IMPROVED FACE TRACKING THANKS TO LOCAL FEATURES CORRESPONDENCE

Alberto Piacenza, Fabrizio Guerrini, Nicola Adami, Riccardo Leonardi

Dpt. of Information Engineering, Signal Communication Laboratory, University of Brescia, Italy

ABSTRACT

In this paper, we propose a technique to enhance the quality of detected face tracks in videos. In particular, we present a tracking algorithm that can improve the temporal localization of the tracks, remedying to the unavoidable failures of the face detection algorithms. Local features are extracted and tracked to “fill the gaps” left by missed detections. The principal aim of this work is to provide robust and well localized tracks of faces to a system of Interactive Movietelling, but the concepts can be extended whenever there is the necessity to localize the presence of a determined face even in environments where the face detection is, for any reason, difficult. We test the effectiveness of the proposed algorithm in terms of faces localization both in space and time, first assessing the performance in an ad-hoc simulation scenario and then showing output examples of some real-world video sequences.

Index Terms**—** Face Tracking, Feature Extraction, Interactive Storytelling.

1. INTRODUCTION

This paper presents a technique to enhance face tracking as obtained through secondary tracking of local features extracted only in the detected face bounding box.

Face detection, tracking and recognition in video content are applied for a variety of purposes in many different applications, ranging from information retrieval to multimedia security [1] [2] [3] and it is usually considered a special instance of the yet more general case of object-based applications. The modification of the general aspect of the face in different portions of the video due to e.g. disguise is an important problem when model-based tracking is employed, i.e. when a particular face is searched in the video content, and in particular in those applications aiming to recognize the particular person, e.g. naming the same actor across different movies [4]. A more general class of problems involve challenges in the face detection stage such as momentary occlusions, pose and illumination changes and so on. When detecting and then tracking a face, these problems leave ”gaps“ in the track in those frames where the detection has not been successful. This has two undesirable effects: unnecessarily duplicating the recognition process in the split tracks and failing to identify the presence of the character in-between true detections. The objective of the technique presented in this paper is to tackle this particular problem.

The intended application for the enhanced face tracking is our Interactive Movietelling system [5], specifically to allow automatic character recognition in movies. The objective is to identify as precisely as possible those frames in which a certain character is present to provide a semantic description of the video content. The goal is similar to those found in other applications as well, such as video summarization [6], but in our case there is a narrative being presented that requires high descriptive precision. Instead of simply letting the author responsible for setting up the system to manually parse the video content and label each shot, the description process is sped up using automatic tools: in particular, face detection and tracking presently implemented in OpenCV are first employed to obtain raw face tracks. Face recognition is performed on each raw track to cluster them and subsequently the author can refine and possibly correct the resulting character attributes using a very simple user interface.

However, the poor results of these off-the-shelf processes cause a significant work overhead for the author. In particular, problems with raw face tracks include imprecise or missed detection and face bounding box drifting. Therefore, tracking by detection is not suitable and it is necessary to “fill in the gaps” of the raw tracks to help the task of the author or to render the system fully automatic altogether. When the face tracks are consolidated through our enhancement process, face recognition could then be applied to those frames in which the face has been robustly detected.

Even if the use of additional data such as the movie script, audio cues such as speaking parts and so on could and will help identify the characters in a given segment of the movie, such as in [4], the present work is intended to provide a solid foundation based on the visual data alone to be enriched in a subsequent stage. The good results obtained, however, encourage fostering this methodology in other applications as well. Therefore, despite being a solution studied specifically for the Interactive Movietelling system, we believe that the face tracks enhancing technique presented in this paper could be useful for all those works using face tracks as input for more complex tasks too such as emotion detection.

This work has been funded (in part) by the European Commission under grant agreement IRIS (FP7-ICT-231824).

The rest of the paper is organized as follows. Section 2 details the face tracks enhancing technology. Section 3 reports the experimental results conducted on Michael Radford's screen adaptation of Shakespeare's play *The Merchant of Venice* [7] and Section 4 concludes the paper.

2. FACE TRACKING SYSTEM DESCRIPTION

The proposed method is depicted in the diagram of Figure 1. Two main processing stages can be identified: the first stage derives from the work presented in [8] and provides the so called "raw face tracks" (Figure 1 top left). Then, in the second stage, the temporal localization of these raw tracks are improved by the introduction of a new face tracker that produces the final enhanced face tracks (Figure 1 bottom left). The input of the system is a single movie shot, and the entire process should be repeated for all the considered shots: to perform character recognition, the tracks should be first clustered. The focus of the paper is however on the operations performed on a given single shot.

First, a frontal face detector is run on each frame of the considered video shot. For this task we decided to use the Viola-Jones face detector [3] that achieves a high detection accuracy. Then, the detected faces corresponding to the same character (present in the script or not) have to be somehow connected in order to obtain the face tracks. When obtaining the raw face tracks for a single shot, face recognition is not required since the motion information alone is sufficient to identify matching faces across different frames. Therefore, the Kanade-Lucas-Tomasi (KLT) pyramidal tracker [9, 10] is applied. Given a video shot, its task is to create tracks of "interest points", roughly explained away as the strongest corners. These corners are extracted in the first frame, then they are tracked by the pyramidal implementation of the iterative optical flow proposed in [11] that provides robustness to large motion, while the original implementation of Lucas-Kanade could guarantee just local tracking accuracy. Those points which cannot reliably be propagated from one frame to the next are discarded and replaced by new points. The output of this operation is a set of point tracks starting at some frame in the shot and continuing until some later frame.

After the two sub-stages (face detection and KLT tracking) have been computed, a set of faces and a set of point tracks are available. The last step of the first processing stage is to use the information contained in the point tracks to merge faces that belong to the same character: a standard agglomerative clustering method is applied based on a confidence measure that counts the number of point tracks that intersect the faces, i.e., that has a point within the face bounding box in the corresponding frame. For a given pair of faces A and B, in different frames, three classes of point tracks can be defined: tracks that intersect both A and B, A but not B or B but not A. The confidence measure $c\left(A, B\right)$ is thus defined as the ratio of the number of point tracks that intersect both faces with respect to those that intersect only one face:

$$c\left(A, B\right) = \frac{|A \cap B|}{|A| + |B|}$$

At every iteration, the cluster algorithm merges the two most similar clusters. The confidence measure between clusters C is defined as the maximum c for every pair of faces in the clusters. If two clusters contain faces within the same frame, C is put to 0 in order to prevent them from merging. The clustering process stops when a minimum confidence threshold is reached, in this case set to 0.5.

This tracking procedure is extremely robust w.r.t. methods based on some face-specific or general appearance features because it can match faces that have not been continuously detected due to pose variation or change of expression and moreover it does not suffer from "drift" of the appearance model kept by the tracker of one object into another in the video. Unfortunately, this approach has also a main drawback that can often compromise an accurate temporal localization of the face tracks. In fact, this method heavily relies on the performance of the face detector, since it creates tracks just for detected faces. Although the face detection algorithms are quite efficient, sometimes faces that are not completely frontal, are in poor illumination conditions, or are even under situation of bad camera focus, cannot be reliably detected. For this reason, the faces in those frames where one of these problems arises cannot be localized and therefore are not included in the face tracks. This problem is well highlighted in the track depicted in Figure 1 (top right). In this example, the face detection process sometimes misses depending on the head pose. Therefore, it is necessary to include the missed faces into the track and thus fill the gaps.

For this purpose, we introduce the second processing stage to connect the detected faces within a face track by implementing a tracking algorithm that can extend the bounding boxes of the previously detected faces also in those frames where the detection has not been successful. Furthermore, it is necessary to identify when a face occlusion has occurred: in these cases the tracker has to disconnect from his target, waiting for another detection. The key idea of this stage is to reapply the KLT pyramidal tracker, however no longer initializing the interesting points within the whole frame, but just within the bounding boxes of the detected faces belonging to a given face track.

First, let's assume that in the frame i there is a detected face which is missed in the frame $i + 1$. In the face bounding box in the frame i, we extract the interest points and track them using the pyramidal tracking in the frame $i + 1$. Then, the bounding box in the frame $i + 1$ is defined employing the RANSAC algorithm [12] to estimate the model of the transformation occurred between the two sets of points and then applying the same transformation to the corners of the bounding box of the frame i. One of the main advantages of using RANSAC is that the algorithm is able to find a good trans-

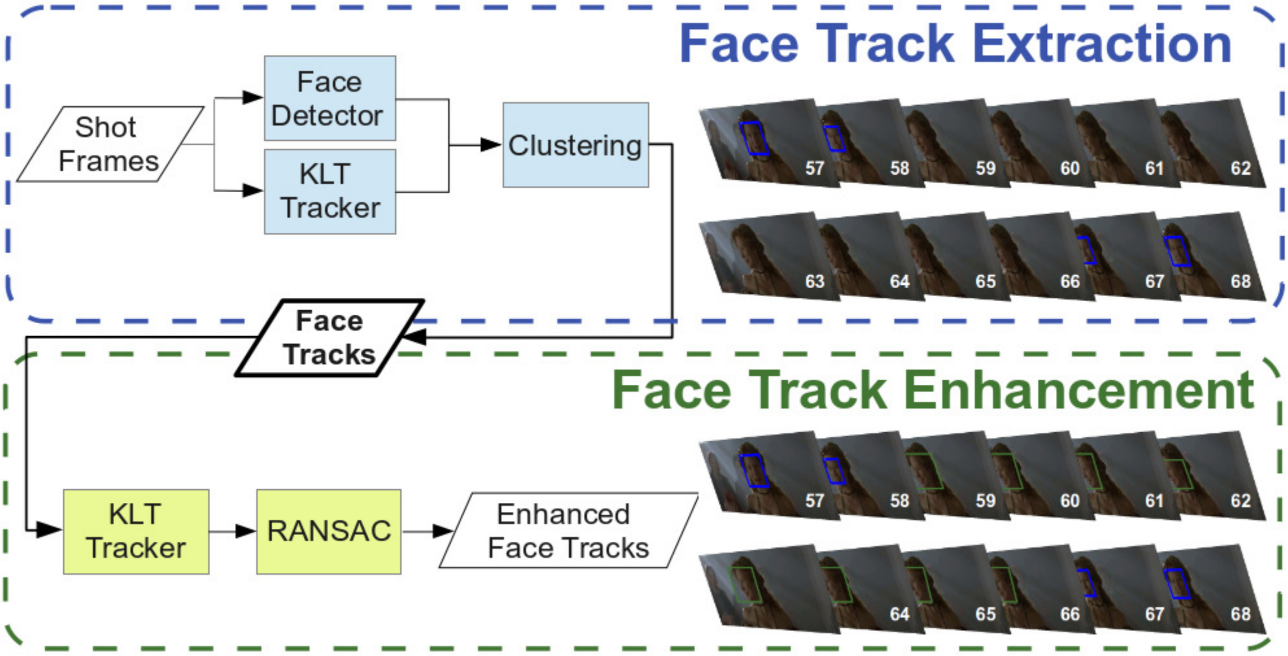

Fig. 1. (left) A flowchart of the operations involved in the creation of the enhanced face tracks. The last row is the focus of this paper. (right) Depiction of the output tracks: the frames of a small excerpt of one shot are presented. The blue rectangles in the top represent the detected faces that have been correctly identified in a given face track, but the face detection has failed to find the face in the in-between frames. The application of the face tracks enhancement allow to find the face in those frames, as highlighted by the green rectangles in the bottom track.

formation model even in presence of a high ratio of outliers, which are interest points extracted from the background because sometimes the bounding box around a detected face is not perfectly adherent to the face.

In case of a partial or complete occlusion of the face, three possibilities arise: the pyramidal tracker cannot reliably track the points, RANSAC cannot find a plausible transformation model or RANSAC finds a model using a low number of inliers. In all of these cases, the target is declared lost and the bounding box for that frame is not defined.

Since we are working offline, the frame correspondence $i \rightarrow i+1$ is not necessarily in the forward direction. In fact, the entire process is also run backwards, which is useful to extend the track in the past when the face is first detected in a later frame. Thus, it is possible that in some frames two bounding boxes are available for a particular face, coming from both the forward and backward tracking. In this case, the tie is resolved using the bounding box derived from the nearest frame with a detected face.

3. EXPERIMENTAL RESULTS

The face tracking system described here has been entirely implemented in the C++ language, exploiting the potentiality of the OpenCV2.4 libraries [13]. In particular, we use the following well-tested key functions: *detectMultiscale* using the standard *haar_cascade_frontalface.xml* for the face detection; *goodFeaturesToTrack* for the detection of the interesting points; *calcOpticalFlowPyrLK* for the pyramidal KLT tracking of points; and *find_Homography(CV_RANSAC)* for the implementation of the RANSAC algorithm.

For evaluating the performance of our face tracking system, we run the tracker on the YouTube Faces Database presented in [14]. This database contains a set of video clips downloaded from YouTube with duration ranging from 48 frames to 6070 frames, with an average length of 181. In particular, we focus our evaluation on the performance of the second stage of our system. To provide ourself with a useful ground-truth for this database, we re-build the face ground-truth running the the first stage of Figure 1 and discarding those clips in which there is not a face track detected in a continuous fashion on the whole duration. Taking this as our ground-truth, we evaluate the ability of recovering the faces bounding boxes when the face detection fails by obscuring the detection result for a certain percentage of the frames. For example, in a clip of 200 frames, if we introduce detection misses for the 20% of the frames we do not consider the output of the first stage for 40 frames. In this way, we simulate the failure of the face detector and thus we are able to give a systematic measure of the performance of the enhanced face

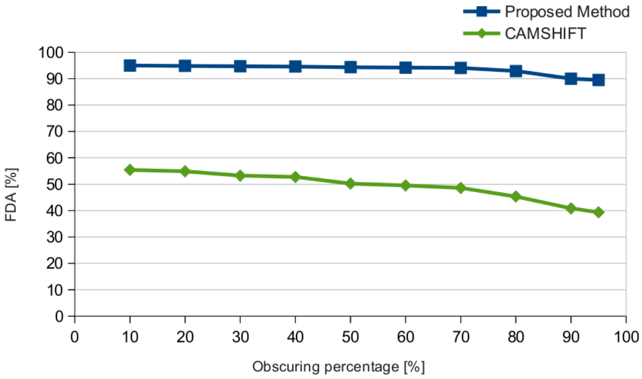

Fig. 2. Comparison of the FDA percentages obtained by our system and the CAMSHIFT algorithm, in relation to the percentage of forced misses in the face detection process.

tracking stage.

For a given frame t, the *Frame Detection Accuracy (FDA)* measure calculates the spatial overlap between a pair of ground-truth and system output objects (in our case, the faces bounding boxes) as a ratio of the spatial intersection between the two objects and the spatial union of them. The sum of all of the overlaps is normalized over the average between the number of ground-truth objects $N_G^{(t)}$ and the number of detected objects $N_D^{(t)}$ [15]. Thus, we define $FDA\,(t)$ as:

$$FDA\,(t) = \frac{Overlap\_Ratio}{\frac{N_G^{(t)}+N_D^{(t)}}{2}}$$

$$Overlap\_Ratio = \sum_{i=1}^{N} \frac{\left|G_i^{(t)} \cap D_i^{(t)}\right|}{\left|G_i^{(t)} \cup D_i^{(t)}\right|} \tag{1}$$

computed on the N objects for which the intersection is not zero. In Eq. 1, $G_i^{(t)}$ denotes the i-th ground-truth object in the frame t and $D_i^{(t)}$ denotes the i-th detected object in the same frame. The FDA for a particular shot is the temporal average of $FDA\,(t)$ computed on all its frames.

The chart in Figure 2 presents the FDA performance of our tracking system compared with those provided by the CAMSHIFT [16] algorithm. CAMSHIFT stands for “Continuously Adaptive Mean Shift” and it is considered the basis for the face-tracking algorithm in OpenCV. The core of CAMSHIFT is identified by the combination of the basic Mean Shift algorithm with an adaptive region-sizing step. The mean shift algorithm operates on probability distributions, so to track objects in video frame sequences the color image data has to be represented as a probability distribution. In general this is accomplished using color histograms.

Analyzing the results presented in Figure 2, it clearly appears that the performance provided by our system are quite preferable than those given by the CAMSHIFT algorithm. This is mainly due to the fact that when CAMSHIFT calculates the target histograms based on the bounding box of the detected faces, these latter often result perturbed by the background color. For this reason, in some cases the bounding boxes computed by CAMSHIFT result well centered around the face to be tracked, but they are larger than necessary simply because the algorithm tries to include in the tracking process portions of background. This problem can at best just marginally influence the behavior of our system thanks to the application of RANSAC, that in most cases discard the background information giving a more stable and precise localization of the faces.

Misses [%]	10	20	30	40	50	60	70	80	90	95
FDA [%]	95	95	95	95	94	94	94	93	90	89
TO [%]	100	98	99	99	97	97	97	96	82	80

Table 1. Performance measure expressed in terms of FDA and TO relative to the misses percentage on the YouTube database.

Another indicator of the performance of the tracker is the *Temporal Overlap (TO)*. The *TO* is the length of the intersection between the ground-truth track and the system output track, divided by the length of the ground-truth track:

$$TO = \frac{Length\,(G_i \cap D_i)}{Length\,(G_i)}$$

where the operator $Length(\cdot)$ returns the number of frames. The results are presented in Table 1. The TO shows that the faces are generally well tracked even w.r.t. their time localization. The reason why the faces are sometimes lost during the tracking process is that in those cases the detected faces are too small or blurred to allow the extraction of a good number of interest point inside the corresponding bounding box.

Some example clips showing the effectiveness of our algorithm can be found at [17]. The results of the proposed enhanced face tracking and CAMSHIFT are shown on a small selection from the database. In particular, in these examples it can be easily observed the influence of the background in the tracking process. Moreover, we also report some results of tracking on some shots of the baseline movie [7]. Here, there is no need to simulate face detection misses which instead actually occur when applying the Viola-Jones detector. In most cases, such misses are effectively recovered by our system,

4. CONCLUSIONS

In this paper, we presented a technique to enhance the quality of detected face tracks in videos. While the work is framed in the video processing stage of our application of Interactive Movietelling for automatic semantic description of the video content, the concepts introduced are completely general and can be applied in any application where the detection of face tracks with reliable time and space localization is desirable. The key idea is to combine the face tracks detection stage with a new tracking algorithm that is capable of extending the bounding box of the detected faces even in those video frames where the face detection is not successful.

We implemented all the processing stage in the `C++` language exploiting the potentiality of the OpenCV2.4 libraries and tested the effectiveness of the tracking algorithm in terms of localization of faces both in space and time, comparing its results with those achieved by the CAMSHIFT algorithm. Furthermore we presented experiments on real-world video sequences showing how the face tracks consistently improve.

5. REFERENCES

[1] Stan Z. Li and Anil K. Jain, Eds., *Handbook of Face Recognition, 2nd Edition*, Springer, 2011.

[2] Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 34–58, Jan. 2002.

[3] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[4] J. Sang and C. Xu, "Robust face-name graph matching for movie character identification," *IEEE Transactions on Multimedia*, vol. 14, no. 3-1, pp. 586–596, 2012.

[5] Alberto Piacenza, Fabrizio Guerrini, Nicola Adami, Riccardo Leonardi, Julie Porteous, Jonathan Teutenberg, and Marc Cavazza, "Generating story variants with constrained video recombination," in *Proceedings of the 19th ACM international conference on Multimedia*, New York, NY, USA, 2011, MM '11, pp. 223–232, ACM.

[6] L. Ballan, M. Bertini, A. Del Bimbo, and W. Nunziati, "Soccer players identification based on visual local features," in *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, 2007, pp. 258–265.

[7] M. Radford, "The Merchant of Venice (film adaptation)," MGM Home Ent. (Europe) Ltd., 2004.

[8] M. Everingham, J. Sivic, and A. Zisserman, "Taking the bite out of automated naming of characters in tv video," *Image Vision Comput.*, vol. 27, no. 5, pp. 545–559, 2009.

[9] J. Shi and C. Tomasi, "Good features to track," in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, 1994, pp. 593 – 600.

[10] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, April 1981, pp. 674–679.

[11] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[12] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[13] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[14] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *in Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2011.

[15] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 319–336, Feb. 2009.

[16] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, vol. 2, no. 2, pp. 12–21, 1998.

[17] *www.ing.unibs.it/alberto.piacenza/TrackWithPoints.*