Online Multi-Face Detection and Tracking using Detector Confidence and Structured SVMs

Francesco Comaschi[1], Sander Stuijk[1], Twan Basten[1,2] and Henk Corporaal[1]

[1]Eindhoven University of Technology, The Netherlands
[2]TNO Embedded Systems Innovation, Eindhoven, The Netherlands
{f.comaschi, s.stuijk, a.a.basten, h.corporaal}@tue.nl

Abstract

Online detection and tracking of a variable number of faces in video is a crucial component in many real-world applications ranging from video-surveillance to online gaming. In this paper we propose FAST-DT, a fully automated system capable of detecting and tracking a variable number of faces online without relying on any scene-specific cues. FAST-DT integrates a generic face detector with an adaptive structured output SVM tracker and uses the detector's continuous confidence to solve the target creation and removal problem. We improve in recall and precision over a state-of-the-art method on a video dataset of more than two hours while providing in addition an increase in throughput.

1. Introduction

With the number of cameras installed all around the world steadily growing, real-time analysis of video content is becoming increasingly important. Among objects of possible interest, faces have received significant attention both in academia and in industry [15]. The accurate localization of faces in a video is a necessary first step to initialize other computer vision tasks, such as face recognition.

Despite recent advances in both detection [7] and tracking [9] of visual objects, relatively few works directly address the problem of effectively integrating these two components in order to provide a fully automated system. When detecting faces from a video online, the detection component can highly benefit from the integration with a tracker for two main reasons. Firstly, even the multi-view face detectors recently proposed in literature [7, 16] cannot adapt to the targets' appearance variations that occur at run-time. Secondly, even fast face detectors described in literature are quite computationally intensive [13], and their application at every video frame impedes to fulfil the real-time requirement. Relying on temporal coherence through a tracking algorithm can improve the system capability of continuously monitoring an object location in time. From the tracking perspective, any tracking algorithm needs an object detector to provide a fully automated system. In the first place, even the most robust tracking algorithms inherently tend to drift over time, and an object detector can prevent a tracker from loosing its target. Secondly, target creation cannot be addressed by a simple tracker, which therefore needs an object detector to find possible new candidates for tracking. Regarding target removal, even though several tracking algorithms implement some tracking-failure detection mechanism, the support of an object detector is highly beneficial in determining when a target is no longer visible.

For these reasons, building a fully automated system capable of autonomously and effectively locating a variable number of faces online is an important challenge. When building such a system, the most relevant aspects to be considered are the following: (i) the detection component; (ii) the tracking component; (iii) the target creation and removal component; (iv) the design of the integrated framework.

In this paper we address all the mentioned aspects by realising FAST-DT (*FAce STructured Detection and Tracking*) a fully automated system capable of accurately localizing a variable number of faces from a video online. Our main contributions are the following:

1. We address the problem of automatically detecting and tracking a variable number of faces online by building a framework which effectively combines a generic face detector with an adaptive structured output Support Vector Machine (SVM) tracker[1].
2. We provide a solution to the target creation and removal problem based on the continuous confidence provided by the face detector, without relying on any scene-specific long-term observation.
3. We prove the robustness of FAST-DT by testing it on a challenging video dataset of more than two hours [2], where we considerably improve over the state-of-the-art in both recall and precision.
4. We provide an analysis of the computational cost of the different components building FAST-DT and report an increase in throughput over previous work.

[1]Our C++ implementation of FAST-DT is freely available on the website http://www.es.ele.tue.nl/video/

978-1-4673-7632-7/15/$31.00 ©2015 IEEE

2. Related Work

Despite much progress in recent years on multi-view face detection [7, 16], solely relying on a face detector is non-effective in coping with the challenges posed by real-world applications. When training a face detector offline, it is very difficult to provide the detector with enough training samples to cover the full spectrum of appearance variations that will occur at run-time. Also, even the faster face detectors recently proposed in literature [13] require a considerable computational effort, and their application on every video frame will most probably prevent the application from fulfilling its real-time requirements.

An approach that has been recently applied to multi-face tracking is tracking-by-association [8, 11]. These algorithms firstly scan large temporal windows of the video sequence with one or many face detectors and at a later stage rely on several affinity measures to find the best-matching associations between consecutive detections. These methods are meant for offline applications since they rely on information from future frames and the exhaustive application of face detection at every frame represents a computational bottleneck.

A tracking technique which has proven to be very effective is tracking-by-detection [12]. These algorithms adapt to the continuous target appearance variations by training a target-specific classifier online which learns to distinguish the object from the background and from other targets. The robustness of these methods has been mostly proven on short video sequences [12, 9] with the target position manually annotated in the first frame rather than automatically located by a detector. In this paper we integrate a robust structured output SVM tracker with a detector component in order to provide a fully automated system. We prove the system capability of accurately tracking multiple faces across two video sequences of more than one hour each.

The approaches most related to our work are described in [1, 3]. In [1], the authors integrate a generic pedestrian detector with a particle filter in order to track multiple people in a video online. However, their approach is applied to pedestrian tracking, and the authors report an average throughput of 0.4-2 frames per second (fps) on an Intel PC. In this work we specifically address the problem of multi-face tracking and we reach an average throughput of more than 30 fps on an Intel PC with similar computational power.

In [3], the authors effectively track multiple faces from a video in a particle filter framework and they adopt two Hidden Markov Models (HMM) to solve the target creation and removal problem. The system proposed in [3] relies on several long-term observations, some of which assuming a static scenario, like a pixel-based tracker memory for target creation. FAST-DT robustly tracks multiple faces through a structured output SVM and solves the target creation and removal problem solely relying on the continuous confidence provided by a generic face detector. When comparing our results to [3] through the performance metrics proposed by the authors in their work, we reach very similar performance. However, when comparing our results according to the widely used PASCAL overlap measure [4, 9], we reach a considerable improvement in both recall and precision. These results will be motivated in Sec. 4. We prove our system capability to process a video in real-time with an average throughput of more than 30 fps.

3. FAST-DT

Our objective is to build a fully automated system capable of detecting and tracking a variable number of faces online. To reach our aim, we propose the overall framework depicted in Fig. 1. Hereby we motivate the choice of the algorithms composing our framework and we provide a short description for each of them.

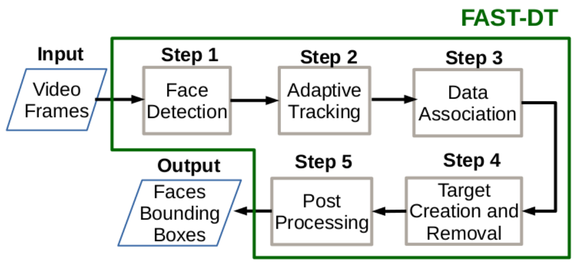


Figure 1: Multi-face detection and tracking framework.

3.1. Detection

In step 1 of FAST-DT, the current video frame is scanned in parallel by multiple face detectors (one for frontal and two for profile view). For effective online multi-face detection, the selected face detectors need to provide a good trade-off between robustness and computational efficiency. In our framework, we decided to use the AdaBoost classifier developed by Viola and Jones [10] for several reasons: (i) the Haar features employed by the Viola-Jones face detector proved to be very effective when it comes to faces [15] and have become almost a standard in face detection; (ii) the sliding-window search mechanism allows to rapidly scan for faces at multiple scales, thus helping the SVM tracker in tracking across rapid scale variations; (iii) the cascade structure provides a natural mechanism for a rapid evaluation of the detector confidence at a given image location, which is used for step 4 of FAST-DT.

Even though more sophisticated face detectors have been proposed in literature [16, 7, 13], the robustness of our tracking component allows us to favour computational efficiency over accuracy when selecting the detector.

3.2. Structured output SVM tracker

In real-world applications, targets normally undergo considerable appearance variations while tracking. For this reason it is important to implement a tracking mechanism which is capable of effectively adapting to these variations in a online fashion. An approach to tracking which has proven to be very effective in addressing this challenge is tracking-by-detection [12], which treats the tracking problem as a detection task applied over time. Most of these algorithms maintain a classifier trained online for each target in order to distinguish the object from the background or other targets [12]. In [6], the authors take a different approach, and treat the tracking problem as one of structured output prediction, in which the task is to directly predict the change in object location between frames. A recent survey [9] proved the superiority of this approach with respect to many contenders. Moreover, in [6] the authors introduce the idea of a budget mechanism which allows to reduce the computational cost of online tracking without noticeably decreasing the tracking performance. This is an important feature for our framework where we aim at tracking multiple targets in real-time. The use of Haar features for image representation makes this approach even more suitable for face-related applications.

For these reasons we based our tracking algorithm (step 2 from Fig. 1) on the structured output SVM framework first introduced in [6]. A kernelized structured output SVM is learned online to provide adaptive tracking. In [14], the authors apply the idea of online structured learning to multiple object tracking by inserting additional constraints. In FAST-DT the use of additional cues, like explicit occlusion detection (Sec. 3.5) and the integration of a generic face detector, allows to effectively learn an independent SVM for each of the tracked targets without taking into account additional constraints in the learning phase.

Each tracker maintains an estimate of the position $\mathbf{p} \in \mathcal{P}$ of a 2D bounding box within frame $\mathbf{f}_t \in \mathcal{F}$, where $t = 1, \ldots, T$ is time. Given a bounding box position $\mathbf{p}$, Haar features are extracted from an image patch within the bounding box: $\mathbf{x}_t^{\mathbf{P}} \in \mathcal{X}$. The objective of the tracker at time t is to estimate a 2D translation of the target $\mathbf{y}_t \in \mathcal{Y}$, where $\mathcal{Y} = \{(\Delta u, \Delta v)|\ \Delta u^2 + \Delta v^2 < r^2\}$, r being a search radius. If $\mathbf{p}_{t-1}$ is the target position at time $t-1$, the 2D position of the target at time t can be found as $\mathbf{p}_t = \mathbf{p}_{t-1} \circ \mathbf{y}_t$ which is given by $(u_t, v_t) = (u_{t-1}, v_{t-1}) + (\Delta u, \Delta v)$. In structured SVM a discriminant function $F : \mathcal{X} \times \mathcal{Y}$ is introduced to find the translation function according to:

$$\mathbf{y}_t = f(\mathbf{x}_t^{\mathbf{p}_{t-1}}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}_t^{\mathbf{p}_{t-1}}, \mathbf{y}) \qquad (1)$$

where F is restricted to be in the form $F(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \mathbf{\Phi}(\mathbf{x}, \mathbf{y}) \rangle$, $\mathbf{\Phi}(\mathbf{x}, \mathbf{y})$ being a joint kernel map. F measures the compatibility between $(\mathbf{x}, \mathbf{y})$ pairs and gives a high score to those which are well matched. In particular, the new tracker location $(\mathbf{x}_t^{\mathbf{p}_t}, \mathbf{y})$ is supplied to the prediction function as a positive labelled sample, while other samples are weighted according to a loss function based on bounding box overlap:

$$\Delta(\mathbf{y}, \mathbf{y}_i) = 1 - s_{\mathbf{p}_t}(\mathbf{y}, \mathbf{y}_i) \qquad (2)$$

where $s_{\mathbf{p}_t}(\mathbf{y}, \mathbf{y}_i)$ is defined as follows:

$$s_{\mathbf{p}_t}(\mathbf{y}, \mathbf{y}_i) = \frac{area((\mathbf{p}_t \circ \mathbf{y}) \cap (\mathbf{p}_t \circ \mathbf{y_i}))}{area((\mathbf{p}_t \circ \mathbf{y}) \cup (\mathbf{p}_t \circ \mathbf{y_i}))} \qquad (3)$$

In this way, F can be learned from a set of example pairs $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ by minimising a convex objective function [6]. During tracking, each tracker maintains a set of support vectors $(\mathbf{x}_i, \mathbf{y}) \in \mathcal{S}$, $\mathbf{x}_i$ being a support pattern. Since evaluating $F(\mathbf{x}, \mathbf{y})$ requires the computation of inner products between $(\mathbf{x}, \mathbf{y})$ and each support vector, and considering that the number of support vectors is not bounded, both the computational and storage costs grow linearly with the number of support vectors. In order to solve this issue only a maximum number of support vectors B for each tracker is allowed. Once the budget of a tracker is exceeded, before the addition of a new support vector a suitable support vector is identified for removal. In general, a higher value of B allows for more accurate tracking while increasing the computational cost. Our experimental validation proves that the integration of the tracking component with a generic face detector allows to reach state-of-the-art performances while keeping the budget as low as $B = 10$, thus allowing for lower computational cost.

3.3. Data Association

In order to decide which detection should guide which tracker, a data association problem needs to be solved. In related work, greedy algorithms or heuristic approaches have been used to limit the computational cost of this step [1, 3]. In our framework it is very important to determine whether a detection shall be matched to a possibly drifting tracker or classified as a new candidate for tracking, therefore we solve the association problem through the Hungarian algorithm which allows for optimal single frame assignment. As we show in Sec. 4.5, the benefits provided by the Hungarian algorithm come at a small cost (a small percentage of the overall framework).

A score matrix S is computed reporting a score for each pair (tr, d) of tracker tr and detection d. Each tracker and each detector are associated to a bounding box $\mathbf{b} = (x, y, w, h)$, where x and y are the top-left corner of the bounding box and w and h are its width and height. The matching score between each pair $(\mathbf{b}_{tr}, \mathbf{b}_d)$ is computed according to the commonly used bounding box overlap ratio [4]:

$$s(\mathbf{b}_{tr}, \mathbf{b}_d) = \frac{area(\mathbf{b}_{tr} \cap \mathbf{b}_d)}{area(\mathbf{b}_{tr} \cup \mathbf{b}_d)} \qquad (4)$$

Only the (tr, d) pairs with a matching score above a threshold, experimentally set to a value of 0.3, are considered valid. Since trackers inherently tend to drift over time, when a matching occurs the detection output is used to reinitialize the associated tracker. We reset the SVM learning process taking the new associated detection as first labelled example. As detailed in the following subsection, unmatched detections are considered as possible candidates for target creation, while unmatched trackers are considered as candidates for removal.

3.4. Target Creation and Removal

Deciding when to add and remove targets is an essential component for an automated detection and tracking system. The cascade structure of the face detectors applied in step 1 of FAST-DT allows to compute the detector confidence at a given image location starting from the stage at which an image patch is rejected from the classifier. This operation is relatively inexpensive, since the computational cost of sliding-window based detectors mostly comes from the large number of image locations to be classified, rather than from the classifier evaluation itself [5].

Let $\mathbf{x^b}$ denote the image patch contained in the bounding box $\mathbf{b}$. A cascade classifier is composed by N stages, each of them computing a real-valued confidence score $c_n(\mathbf{x^b})$, $n \in \{1, \ldots, N\}$ for the given image patch. In the Viola-Jones face detector [10], $c_n(\mathbf{x^b})$ is given by the Haar features computed with the integral image, $c_n : \mathcal{X} \rightarrow \mathbb{R}$. If the image patch is rejected at stage n_r, the overall confidence of the detector for a given image patch $\mathbf{x^b}$ can be computed as:

$$C(\mathbf{x^b}) = k * n_r + c_{n_r}(\mathbf{x^b}) \tag{5}$$

where we select $k = 1000$ to ensure that image patches rejected at higher stages have a higher score and we normalize the value of $C(\mathbf{x^b})$ in $[0,\ 1]$.

For target creation, at each time step t we keep a vector of candidates $\mathbf{v}_t$. In order to limit false positives, unmatched detections from step 3 are not immediately considered as new targets, but they are added to $\mathbf{v}_t$. Then, for a number of f_c consecutive frames, the confidence of each of the three face detectors is evaluated according to Eq. (5) at the corresponding position and the maximum is taken. If the maximum confidence repeatedly scores above a threshold th_c, the candidate is added to the current targets and a new tracker is instantiated, otherwise it is considered to be a false positive and therefore discarded. For target removal, we consider all the trackers which were not matched in step 3 and we evaluate the confidence of the three detectors at the tracker position according to Eq. (5). If the maximum face detector confidence at the target location is below a threshold th_r for f_r consecutive frames, we consider the object to be no longer visible and we remove it from the current targets. The exact values of the mentioned parameters can be found in Sec. 4.4.

The proposed mechanism for target creation and removal requires only 4 parameters to be tuned. It can therefore be adapted to different scenarios. Moreover, since most object detectors provide a confidence density in some form, it can be adapted to different detectors by changing Eq. (5) accordingly. In general, higher values of f_r and f_c are more suitable for relatively static scenarios, while the setting of th_r and th_c depends on the specific detector employed.

3.5. Post-processing

In step 5, we introduce two simple mechanisms to check if any of the current targets is out of scope or largely occluded. In the first mechanism, for each of the current trackers we check if the corresponding bounding box $\mathbf{b}_{tr}$ is out of the boundaries of the current frame. If this condition is met, we consider the target to have abandoned the scene and we remove it from the current trackers. In the second mechanism, we compute the overlap between each pair of trackers according to Eq. (4) to check if any of the trackers is occluded by a bigger object. If the overlap is above a threshold th_o, the smaller bounding box is considered to be occluded and therefore removed from the current trackers.

By adding this simple post-processing step we can avoid the implementation of costly tracking-failure detection mechanisms within the tracker itself, and we avoid additional constraints in the learning phase of the SVM trackers which would slow-down the tracking process.

4. Experimental Results

4.1. Data Set

The great majority of detection and tracking algorithms are either tested on own recorded sequences, which are not available for comparison, or on short videos of a few minutes length [9, 12]. Short videos can hardly capture the great variety of challenging conditions related to tracking, and they are non-effective in proving a system's capability of adding and removing tracks at the proper time.

In order to prove the robustness of FAST-DT, we tested it on the publicly available TA2 dataset [2] which is composed by two videos of more than one hour each. These videos present several challenges such as frequent pose variations, partial or full occlusions, targets entering/leaving the scene and different illumination conditions. Moreover, the selected dataset allows us to provide a quantitative comparison with the state-of-the-art [3].

4.2. Annotations

We compare the output of our system against the publicly available ground truth [2], where the positions and sizes of the faces are described as bounding boxes. Not all the video frames are annotated, and the time between annotated video frames δ_t varies from 0.04 s to 12 s according to the dynamics of the scene.

4.3. Performance Measures

We test FAST-DT using two different sets of metrics. The first set has been introduced in [3] and provides a good indication whether the tracking is completely lost or still on the object, even slightly. These metrics are particularly useful in measuring the performance of a system where the accuracy of the bounding box position is not a strict requirement, e.g., a human-supervised system where the tracking output mainly serves as a visual feedback for a human operator. However, in automated applications in which detected faces are further processed an accurate localization of the tracked faces is an important requirement. For example, if a face recognition step follows the tracking algorithm, it is important to provide the recognizer with an accurate location of the face, including the main facial features.

For these reasons, we also report a second set of metrics which is more restrictive in determining a tracking success [4]. In both set-ups, for every annotated frame we associate the bounding boxes in output from FAST-DT with the ground-truth annotations through the Hungarian algorithm. However, the matching scores are defined differently.

1) Metrics I: in the first set of metrics, the score between matching pairs is computed according to:

$$F = \frac{2 * area(\mathbf{b}_o \cap \mathbf{b}_g)}{area(\mathbf{b}_o) + area(\mathbf{b}_g)} \tag{6}$$

where $\mathbf{b}_g$ is the ground-truth bounding box and $\mathbf{b}_o$ is the bounding box output from FAST-DT. A match is considered successful if the $F\text{-}measure$ as defined in Eq. (6) is greater than 0.1. In [3] the authors further define R and FP as[2]

$$R = \frac{\sum_{i=2}^{G} \delta_i d_i}{\sum_{i=2}^{G} \delta_i}, \qquad FP = \frac{\sum_{i=2}^{G} \delta_i f_i}{\sum_{i=2}^{G} \delta_i} \tag{7}$$

where G is the number of annotated frames, d_i is the proportion of correctly detected/tracked faces in frame i, f_i is the number of false positives divided by the number of ground truth objects in frame i and δ_i is the time difference between frames i and $i-1$.

2) Metrics II: in the second set of metrics, we adopt the standard followed by the PASCAL VOC challenge [4]. In this case, the overlap between $\mathbf{b}_o$ and $\mathbf{b}_g$ is computed according to Eq. (4), and a threshold of 0.5 is considered. This criterion is more restrictive and therefore it is a better indication of the accurate localization of the faces by the tracking system. We further measure the widely adopted $recall$ and $precision$ as defined in [9], but in order to account for the time difference between consecutive annotated frames, we weight them according to δ_i, as we already did for the previous set of metrics:

$$recall = \frac{\sum_{i=2}^{G} \delta_i r_i}{\sum_{i=2}^{G} \delta_i}, \quad precision = \frac{\sum_{i=2}^{G} \delta_i p_i}{\sum_{i=2}^{G} \delta_i} \tag{8}$$

where r_i is the proportion of correctly detected/tracked faces in frame i and p_i is the number of true positives divided by the number of system outputs. We remark here that d_i and r_i are defined in the same way. However they are based on different matching criteria (Eq. (6) and Eq. (4) respectively, with thresholds of 0.1 and 0.5); therefore they lead to different numbers.

4.4. Experimental Set-up

In order to provide a fair comparison with the state-of-the-art, in our experimental set-up we followed the indications provided by the authors in [3]. Step 1 of Fig. 1 is applied to a down-scaled version of the video frames (640 $\times$ 360 pixels) and the original video frame rate of 25 fps has been changed to 12.5 fps. This means that for odd frames, we output the bounding boxes from the previous frame. The face detector threshold for non-maximum suppression (Sec. 3.1) has been set to 4. The detection step is run only once every 10 frames, as in [3]. Also steps 3 and 4 from Fig. 1 are run once every 10 frames. In principle, the detector confidence evaluation for target creation and removal could be run every processed frame. However, in our experiments we found that applying these steps every frame slows down the algorithm without noticeably improving the accuracy. The budget for the online SVM trackers has been set to $B = 10$. All the other tracking parameters have been set to the values reported by the authors in [6]. The thresholds for target creation and removal have been set to $th_c = 0.8$ and $th_r = 0.5$, respectively. We set $f_r = 25$ and $f_c = 5$, which correspond to 10 s and 2 s of video respectively, considering that step 4 is run once every 10 frames. The overlap threshold is set to $th_o = 0.3$ for both step 3 and the occlusion detection mechanism of step 5. All the experiments have been run on a Intel Core i7 with a 3.07 GHz clock (single thread).

[2]We use the terms R and FP rather than $Recall$ and $False\ positive\ rate$ to better distinguish between the two different set of metrics we report.

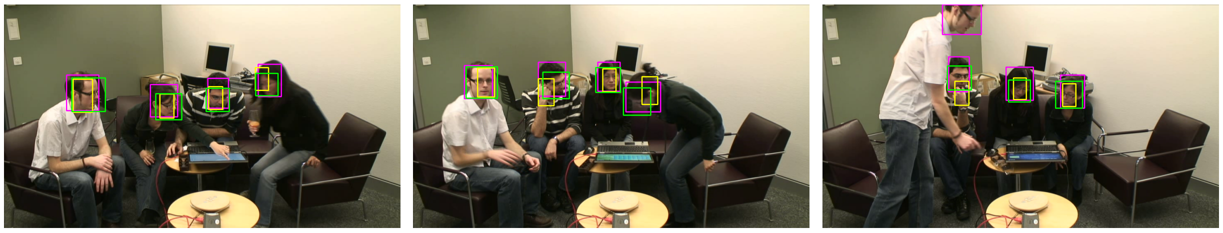
4.5. Quantitative Results

	Metrics I [3]		Metrics II [4]		
Method	R	FP	Recall	Precision	Throughput
FAST-DT	93.4%	2.2%	85.2%	89.6%	34 fps
Duffner & Odobez [3]	93.7%	1.2%	42.1%	45.4%	20-23 fps

Table 1: Performance comparison with the state-of-the-art.

1) Accuracy: in Tab. 1 we compare FAST-DT with the state-of-the-art results reported in [3]. For the recall and precision metrics, we computed them from the output file made available by the authors. We remark that because of the randomness involved in our tracking algorithm [6], the reported numbers for FAST-DT are the median results of three consecutive runs over the video sequences. Therefore, decimal differences reported in the table should be considered with care. From Tab. 1, we can see that when reporting the metrics adopted in [3], we reach similar performance, with a 1% difference in FP. For the PASCAL VOC criteria [4], we notice a considerable improvement in both $recall$ and $precision$. This is because FAST-DT is more effective in accurately determining the exact location of the faces from the video. The improvement in recall and precision can be ascribed to the robustness of the structured SVM tracker; our detection component is the same as in [3], while a more robust target creation and removal component would have resulted in an improvement also in R and FP. In Fig. 2 we report some example frames to illustrate the better recall and precision of FAST-DT with respect to [3].

2) Computational Cost: in Fig. 3 we report the average time cost in seconds for each of the components of FAST-DT, plus the overhead due to frame decoding and conversion. The average time per frame when running FAST-DT on the TA2 dataset is 0.029 s, thus leading to an average throughput of 34 fps, which represents around a 50% improvement over [3], as shown in Tab. 1. Our results and those reported in [3] have been obtained on Intel PCs with similar processing power. The average time per frame spent by the detection component reported by the authors in [3] is very close to the one we obtain, these being 0.013 s and 0.012 s respectively. A more detailed comparison is not possible because the implementation of [3] is not available;



(a) (b) (c)

Figure 2: Example frames of FAST-DT output (green), Duffner & Odobez [3] output (yellow) and ground-truth (pink). Even though the outputs from both FAST-DT and [3] are in the proximity of the targets, we can see that FAST-DT is more effective in accurately determining the faces location.

moreover, [3] does not report the time cost for all the algorithmic components, and the work includes a person identification step that has no equivalent in our framework.

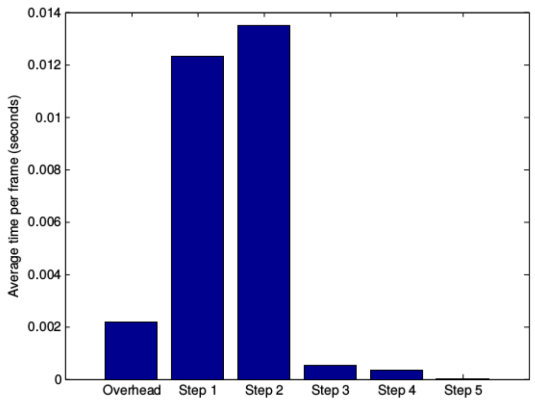


Figure 3: Average time cost (seconds) per frame of the proposed method on the TA2 dataset.

From Fig. 3 we can see that even though the detection (step 1) is run only once every 10 frames, it still accounts for 41% of the total computation time. The tracking component (step 2) is the most expensive, requiring 48% of the total time. This implies that the detection and tracking components alone account for nearly 90% of the total computation time. Our track creation and removal approach (step 4) is computationally efficient and its cost is almost negligible with respect to the overall framework.

5. Conclusions and Future Work

In this paper we propose FAST-DT, a completely automated system for online multi-face detection and tracking. FAST-DT combines multiple Haar-feature based face detectors with an adaptive structured output SVM tracker. It solves the target creation and removal problem using the continuous detector confidence. We noticeably improve the performance over the state-of-the-art while providing a higher throughput. In future work, additional target-specific knowledge could be added to allow for target identification throughout a video sequence.

Acknowledgment: This work was supported in part by the COMMIT program under the SenSafety project.

References

[1] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. J. V. Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(9):1820–1833, 2011.

[2] S. Duffner, P. Motlicek, and D. Korchagin. The ta2 database: A multi-modal database from home entertainment. In *ICSAP*, 2011.

[3] S. Duffner and J. Odobez. Track creation and deletion framework for long-term online multiface tracking. *IEEE Trans. Image Processing*, 22(1):272–285, 2013.

[4] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[5] G. Gualdi, A. Prati, and R. Cucchiara. Multistage particle windows for fast and accurate object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(8):1589–1604, 2012.

[6] S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011.

[7] M. Mathias, R. Benenson, M. Pedersoli, and L. V. Gool. Face detection without bells and whistles. In *ECCV*, 2014.

[8] M. Roth, M. Bäuml, R. Nevatia, and R. Stiefelhagen. Robust multi-pose face tracking by multi-stage tracklet association. In *ICPR*, 2012.

[9] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1442–1468, 2014.

[10] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

[11] B. Wu, S. Lyu, B. Hu, and Q. Ji. Simultaneous clustering and tracklet linking for multi-face tracking in videos. In *ICCV*, 2013.

[12] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013.

[13] J. Yan, Z. Lei, L. Wen, and S. Z. Li. The fastest deformable part model for object detection. In *CVPR*, 2014.

[14] W. Yan, X. Han, and V. Pavlovic. Structured learning for multiple object tracking. In *BMVC*, 2012.

[15] C. Zhang and Z. Zhang. A survey of recent advances in face detection. *Technical report MSR-TR-2010-66*, 2010.

[16] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, 2012.