Research Article
Sensor-Assisted Face Tracking

Dingbo Duan\(^1\) and Jian Ma\(^2\)

\(^1\)Beijing University of Posts and Telecommunications, Beijing 100876, China
\(^2\)Smart Sensing Stars, Co. Ltd., Wuxi 214000, China

Correspondence should be addressed to Dingbo Duan; fatcat.bupt@gmail.com

Received 10 October 2014; Revised 11 December 2014; Accepted 12 December 2014

1. Introduction

Locating and tracking faces in video streams have long been the most fundamental techniques in computer vision. They are step stones of almost all facial analysis algorithms including face alignment, face modeling, face recognition, and gender/age recognition and have enabled numerous applications, such as human-computer interaction (HCI), video surveillance, and many other multimedia applications. Particularly in the context of HCI, only when computers could understand human faces well could they begin to truly figure out people’s intentions and thoughts and react in a proper manner.

In general, the goal of face detection is to determine whether or not there are any faces present in an arbitrary image and, if present, return the location and extent of each face. While this appears to be a simple task for human beings, it is very difficult for computers and has been a hot topic in machine vision that attracts top researchers all over the world in the past few decades. The difficulties associated with face detection can be attributed to many variations of lighting conditions, scale, location, orientation, pose, facial expression and occlusions, and so forth. In addition, intraclass interferences that arise from make-up, beard, mustache, and glasses of the same person make face detection problem even more hard.

In recent years, face detection has made significant progress and been increasingly utilized in real world applications and products, like Google’s Picasa. Nowadays most digital cameras are equipped with built-in face detector to help autofocus. However, face detection in unconstrained settings remains a challenging task. Modern face detection algorithms are mostly based on low level feature extraction and statistic model training and focus their attention wholly on visual data analysis. Herein more and more complex features and rigorous learning algorithms are developed to extract as much information as possible from the visual content.

Automatic face tracking requires face detection to initialize the tracking process. It is an application of object tracking. In its simplest form, tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene [1]. In terms of face tracking, a tracker assigns consistent labels to detected faces in consecutive video frames. The main challenge in tracking is clutter. Clutter is the phenomenon when features expected from the target are difficult to discriminate against features extracted from other objects in the scene. Another challenge
is introduced by appearance variations of the target itself. Intrinsic appearance variability includes pose variation and shape deformation, whereas extrinsic appearance variability includes illumination change, camera motion, and different camera viewpoints [2].

In this paper, we focus our attention to handle the task of face tracking with a new perspective. Typical face detection and tracking are conducted frame by frame and window by window. In terms of face detection, the time spent on filtering out a faceless frame is comparable to that on identifying a frame containing faces as every search window needs to be checked to ensure all possible faces are detected. Faces are tracked in following frames using relatively less computationally expensive methods. In case of track failure, face detection runs again to reinitialize the tracker. Large amount of time is wasted on searching faces in faceless frames. For example, when a subject in video turns his back and walks away from the camera, his face totally disappears and from that moment on there is no need to apply face detection and tracking. To improve the performance and cut down time cost, we take advantage of context metadata collected at the time of video capture in a sensor-assisted environment to rule out potential faceless frames.

The rapid advances in consumer electronics have led to a wide proliferation of cheap powerful wearable sensors, such as accelerometer, digital compass, gyroscope, and GPS. The availability of these sensors initially included in smartphones to improve user experience is now changing the landscape of potential applications and providing reliable sources of contextual information which helps to model human behavior. In this study, we employ smart phones as sensing platforms to collect measurements of orientation sensor and help interpreting human moving direction which is explored and utilized to improve face detection and tracking. To summarize, the main contributions of this paper are twofold. First, we present a sensor-assisted fast face detection and tracking approach. As far as we know, it is the first attempt to integrate personal sensing technologies into face detection and tracking in video. This integration of a new sensing model broadens the domain of semantic analysis of visual content and will be catalyzed by the growing popularity of wearable devices and concurrent advance in ubiquitous computing. Second, we implement a set of state-of-the-art multijbject tracking algorithms and conduct extensive experiments to evaluate our method.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces the problem we tend to deal with. Section 4 details the proposed method. Section 5 describes our experiments together with result analysis. Concluding remarks are placed in Section 6.

2. Related Work

2.1. Face Detection. There have been hundreds of reported approaches to handle the problem of face detection. Based on early works of Yang et al. [3], existing face detection approaches can be grouped into four categories: knowledge-based methods, feature invariant approaches, template matching methods, and appearance-based methods. Knowledge-based methods employ predefined rules to determine face presence based on human knowledge; feature invariant approaches aim to find face structure features that are robust to pose and lighting variations; template matching methods make use of prestored face templates to judge if a face exists in an image; appearance-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and nonface images. The learned characteristics are in the form of distribution models or discriminant functions that are consequently used for face detection. Meanwhile, dimensionality reduction is usually carried out for the sake of computation efficiency and detection efficiency. Among these approaches, appearance-based methods have distinguished themselves as the most promising ones and had been showing superior performance to the others.

There are mainly two important factors that determine the success of a face detector: the features used for representing face images and the learning algorithm that implements the detection. Histogram based features have become very popular in recent years due to their excellent performance and efficiency, including local binary patterns [4], local ternary patterns [5], and histograms of oriented gradients [6]. Most state-of-the-art face detection methods usually use a combination of these features by concatenating them or by optimizing combination coefficients at the learning stage.

In terms of learning, most approaches treat face detection problem as a binary classification problem and determine whether current search window contains a face. Various machine learning methods ranging from the nearest neighbor classifier to more complex approaches such as neural networks, convolution neural networks, and classification trees have been employed for face detection. Among them, boosting based cascades have attracted a lot of research interest. Viola and Jones [7] introduced a very efficient face detector by using AdaBoost to train a cascade of pattern-rejection classifiers over rectangular wavelet features. Each stage of the cascade is designed to reject a considerable fraction of the negative cases that survive to that stage, so most of the windows that do not contain faces are rejected early in the cascade with comparatively little computation. As the cascade progresses, rejection typically gets harder so the single-stage classifiers grow in complexity. The structure of the cascaded detection process is essentially that of a degenerate decision tree. In our work, we employ this detector to initialize face tracking.

2.2. Object Tracking. Two tracking paradigms have been presented in [8]. Recursive tracking methods estimate current state \( x_t \) of an object by applying a transformation on the previous state \( x_{t-1} \) based on measurements \( z_1 \cdots z_t \) taken in the respective images. The recursive estimation of a state depends on state of the object in previous frame and is susceptible to error accumulation. For instance, Lucas and Kanade [9] propose a method for estimating sparse optical flow within a window around a pixel. The optic flow is fit into a transformation model that is used to predict new position
of the object. Comaniciu et al. [10] propose a tracker based on mean shift. The transformation of the object state is obtained by finding the maximum of a similarity function based on color histograms, while tracking-by-detection methods [11, 12] treat tracking as a classification problem and train a classifier to distinguish the object from the background. The object detector (classifier) could be static or updated online. In [13], a SVM (support vector machine) classifier is integrated into an optic-flow-based object tracker. However, the SVM classifier is trained beforehand and unable to adapt. Adaptive discriminative trackers [14, 15] train and update a classifier using new training examples acquired during tracking.

The research efforts mentioned above focus their attention wholly on the analysis of visual data. While in this study, we provide a novel method to exploit contextual information collected at the point of video recording to help face detection and tracking in video.

### 3. Problem Formulation

Subjects carrying smart phones move around casually in the FOV of a fixed digital camera. Video data are continuously recorded and motion measurements are collected by embedded sensors on the phone. As depicted in Figure 1, direction measurements captured by orientation sensor contained in the red brace indicate that the subject is moving toward camera, and during this period the camera could most likely capture clear faces. Based on this judgement, we could apply face detection and tracking directly to frames recorded in this period and skip faceless frames before and after. Our objective is to figure out these advantageous situations and improve face detection and tracking in various situations with the help of on-body sensors.

### 4. Proposed Method

In this section, we elucidate the proposed method in detail. As illustrated in Figure 2, \( \overrightarrow{AO} \) is direction of camera lens and can be adjusted as needed. As camera FOV is symmetric about \( \overrightarrow{AO} \), we just focus our analysis on half of the FOV. \( \overrightarrow{OC} \) is moving direction of a subject carrying smart phones. We assume that moving direction of a subject keeps consistent with his facing direction. The less \( \angle AOC \) is, the more likely a face will appear. When \( \angle AOC \) equals 0\(^\circ\), the subject moves right towards the camera and frontal faces could be then captured. When \( \angle AOC \) is less than 90\(^\circ\), a portion of facial features are lost and performance of face detection and tracking varies with different algorithms. When \( \angle AOC \) is equal to or greater than 90\(^\circ\), detailed features of frontal faces totally disappear and most face detection algorithms fail to locate any faces. Thus we only need to consider situations satisfying 0\(^\circ\) \( \leq \angle AOC \leq 90\(^\circ\).

Based on the above analysis, we propose a two-stage automatic face tracking framework. In first stage, context metadata collected at the time of video capture are scanned to identify advantageous situations for visual analysis. Video frames are automatically labeled to indicate whether they contain faces. In second stage, face detection and tracking are conducted over the labeled frames to locate and track all faces.

#### 4.1. Sensor Description

Two types of sensor are involved in the proposed method, camera sensor and orientation sensor. Video streams recorded from the camera sensor are saved on the disk as discrete files. In this subsection, we put emphasis on orientation measurements collection from orientation sensor on smart phones.

Currently most smart phones are equipped with various types of specialized sensors originally aimed at improving user experience, including orientation sensor. An orientation sensor usually consists of an accelerometer and a magnetometer and can sense the orientation of a smart phone relative to the earth with three different values, \( \textit{pitch} \), \( \textit{roll} \), and \( \textit{azimuth} \), as shown in Figure 3. \( \textit{Pitch} \) indicates rotation about the \( x \)-axis and ranges from \(-180^\circ \) to \( 180^\circ \) inclusively, with positive values when the \( z \)-axis tilts toward the \( y \)-axis; \( \textit{roll} \) indicates rotation about \( y \)-axis and ranges from \(-90^\circ \) to \( 90^\circ \) inclusively, with positive values when the \( z \)-axis tilts toward the \( x \)-axis. \( \textit{Azimuth} \) indicates angle between the \( y \)-axis and magnetic north direction and ranges from \( 0^\circ \) to \( 359^\circ \) inclusively. Experiments have demonstrated that with the specified phone attachment shown in Figure 3, \( \textit{azimuth} \) angle of the smart phone could be utilized to estimate moving direction of human body.
4.2. Frame Labeling. We conduct a preliminary experiment to quantitatively measure impacts of moving direction on face detection. First, we divide \( \angle AOB \) equally into six sectors labeled from sector 1 to sector 6 counterclockwise. Each sector covers an angle of 15°, as shown in Figure 4. We recruit three participants to move towards the camera in each sector from a far point where his face could be barely distinguished with the naked eye. To rule out influence of interference factors, we require that the participant moves in his most relaxed way in order to maintain the same motion status in all sectors and keep facing his moving direction as far as possible. We use a Logitech C615 HD webcam to record video clips, as shown in Figure 3. Video resolution and frame rate are set to 640 \( \times \) 480 and 20 frames per second, respectively. Video frames are encoded with libx264 and stored in the format of mp4 files on a Ubuntu 12.10 PC.

We manually count the number of faces in every video clip and then perform face detection using a Haar feature based face detector [7] in OpenCV [16]. Statistics about collected data and detection results are listed in Table 1, where \( VNo(i) \) is the number of video clips recorded in sector \( i \). Three participants each repeat their motion five times and 15 video clips are recorded in each sector. \( \text{AvgFr}(i) \) is averaged number of frames in video clips in sector \( i \), where

\[
\text{AvgFr}(i) = \frac{\sum_{j=1}^{VNo(i)} \text{FrNo}(i, j)}{VNo(i)}, \quad 1 \leq i \leq 6; \quad (1)
\]

\( \text{FrNo}(i, j) \) is the number of frames of video clip \( j \) in sector \( i \). As listed in Table 1, \( \text{AvgFr}(i) \) decreases from 102 in sector 1 to 52 in sector 6. This is mainly due to the fact that when each participant starts from the same position and moves towards the camera with the same speed, the smaller the angle \( \angle AOC \) in Figure 2 is, the shorter the time he stays in camera FOV and the less the frames will be recorded in the corresponding sector. \( \text{AvgFa}(i) \) and \( \text{AvgFaD}(i) \) are, respectively, the averaged
Input:

\( O_j \): azimuth angle from orientation sensor, \( 1 \leq j \leq M \).
\( O_c \): direction of camera lens.
\( F_i \): video frame \( i \), \( 1 \leq i \leq N \).
\( \theta \): threshold angle for frame labeling.
\( w \): time window in seconds.

Functions and Variables:
\( t() \): timestamp extraction function.
\( t_1 \): starting point of a time window.
\( t_2 \): ending point of a time window.
\( b \): a boolean variable indicating whether frames within a time window should be labeled positive.

Output:
\( f_i \): boolean variable indicating whether \( F_i \) contains faces.

1. \( f_i \leftarrow false \), \( t_1 \leftarrow \max(t(F_i),t(O_j)), t_2 \leftarrow t_1 + w \)
2. while \( t_2 \leq \min(t(F_N),t(O_M)) \) do
3. \( b \leftarrow false \)
4. for all \( O_j, t_1 \leq t(O_j) \leq t_2 \) do
5. if \(|O_j - O_c| \leq \theta \lor (360 + O_j - O_c) \leq \theta \lor (360 - O_j + O_c) \leq \theta \) then
6. \( b \leftarrow true, break \)
7. end if
8. end for
9. if \( b \) is true then
10. for all \( t_1 \leq t(F_i) \leq t_2 \) do
11. \( f_i \leftarrow true \)
12. end for
13. end if
14. \( t_1 \leftarrow t_2, t_2 \leftarrow t_2 + w \)
15. end while

Algorithm 1: Frame labeling.

numbers of manually labeled and detector predicted faces in video clips in sector \( i \), where

\[
\text{AvgFa}(i) = \frac{\sum_{j=1}^{\text{VNo}(i)} \text{FaNo}(i,j)}{\text{VNo}(i)}, \quad 1 \leq i \leq 6,
\]

\[
\text{AvgFaD}(i) = \frac{\sum_{j=1}^{\text{VNo}(i)} \text{FaDNo}(i,j)}{\text{VNo}(i)}, \quad 1 \leq i \leq 6;
\]

\( \text{FaNo}(i,j) \) and \( \text{FaDNo}(i,j) \) are, respectively, the number of manually labeled and detector predicted faces in video clip \( j \) in sector \( i \).

From the result we could conclude that faces, either manually labeled (AvgFa) or detector predicted (AvgFaD), start to decrease dramatically from sector 3, and the detector could barely detect any faces when participants move in the last two sectors (sector 5 and sector 6), even though a dozen of profile faces could be distinguished by human eyes. This is due to the absence of enough facial features. The detected faces in sectors 4 and 5 are mostly false positive results (fake faces). Thus we could define advantageous situations as \( |\triangle AOC| \leq 45^\circ \) and skip frames on other occasions. Although it is a coarse-grained threshold, we could demonstrate the great improvement it brings to face detection and tracking in the following sections.

With the obtained threshold, we design a frame labeling algorithm, as listed in Algorithm 1. All collected orientation measurements are first smoothed to reduce noise and then scanned to label video frames recorded at that time period to prepare for face detection and tracking. Once a qualified azimuth sample is detected within a time window, all frames within that window will be labeled positive. By this strategy we could alleviate the adverse impacts of sudden body turning. With respect to situations when multiple subjects exist, to simplify the problem, we assume that all subjects stay within camera FOV in all experiments. Final result of frame labeling is calculated by applying logical OR to the set of \( f_i \) obtained from analyzing orientation measurements of each subject.

4.3. Face Tracking. In this section, we provide three sensor-assisted tracking algorithms to track faces in the labeled frame sequences, including track by detection, track by mean shift, and track by TLD (tracking-learning-detection). In these algorithms, we employ the Viola-Jones face detector [7] to initialize the tracking process. Moreover, to reduce detection time, we filter out nonskin areas from each frame using a skin model presented in [17]. To deal with multiface tracking, we design a face classification algorithm to group different faces, as shown in Algorithm 2. The logic of the algorithm is straightforward. For each face, we search the most similar descriptor by comparing the normalized face patches. In this paper we resize each face to a \( 15 \times 15 \) patch in pixel before comparison. Patch similarity is conducted on
Input:

\( f_{\text{face}} \): an image patch containing a human face.
\( f_{\text{Desc}} \): a vector of face descriptor \( f_{\text{Desc}} = [\text{hist}, b_{\text{Box}}, n_{\text{Patches}}, f_{\text{Idxes}}] \).

\( \text{hist} \) is HSV color histogram of a face.
\( b_{\text{Box}} \) is the bounding box of a face and calculated by \( b_{\text{Box}}(f_{\text{face}}, b_{\text{Box}}) \).
\( n_{\text{Patches}} \) is a vector of normalized face patch \( n_{\text{Patch}} \) calculated by \( \text{resize}(f_{\text{face}}, n_{\text{Patch}}) \).
\( f_{\text{Idxes}} \) is a vector of indexes of frames that containing the face described by \( f_{\text{Desc}} \).

\( f_{\text{Idx}} \): index of current frame.

\( \theta_{\text{ncc}} \): threshold value for similarity of two normalized face patches.

(1) Track by Detection. As shown in Algorithm 3, we apply face detection over each positively labeled frame containing faces. The detected faces are then classified using Algorithm 2. The performance of this algorithm totally relies on the generalizability and representability of training samples of the detector. We use this algorithm as a benchmark for parameter optimization in Section 5.1.

(2) Track by Mean Shift. We provide a tracking algorithm based on mean shift [10] in Algorithm 4. Mean shift is a procedure for locating the maxima of a density function given discrete data sampled from that function. In \( \text{meanShift}(F_{i}, f_{\text{Desc}}, f_{\text{Desc}.b_{\text{Box}}, f_{\text{Desc}.b_{\text{Box}}}} \), a confidence
**Algorithm 3: Track by detection.**

**Input:**
- $F_i$: video frame $i$, $1 \leq i \leq N$, $N$ is video frame count.
- $f_i$: boolean variable indicating whether $F_i$ contains faces.

**Output:**
- $f_{Desc}$: vector of face descriptors recording which frame a face appears.

1. for all $i$, $1 \leq i \leq N$ do
2.   if $f_i$ is false then
3.     continue
4.   end if
5.   Detect faces from $F_i$ and save them to faces
6.   for all face in faces do
7.     classify(face, $f_{Desc}$, $i$)
8.   end for
9. end for

**Algorithm 4: Track by mean shift.**

**Input:**
- $F_i$: video frame $i$, $1 \leq i \leq N$, $N$ is video frame count.
- $f_i$: boolean sequence indicating whether $F_i$ contains faces.
- int: the interval of track reinitialization.
- $\theta$: threshold value for rectangle overlap.

**Output:**
- $f_{Desc}$: vector of face descriptors recording which frame a face appears.

1. $j \leftarrow 0$
2. for all $i$, $1 \leq i \leq N$ do
3.   if $f_i$ is false then
4.     continue
5.   end if
6.   if $j$ % int = 0 then
7.     Detect faces from $F_i$ and save them to faces
8.     for all face in faces do
9.       classify(face, $f_{Desc}$, $i$)
10.    end for
11. else
12.     for all $f_{Desc}$ in $f_{Desc}$ do
13.       Get last index $fIdx$ from $f_{Desc}.fIdxes$
14.       if $fIdx + 1 \neq i$ then
15.         continue
16.       end if
17.       meanShift($F_i$, $f_{Desc}.hist$, $f_{Desc}.bbox$, rect)
18.       if overlap($f_{Desc}.bbox$, rect) > $\theta$, then
19.         $f_{Desc}.bbox \leftarrow$ rect
20.         $f_{Desc}.fIdxes.push.back(i)$
21.         calcHist(rect, hist), $f_{Desc}.hist \leftarrow$ hist
22.       end if
23.     end for
24. end if
25. $j \leftarrow j + 1$
26. end for

map of the face $f_{Desc}$ in current frame $F_i$ is first created using its color histogram $f_{Desc}.hist$ in previous frame; then by searching for a local peak using the confidence map, $rect$ is estimated as most probable position of the face in current frame based on its previous position $f_{Desc}.bbox$. To filter out long lost faces, we only consider face descriptors that keep active till the previous frame. We adopt spatial overlap of face bounding boxes as a metric to distinguish successful detections, as defined in (4). $b_1$ and $b_2$ are bounding boxes of two faces. $overlap(b_1, b_2)$ ranges from 0 to 1. A fast move of
head may cause a zero overlap. When overlap($b_1, b_2$) is below a threshold value, a track failure occurs. To handle these situations, we conduct face detection at a fixed interval to reinitialize the tracking process, as illustrated in Algorithm 4:

\[
\text{overlap}(b_1, b_2) = \frac{b_1 \cap b_2}{b_1 \cup b_2}.
\]

Mean shift algorithm has difficulty in tracking fast moving targets and suffers from local optimal problem caused by the mountain climbing optimum algorithm it used for searching rest.

(3) Track by TLD. TLD was proposed by Kalal et al. [18, 19]. It is a framework designed for long-term tracking of an unknown object in unconstrained environments. The object is tracked and simultaneously learned in order to build a detector that supports the tracker once it fails. The detector is built upon the information from the first frame as well as the information provided by the tracker. The original TLD tracker tracks only one object. We create a multijob object tracker based on OpenTLD provided in [20]. In Algorithm 5, we first initialize TLD tracker in \text{initTLD}(fDescs), where \text{fDescs} contains descriptive information of faces detected from the first $N_1$ positively labeled frames. By initialization, an internal classifier is trained for each face in \text{fDescs} using training data extracted from the first $N_1$ positive frames. Positive training samples are obtained by applying affine warping transformation to the detected face region. Negative samples are obtained by collecting rectangles with similar dimension from nonface areas in the frame. In addition, positive samples of one face are added to negative training sets of other faces for performance improvement. While tracking, \text{TLD}(F_i, rects) tracks and detects the faces in parallel in each new frame $F_i$ and fuses the result of internal tracker and detector into \text{rects}. Then, positive and negative training samples are again extracted based on the fusion result to update the detector. The main limitation of the algorithm is that it cannot track faces that appear after the initialization. Thus, this method can only apply to situations where count of subjects will not increase. In this paper, we set $N_1$ to values that will include all faces that will appear in a video.

5. Experiments

In this section, we conduct extensive experiments to evaluate the proposed method. In addition to video devices and
capture settings used in Section 4.2, we also utilize Android smart phones equipped with orientation sensors. Two subjects are recruited to take part in our experiments and phones are attached to waist belt where moving direction of subjects could be best approximated, as shown in Figure 3. A simple GUI application is created to start and stop data collection on phones. Orientation measurements are recorded and saved in text files on phone SD card and later accessed via USB.

We implement labeling and tracking algorithms proposed in Section 4 based on OpenCV library [16]. Experiments are performed in various situations including indoor single-face, indoor multiple-face, outdoor single-face, and outdoor multiple-face. Subjects move randomly within part of camera FOV where their faces could be distinguished by the naked eye. In each situation, we repeat the experiment four times and each lasts about five minutes. In all we collect sixteen
5.1. Tracking Optimization. To label frames using Algorithm 1, we set $\theta$ to $45^\circ$ which has been experimentally proved in Section 4.2. In terms of time window $w$, a larger window leads to less missed faces at the cost of potentially more faceless frames. We define a metric in (5) to measure the performance of Algorithm 1 on video $i$. $P(i)$ is a ratio between number of positively labeled frames containing faces and number of positively labeled frames. It ranges from 0 to 1. The larger $P(i)$, the more efficient Algorithm 1. To rule out interference factors in multiple-face situations like mutual occlusion of human bodies, we run Algorithms 3 and 1 over the single-face data with different $w$. The averaged $P$ obtained in indoor and outdoor situations is illustrated in Figure 5. An optimal $P$ is achieved in the vicinity of $w = 0.5$:

$$P(i) = \frac{\#\text{positively labeled frames having faces}}{\#\text{positively labeled frames}}. \quad (5)$$

The threshold $\theta_{acc}$ in Algorithm 2 affects face classification accuracy and varies with different tracking algorithms and application scenarios. We use a value of $\theta_{acc} = 0.65$ for all our experiments. In Algorithm 4, two parameters...
Figure 10: Extracted frames of tracking results, where (a1)–(f1) are from track by detection, (a2)–(f2) are from track by mean shift, and (a3)–(f3) are from track by TLD.
will affect tracking performance. When overlap between bounding boxes of a face tracked across two consecutive frames is below $\theta_o$, a track failure occurs. The larger $\theta_o$ is, the more likely a positive true tracking result will be obtained. An optimal $\theta_o$ varies with different human motion patterns. In our experiment, we set $\theta_o$ empirically to 0.4. The parameter $int$ specifies the interval at which we reinitialize trackers to deal with tracking failures. When $int = 1$, Algorithm 4 degrades into Algorithm 3 and it applies face detection on each positively labeled frame. When $int$ is set to the length of a video clip, Algorithm 4 stops as soon as mean shift fails. A larger $int$ costs less detection time and suffers the risk of high probability of false faces. We employ metrics defined in ((6a), (6b), and (6c)) and run Algorithm 4 on the single-face data.

$F$-score can be interpreted as a weighted average of precision and recall; precision is the fraction of face detections that are true faces; recall is the fraction of true faces that are detected; $tp$ is the number of detections that are true faces; $fp$ is the number of detections that are false faces; $fn$ is the number of missed faces. $Precision$, $recall$, and $F$-score reach their best at 1 and worst at 0.

As illustrated in Figure 6, Algorithm 4 achieves its best result in both indoor and outdoor environment when using an interval of 20 frames.

$$
\text{precision}(i) = \frac{tp(i)}{tp(i) + fp(i)}, \quad (6a)
$$

$$
\text{recall}(i) = \frac{tp(i)}{tp(i) + fn(i)}, \quad (6b)
$$

$$
F\text{-score}(i) = 2 * \frac{\text{precision}(i) \times \text{recall}(i)}{\text{precision}(i) + \text{recall}(i)} \quad (6c)
$$

5.2. Tracking Comparison. In this subsection, we compare the sensor-assisted face tracking algorithms depicted in Algorithms 3, 4, and 5 with their sensorless counterparts in terms of performance and processing speed. To conduct sensorless face tracking, we just set elements of $f_i$, the output of Algorithm 1, all to true and keep subsequent tracking algorithms unchanged. We run the optimized algorithms on the collected data and calculate the averaged results under each situation. As shown in Figure 7, the sensor-assisted algorithms achieve comparable performance to their sensorless counterparts in terms of recall, while precision of the sensor-assisted version obviously outweighs the sensorless ones. It is attributed to the removal of false alarms that might be caused over negatively labeled frames, as illustrated in Figure 8. In addition, due to exclusion of faceless frames, sensor-assisted tracking algorithms achieve higher processing speeds, as illustrated in Figure 9. The superiority becomes more evident especially under single-face situations where a comparatively larger percentage of frames are labeled negatively. Extracted frames from the results are illustrated in Figure 10.

6. Conclusion

In this paper, we propose a novel method for fast face tracking. The method innovatively leverages sensor captured contextual information and could be utilized as a preliminary step to assist various algorithms for face detection and tracking in video. Experiment results demonstrate the performance improvement brought by the proposed method. However, the method is limited in the following aspects. First, users have to register and carry their smart phones in order to facilitate the tracking process. This necessary attachment of sensors damages the unobtrusiveness of visual sensing and causes inconvenience to users and limits application of the method to specific groups of people at some restricted places, where their healthcare and security are concerned, such as inpatients in hospitals and elders in nursing homes. Second, frames might be mislabeled on some occasions which may damage performance of the method. For example, when a subject turns head to attractions around the camera while his body is back to it, video frames recorded during this period are labeled negative by the proposed method and his face may be missed. In another case, when subjects move out of camera FOV while still facing the opposite of camera direction, frames recorded at this moment are labeled positive while in fact the subject is not in them and the tracking analysis over these frames is wasted. Third, the proposed method does not apply to video archives created in the past due to the absence of contextual metadata. A lot of work needs to be done to make the method better. In the future, we plan to explore the possibility of applying more other wearable sensors to the field of content analysis of visual data.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by the Project on the Architecture, Key Technology Research and Demonstration of Web-Based Wireless Ubiquitous Business Environment (no. 2012ZX03005008-001), the National Natural Science Foundation of China (Grant nos. 61202436 and 61271041), Natural Science Foundation of Jiangsu Province, China (Grant no. BK20130164), and EU funded iCore project, “Internet Connected Objects for Reconfigurable Ecosystems.”

References

[1] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: a survey,” ACM Computing Surveys, vol. 38, no. 4, article 13, 2006.
[2] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, “Incremental learning for robust visual tracking,” International Journal of Computer Vision, vol. 77, no. 1–3, pp. 125–141, 2008.
[3] M. -H. Yang, D. J. Kriegman, and N. Ahuja, “ Detecting faces in images: a survey,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, pp. 34–58, 2002.
[4] T. Ahonen, A. Hadid, and M. Pietikäinen, “Face description with local binary patterns: application to face recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 12, pp. 2037–2041, 2006.
[5] W.-H. Liao, “Region description using extended local ternary patterns,” in *Proceedings of the 20th International Conference on Pattern Recognition (ICPR'10)*, pp. 1003–1006, IEEE, August 2010.

[6] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, pp. 886–893, San Diego, Calif, USA, June 2005.

[7] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. 1, pp. 1–511, 2001.

[8] V. Lepetit and P. Fua, *Monocular Model-Based 3D Tracking of Rigid Objects*, Now Publishers Inc., 2005.

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

[10] D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 142–149, 2000.

[11] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. van Gool, “Robust tracking-by-detection using a detector confidence particle filter,” in *Proceedings of the 12th International Conference on Computer Vision (ICCV '09)*, pp. 1515–1522, IEEE, October 2009.

[12] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe, “A boosted particle filter: multitarget detection and tracking,” in *Proceedings of the 8th European Conference on Computer Vision (ECCV '04)*, pp. 28–39, Springer, 2004.

[13] S. Avidan, “Support vector tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064–1072, 2004.

[14] H. Grabner and H. Bischof, “On-line boosting and vision,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, vol. 1, pp. 260–267, New York, NY, USA, June 2006.

[15] S. Avidan, “Ensemble tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261–271, 2007.

[16] “OpenCV,” [http://opencv.org/](http://opencv.org/).

[17] A. Elgammal, C. Muang, and D. Hu, “Skin detection,” in *Encyclopedia of Biometrics*, pp. 1218–1224, Springer, 2009.

[18] Z. Kalal, J. Matas, and K. Mikolajczyk, “P-N learning: bootstrapping binary classifiers by structural constraints,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 49–56, San Francisco, Calif, USA, June 2010.

[19] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-learning-detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[20] G. Nebehay, *Robust Object Tracking Based on Tracking-Learning-Detection*, Faculty of Informatics, Vienna University of Technology, Vienna, Austria, 2012.
