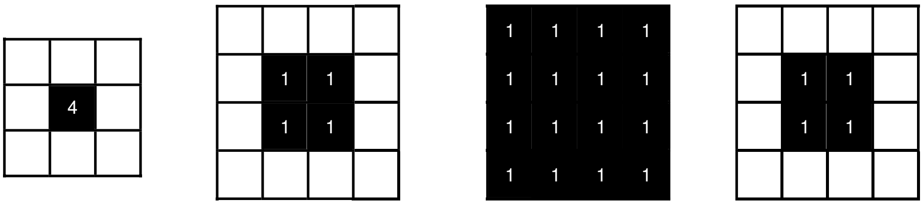

Fig. 2. Principle of setting the likelihood value for an odd sized window (left) and for an even sized window (middle left) when using the window center orientated likelihood map approach. The windows on the right are examples for the area orientated approach. Every pixel of the likelihood map corresponding to a detection window is set to the number of passed stages (middle right). The very right example uses a shrinking factor of 0.5. Values represent the factor with which the number of passed stages of the respective detection window is multiplied.

As previously mentioned, every detection window is weighted the same in the likelihood map. This requires us to compensate the step size caused by a scaling factor used in the detection process. The compensation of the detection window scale is done by weighting the counter of passed stages n_{levels} with the respective window width w_{window}, see (2) for an odd sized window.

$$L_n = L_{n-1} + n_{levels} * 4 * w_{window} \qquad (2)$$

Window Area Orientated Likelihood Map
Another approach for building a likelihood map is to set every pixel of a detection window to the corresponding Viola-Jones result and then adding all the windows up, see **Fig. 2**. By doing so, we do not have to care about even or odd sized windows and we do not have to care about weighting of results of different window sizes. Also the information about the size of the face is preserved. It could be shown that there is nearly no difference in computation time when using a reject level threshold of 15. Compared to the original Viola-Jones implementation the likelihood map building methods need 10% more computation time.
When setting the respective pixels in the likelihood map to the correct value, a shrinking factor was introduced. This shrinking factor shrinks the area which should be set to the Viola-Jones result. The principle of applying a shrinking factor is based on the fact that detection windows are bigger than the faces outlined by them. When interpolating the likelihood map with the flow map there are only motion vectors for the pixels of the face. Therefore setting the whole window area to the respective Viola-Jones result would lead to an inaccurate likelihood map after interpolating with optical flow results.

3.3 Face Extraction from the Likelihood Map

In order to differentiate between face-containing and non-face-containing regions of the image a threshold is applied to the likelihood map. Note that the probabilities which are represented by the pixel intensities of the likelihood map can vary even for the same face region, depending on face detections within neighboring windows and windows with different sizes that detect the same face.

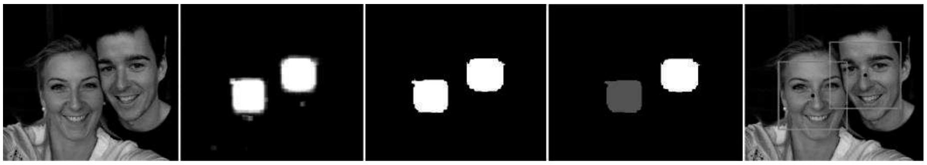

Fig. 3. The left image is the original frame. The middle left image represents the generated likelihood map. Different probabilities are shown by different gray shades. The high values for the two faces can be observed clearly in the likelihood map. The image in the middle is the binary likelihood map which is the result of applying a threshold of 65 to the original likelihood map. The middle right image illustrates the segmented likelihood map and the very right image shows the computed bounding boxes of the face tracker marked in the original frame.

In order to label the face regions uniquely and to visualize the outcome, edge detection is performed on the binary likelihood map. As the contours of the face regions are known, the segmentation algorithm searches for the outer points. Each face region in the likelihood map equals roughly a rectangle. The outer points are stretched by the inverse of the shrinking factor s which was used for downscaling face-containing windows when putting them into the likelihood map. By shrinking the windows with a factor that is small enough, the face-containing regions do not overlap in the likelihood map even when the faces are near to each other. The resizing is done by applying equations (3) and (4) where x_{face} and y_{face} represent the coordinates of the upper left corner of the bounding box which is used to mark a face. The variables $width_{face}$ and $height_{face}$ are the differences of the max and min values of x and y respectively.

$$x_{face} = x_{min} + \frac{width_{face}}{2} - \frac{width_{face}}{2*s} = x_{min} - width_{face} * \frac{1-s}{2*s} \tag{3}$$

$$y_{face} = y_{min} + \frac{height_{face}}{2} - \frac{height_{face}}{2*s} = y_{min} - height_{face} * \frac{1-s}{2*s} \tag{4}$$

4 Tracking During Occlusion

Under certain circumstances the developed face tracking algorithm is able to track faces during partial and complete occlusion. Given that the occlusion is encompassed fast enough (e.g. a passing car) we can take advantage of the fact that the utilized Farnebäck dense flow technique is invariant to very fast motion (see **Fig. 4**).

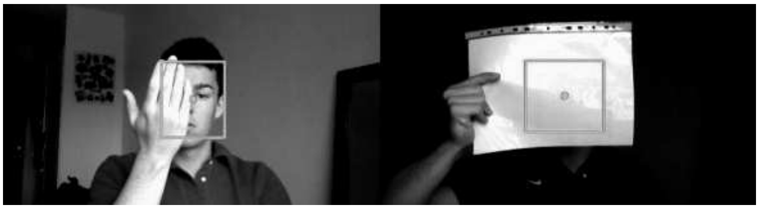

Fig. 4. The optical flow face tracker is able to track faces under partial occlusion (left) and also under complete occlusion (right).

5 Results

The proposed face tracking algorithm was implemented in C++ utilizing the OpenCV library in version 2.4.8.0 with activated parallelization. The face tracker was evaluated on the Boston Head Tracking Database [LSA00]. We used the 45 videos which were recorded under uniform light conditions. Ground truth information was made available through the UVAEYES annotations which represent the positions of the eyes (see **Fig. 5** as example) [VG09].
The method with which the presented approach is compared is the original Viola-Jones face detector. For terms of comparison, the Viola-Jones method performs face detection on every single frame. Within the developed optical flow face tracker, the reject level threshold for accepting windows to contribute to the likelihood map was set to 15. The shrinking factor was set to 1/3.

5.1 Measurement of Detection Rate and Accuracy

In order to measure the accuracy of the tracking algorithm the Euclidean distance of the computed center point to the real center point of the face is calculated. The real center point $[x, y]^T$ is computed by utilizing (5) and (6), which calculate the middle point between the eyes (x_1, y_1 and x_2, y_2 respectively) and add 10 pixels in y direction (origin is upper left corner of the image) for setting the coordinates of the face center. This works quite well as there is not much difference in the sizes of faces.

$$x = x_1 + \frac{x_2 - x_1}{2} \tag{5}$$

$$y = y_1 + \frac{y_2 - y_1}{2} + 10 \tag{6}$$

The evaluated algorithms output a bounding box. The center of this bounding box is set as the face center point. If the Euclidean distance of a center point computed by an algorithm to the real center point is higher than 20 pixels, the detection is classified as a false positive. If there is no detection, it is counted as a false negative.

If there are multiple detections, then all the distances are measured and the nearest one is chosen as valid attempt as long as it is within the threshold defined above. The other false detections cause the number of false positives to increase. If all detections are outside the threshold circle, they are all added to the false positives count.

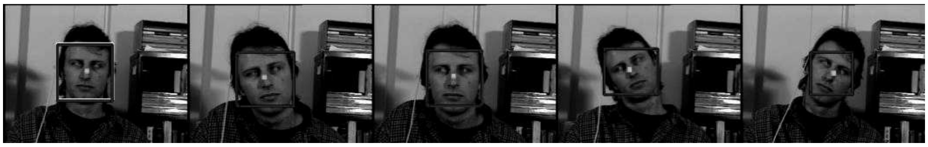

Fig. 5. Example frames (23, 92, 100, 150 and 188) of the video jam1.avi from the Boston Head Tracking Database. The red dots indicate the ground truth positions of the eyes. The green dots represent the computed face centers. The blue rectangles represent the face bounding boxes returned by the optical flow face tracker. In frame 23 a refresh is done as also the Viola-Jones bounding box (white) is visible.

5.2 Detection rate

The detection rate *r* is calculated by summing up the valid detections *d* within a video and dividing the obtained sum by the number of frames *nframes* (see (7)).

$$r = \frac{1}{n_{frames}} * \sum_{i=1}^{n_{frames}} d_i \tag{7}$$

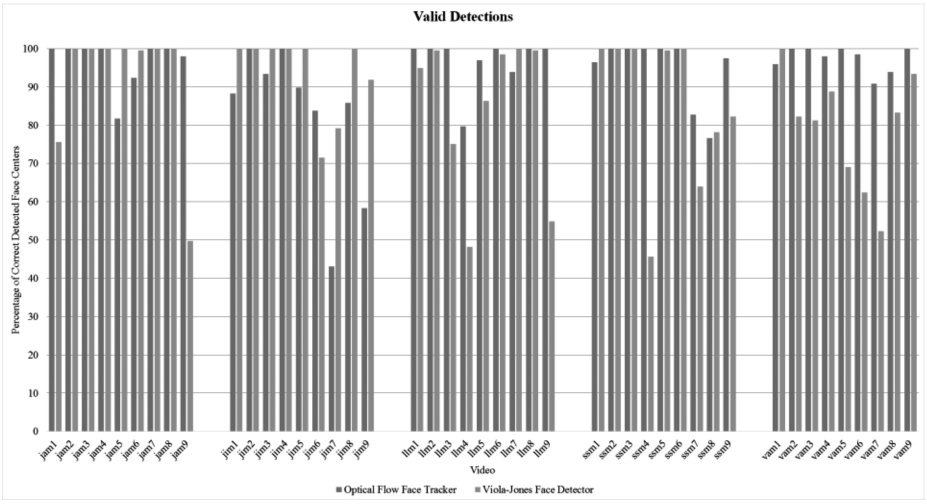

Fig. 6. Illustration of percentage of correctly detected face centers per video.

Taking all videos of the database into consideration the optical flow face tracker shows a better detection rate. The average detection rate of all videos is 79.15% for the optical flow face tracker and 73.71% for the Viola-Jones face detector. **Fig. 6** shows the average detection rate per video of the Boston Head Tracking Database.

5.3 Accuracy and Robustness

When measuring the average accuracy of valid detections we observed that the Viola-Jones algorithm (2.56 pixels offset) is on average in all videos more accurate than the developed face tracker (5.99 pixels offset). It should be considered, however, that the

Viola-Jones method loses the face in several frames. These frames are not considered in the computation of the average accuracy of the Viola-Jones algorithm. The average offset in pixels is illustrated in **Fig. 7**.

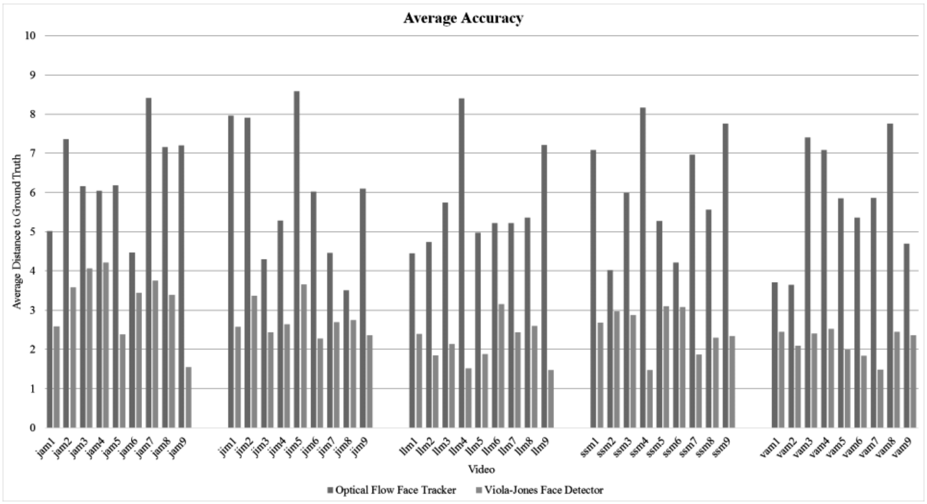

Fig. 7. Average inaccuracy of the optical flow face tracker and the Viola-Jones algorithm.

On average the Viola-Jones algorithm outputs a higher amount of false detections. In particular, the Viola-Jones algorithm returned on average 26 false detections per video and the optical flow face tracker 18 false detections. This is caused by certain head poses that make it impossible for Viola-Jones to detect the face due to the utilized classifier which was trained with upright face images. Therefore these false detections are mostly false negatives.

By design the optical flow face tracker will always detect a face as long as it was initialized with one. If the Viola-Jones algorithm does not detect a face in the refreshment frame it is re-executed until it detects one. If Viola-Jones produces a false negative no refresh on the likelihood map is done, which causes the optical flow face tracker to become inaccurate and produce false positives with increasing time. If Viola-Jones outputs a false positive, the error is propagated as this false detection is followed by the face tracker. By increasing the threshold for segmentation of the likelihood map, the false positive rate can be lowered. However, it is possible that the accuracy gets worse. The Viola-Jones detector returned only a few false positives.

5.4 Stability

Erratic movements within the face tracking process are unpleasing as faces are only moving relatively slow in practice. The stability is measured by taking the Euclidean distance e between the face centers of two successive frames with valid detections (see (8)). The distance *e* is a direct measure for the instability. Center points are considered as valid if their distance to ground truth is less than 20 pixels.

$$e = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \quad (8)$$

In general the optical flow face tracker shows less erratic movements. The average on all videos equals 0.9 pixel for the optical flow face tracker and 1.5 pixels for the Viola-Jones face detector. **Fig. 8** shows the average instability per video in pixels.

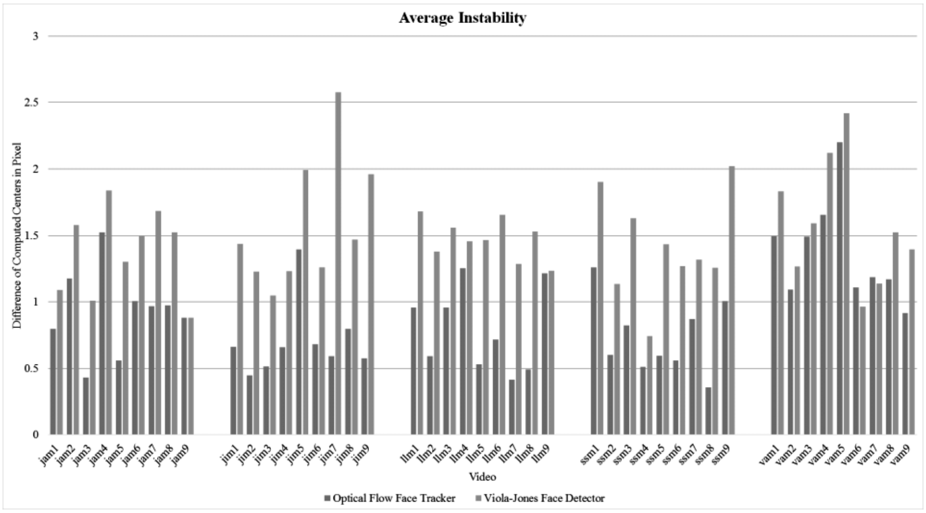

Fig. 8. Average instability of both algorithms applied to all videos of the database.

5.5 Speed

The speed of the algorithms is measured in computation time in milliseconds. Compared to the Optical Flow Face Tracker, the Viola-Jones algorithm needs less time for computation. It has to be outlined that the average CPU usage is higher when executing the Viola-Jones algorithm. This is caused by the high parallelization of the OpenCV implementation of the Viola-Jones method. In contrast, the computation of the Farnebäck optical flow is mostly done by one core which causes the average CPU usage to be low, but the computation time to be high. The optical flow computation takes a majority of the computation time of the face tracking algorithm (in average 11.33 milliseconds). The rest of the computation time of the algorithm is much lower (1.67 milliseconds) compared to the Viola-Jones algorithm (6 milliseconds).
The Viola-Jones algorithm and our algorithm have the same complexity due to being the Viola-Jones algorithm a part of the proposed system. However, because the Viola-Jones algorithm is called not at every frame, one can expect a constant factor of speed increase. This speed increase is given by how often the likelihood map should be updated.

6 Conclusion

We presented a novel real-time face tracker which utilizes a modified version of the Viola-Jones algorithm for face detection. In contrast to a pure Viola-Jones face detector the developed approach calls a modified Viola-Jones method only at every 20th frame for refreshing a likelihood map. The likelihood values within this map are dependent on the numbers of classification stages which detection windows pass when the classification is done.

In order to track the faces the likelihood map is interpolated with a flow map computed by the Farnebäck dense optical flow method. The resulting likelihood map of the modified Viola-Jones algorithm contributes to the system's likelihood map by recursive filtering.

Compared to the original Viola-Jones implementation, the likelihood map approach enables faces to be detected even when they do not pass all of the stages of the cascade classifier. Due to the fact that the likelihood map is never discarded completely, a region gets also a high value in the likelihood map if the respective window passes for example 17, 18 or 19 stages within several executions of the modified Viola-Jones method. Another advantage of the developed face tracker is that it can also track faces under partial and complete occlusion.

The developed face tracking algorithm and the original Viola-Jones face detector were evaluated on the Boston Head Tracking Database. The developed face tracker achieved a higher detection rate than the Viola-Jones face detector. Furthermore, the optical flow face tracker showed less erratic movements of detections.

The developed tracker relies on the Viola-Jones algorithm which means that an error of Viola-Jones algorithm during initialization or refresh of the likelihood map gets propagated. By utilizing also other methods for face detection this effect could be minimized (e.g. execute a face detection method with a low computation time only on the extracted face areas in order to check if the tracker has lost the faces or not).

Bibliography

[LKP03] Lienhart, R.; Kuranov A., Pisarevsky V.: Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. In DAGM 25th Pattern Recognition Symposium, 2003.

[LSA00] La Cascia M., Sclaroff S., Athitsos V.: Fast, Reliable Head Tracking under Varying Illumination: An Approach Based on Registration of Texture-Mapped 3D Models. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000.

[VG09] Valenti R., Gevers T.: Robustifying Eye Center Localization by Head Pose Cues. In IEEE conference on Computer Vision and Pattern Recognition, 2009.

[VJ01] Viola P., Jones M.: Rapid Object Detection using a Boosted Cascade of Simple Features. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.

[ZH07] Zhang, L. et al.: Face Detection Based on Multi-Block LBP Representation. In ICB International Conference, 2007.