International Journal of Control, Automation, and Systems (2015) 13(3):725-732 ISSN:1598-6446 eISSN:2005-4092
DOI 10.1007/s12555-014-0126-0 http://www.springer.com/12555

Real-time Face Tracking with Instability using a Feature-based Adaptive Model

Vo Quang Nhat, Soo-Hyung Kim, Hyung Jeong Yang, and Gueesang Lee*

Abstract: Unstable object tracking usually happens in hand-held video sequences that consist of the movements of both tracked object and camera. The tracked objects in these sequences are highly unpredictable in displacement and appearance changes, and create more challenge than tracking with a static camera. In this paper, we propose a two-mode tracking model for dealing with unstable situations, such as scaling, pose changes, and abrupt movements. The short-range tracking mode is for the scaling and appearance changes. This mode incorporates Lukas and Kanade's optical flow and the CAMShift, in which to achieve high accuracy, both color and corner point features are fused. The long range-tracking mode utilizes particle filter and CAMShift to capture fast and abrupt motion. We design a mode selection strategy based on a failure detection method for adaptation with each tracking case. The proposed tracking model shows high performance with difficult sequences against the recent tracking systems, as well as achieving real-time processing on smart phones.

Keywords: CAMShift, face tracking, optical flow, particle filter, smart phone.

1. INTRODUCTION

Face tracking on hand-held cameras has different characteristics, compared with tracking with static devices. First, both the motion of the face and the camera have to be considered, which might result in large changes in appearance and scale. Abrupt movements and shaking of the camera also cause blurring with significant displacements of the object. Secondly, there is a need to implement tracking modules on mobile platforms, such as smart phones or digital cameras. However, the lack of computing resources in those platforms makes it difficult to achieve performance for real-time tracking. Tracking algorithms proposed recently could be assigned into two main approaches: the generative, and the discriminative models.

Generative tracking methods [1-5] try to find the most similar region to the target within a local image area. Recent researches have started to apply the sparse representation to visual tracking, by modelling the target appearance using a sparse approximation over a template set [1-4]. The common characteristic of these approaches is to solve an ℓ_1 norm-related minimization problem. Although these trackers give good performance under certain conditions, they are time-consuming, and quite sensitive to changes in appearance.

Discriminative methods try to train a binary classification, which distinguishes the target object from the background [6-8]. They try to extract the feature information of the object model and the background; and then, a binary classification is created, to linearly separate the two groups of features. Different types of features and classifications are proposed. The ensemble tracking [6] uses the local orientation histogram, and pixel colors as object features. The classification is a combination of weak classifiers, trained by AdaBoost. A Gabor filter is applied to extract the texture of tracked object and background regions [7]. While most of the update mechanisms adapt the classifier to the new object appearance, and remove the old one, Kalal *et al*. [8] propose a long term Tracking-Modeling-Detection (TMD) framework that trains the classifier online for all appearances of the object.

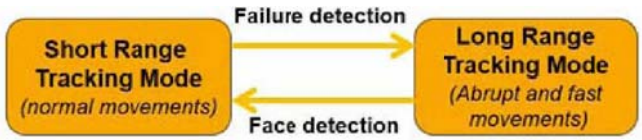
When put into practice, the above algorithms are too heavy to be installed on mobile platforms. From another aspect, they also have problems under instable tracking situations, such as abrupt movements, or significant pose changes. In this paper, an adaptive model for real-time face tracking is proposed to cope with instability of video sequences captured by hand-held cameras. The entire process is shown in Fig. 1. In the proposed tracking system, the face detection method of Viola and Jones [9] is applied to locate the face beforehand. Once the face area is specified, features including corner points and color histogram are extracted. At first, the system is initialized to the short-range tracking mode. For accomplishing tracking accuracy, we incorporate both color and corner point features under the CAMShift [10] and optical flow [11] framework. The tracking model adapts to abrupt movements of the face and the mobile

Manuscript received March 24, 2014; accepted August 28, 2014. Recommended by Associate Editor Myung Geun Chun under the direction of Editor Euntai Kim.

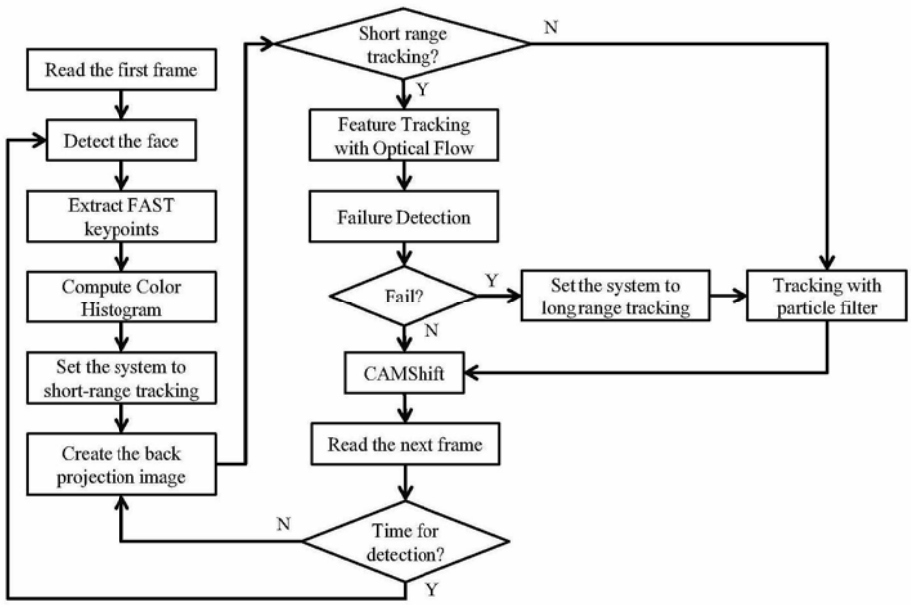
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (2014-024950) and by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (NIPA-2014-H0301-14-1014) supervised by the NIPA (National IT Industry Promotion Agency).

Vo Quang Nhat, Soo-Hyung Kim, Hyung Jeong Yang, and Gueesang Lee are with the Department of Electronics and Computer Engineering, Chonnam National University, 300, Yong-Bong-dong, Kwangju, Korea (e-mails: vqnhat@gmail.com, shkim @jnu.ac.kr, hjyang@chonnam.ac.kr, gslee@jnu.ac.kr).

* Corresponding author.



(a) Two-mode tracking model.



(b) Entire tracking process.

Fig. 1. The proposed tracking model.

camera, by detecting the failure of corner point tracking. Because feature tracking fails under the blur effect and large displacement of the face, the system employs Particle Filter [12] and CAMShift in long-range tracking mode, to keep chasing the face. When the face remains stable in video frames, it is detected to reset the tracking. In both tracking modes, the role of CAMShift is to refine the search position which is given by the particle filter and the feature tracking.

The rest of this paper is organized as follows. First, the collaboration of CAMShift and optical flow for appearance and scale change is presented in Section 2. In Section 3, the failure detection is discussed. In Section 4, the use of the particle filter and CAMShift is presented for tracking abrupt movements. The experimental results are given in Section 5 and finally, Section 6 gives the conclusion.

2. TRACKING IN SHORT-RANGE MODE WITH OPTICAL FLOW AND CAMSHIFT

2.1. Feature tracking with FAST corners

To detect the feature points in the face region quickly, we take advantage of FAST corner detection [13]. As shown in Fig. 2, it will classify pixel p as a corner point, by analyzing a circle of 16 pixels around it. If there are at least n adjacent pixels that have intensities larger or smaller than the intensity of p by a specific threshold T, p is considered as a corner point. Fig. 3 presents key feature points in the detected face region. These points

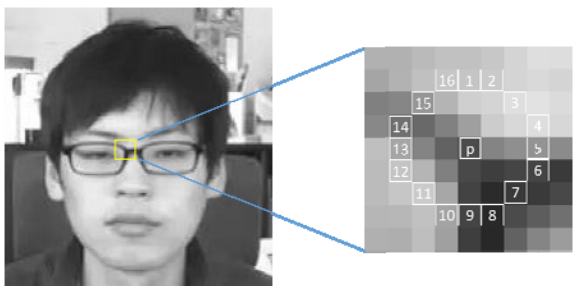


Fig. 2. Example of FAST corner detection method.

(a) Extracted corners in the face region. (b) The divergence problem.

Fig. 3. Feature tracking.

mostly appear at the eyes, the nose, and mouth, and would be tracked in a frame sequence by using optical flow [11].

By computing the distance of all pairs of corner points with the average, the scale is adjusted. Scaling is defined as the ratio of two average distance values at two consecutive frames.

$$s = \frac{avgD_t}{avgD_{t-1}}, \tag{1}$$

$$avgD_t = \frac{2}{N_t(N_t - 1)} \sum_{i=1}^{N_t - 1} \sum_{j=i+1}^{N_t} \left\| p_t^i - p_t^j \right\|, \tag{2}$$

where s is the ratio of scale changes. N_t and $avgD_t$ are the number of tracked corner points and the average distance of all pairs of corner points in frame t, respectively. p_t^i is the coordinate of the i-th feature point. It is assumed that the scale does not change too much between two consecutive frames. Therefore, the tracked window size is updated when the value of s is in the range [0.8, 1.2].

Usually the tracked window is centered at the mean position of the tracked corner points. However, this might cause a divergence problem, as the mean position deviates from the face center due to the unbalanced distribution of corner points. Therefore, in order to further improve the tracking accuracy, CAMShift is applied.

2.2. Color probability distribution image

Color probability distribution image, or the back projection image, represents the probability that a color pixel belongs to the tracked object. This probability is denoted as $P(C \mid O)$, in which C represents the color value, and O is the tracked object. It is modeled from the color histogram of the tracked object region. For creating the color histogram, we could use RGB, the Hue channel in HSV, or the UV channels in YUV color systems. In this paper, the UV color channels in YUV are used, which is the default color system of the Android camera, so the time to convert the entire image to another color system could be saved. Let $\{x_i\}_{i=1..k}$ be the pixels inside the detected face region, and m and n be the number of bins in the U and V axis of the 2D color histogram, respectively. the histogram is construct according to the following equation:

$$H_{uv} = \sum_{i=1}^{k} \delta[C_U(x_i) - u]\delta[C_V(x_i) - v], \quad u = 1...m, \quad v = 1...n, \tag{3}$$

where δ is the Kronecker delta function, and C_U and C_V functions map the color value of the pixel at location x_i, to the corresponding coordinates on the U and V axes, respectively.

Using the 2D histogram, the probability value is computed, and scaled to the range of [0, 255].

$$P(C_{uv} \mid O) = \frac{H_{uv}}{\max(H)}, \tag{4}$$

$$p_{uv} = P(C_{uv} \mid O) * 255, \quad u = 1...m, \quad v = 1...n, \tag{5}$$

where H is the 2D histogram, and H_{uv} is the histogram value of a pixel color C_{uv}.

Up to this time, we only consider the color histogram of the face region. In practice, the background region could have colors that are similar to colors in the face area. Therefore, this creates noise in the probability distribution image, and affects the efficiency, when apply tracking algorithms in the next steps. To create a better distribution image for tracking, the color background modeling [14] is applied, to stress the color value that appears in the face, but not in the background; and suppress the color features that appear both in the face, and the background. A new probability function is computed as:

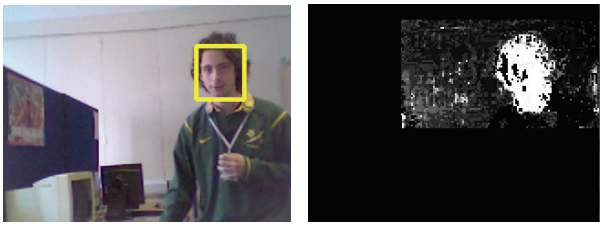


Fig. 4. Color probability distribution image created in the region around tracked window.

$$P(O \mid C) = \frac{P(C \mid O)P(O)}{P(C \mid O)P(O) + P(C \mid B)P(B)}, \tag{6}$$

where $P(C \mid B)$ and $P(C \mid O)$ are the color models of the tracked object and the background, respectively. $P(O)$ and $P(B)$ are computed by the ratio of the object and the background region. The probability is scaled to range [0, 255] in the back projection image.

$$p'_{uv} = P(O \mid C_{uv}) * 255 \tag{7}$$

Only the back projection image in a region surrounding the current tracked window is needed for tracking. This is based on the observation that the object locations between two consecutive frames are not too far from each other. This approach could reduce the effect of background colors and it can speed up the computation. Fig. 4 shows a color probability distribution image computed by (3).

2.3. Refinement with CAMShift

CAMShift is used to shift the search window of the previous frame, to the centroid of the distribution in the current frame. The summation of probability values of all pixels in the search window will decide the size of the window. CAMShift uses the first and the zero-th moment of the distribution image, to estimate the shift vector.

$$x_c = M_{10} / M_{00}, \quad y_c = M_{01} / M_{00}, \tag{8}$$

where

$$M_{00} = \sum_x \sum_y I(x,y); \quad M_{10} = \sum_x \sum_y xI(x,y);$$
$$M_{01} = \sum_x \sum_y yI(x,y),$$

where I is the back projection image. The mean position of corner points tracked by optical flow is computed, and set as the center of the initial search window for CAMShift. The collaborative method can be summarized in the following steps:

1) Input the color probability distribution image created in Section 2.2.
2) Calculate the mean position of corner points tracked by optical flow, and set it as the initial location for tracking.

$$m_x = \frac{\sum_{i=1}^{N} x_i}{N}, \quad m_y = \frac{\sum_{i=1}^{N} y_i}{N}, \tag{9}$$

where N is the number of corner points, and (x_i, y_i) is the coordinate of the i-th corner point.
3) Compute the center of the tracked window, using (5).
4) Center the tracked window in the position obtained from step 3, and iterate step 3 until convergence.

When feature tracking is used, we only need to repeat the CAMShift operator a small number of times with the support of tracked corner points. Moreover, the combination of CAMShift and optical flow will prevent the search window from moving to other regions of similar color to the face, such as the neck or background.

3. TRACKING MODE ADAPTATION

The corner points in the face region may move to wrong positions under abrupt movements, hence, it can create a bad initial search position for CAMShift. In order to remove the failed tracked points, the error of local patches is measured in two consecutive frames. The sum of square differences [15] is used in three color channels of the YUV color system:

$$SSD_Y(p^t) = \sum_{m,n}[Y_{t-1}(p_x^{t-1}+m, p_y^{t-1}+n) - Y_t(p_x^t+m, p_y^t+n)]^2, \tag{10}$$

$$SSD_U(p^t) = \sum_{m,n}[U_{t-1}(p_x^{t-1}+m, p_y^{t-1}+n) - U_t(p_x^t+m, p_y^t+n)]^2, \tag{11}$$

$$SSD_V(p^t) = \sum_{m,n}[V_{t-1}(p_x^{t-1}+m, p_y^{t-1}+n) - V_t(p_x^t+m, p_y^t+n)]^2, \tag{12}$$

in which, p^t is the coordinate of a corner point in the tracked image at time t, and Y_t, U_t, and V_t are its color channel images. Finally, the error function is computed, as follows:

$$SSD^* = \alpha SSD_Y + \beta SSD_U + \gamma SSD_V, \quad \alpha+\beta+\gamma = 1 \wedge \alpha,\beta,\gamma > 0. \tag{13}$$

Tracked corner points with image patches having an *SSD** value that surpasses a specific threshold would be eliminated from the tracking list. A further failure checking is performed by employing backward tracking. The corner point p^t in frame t is tracked backward to frame t-1, to yield the point q^{t-1}. The forward tracking of p^t is considered successful, if the distance between two points is smaller than a specific threshold:

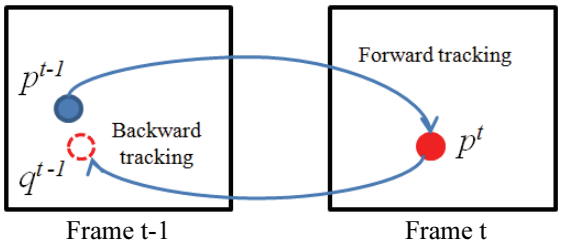


Fig. 5. Forward and backward feature tracking for failure detection.

$$\left\| p^{t-1} - q^{t-1} \right\| < \lambda. \tag{14}$$

Let N_t denote the number of corner points remaining at image I_t. The failure is detected, if:

$$\begin{cases} N_t < T \\ or \quad N_t < N_{t-1}/M, \end{cases} \tag{15}$$

where T is a specific threshold, and M is a defined ratio. Therefore, if the number of successful tracked points is too small, or it decreases too fast, the tracking is considered as a failure.

4. TRACKING IN LONG-RANGE MODE

If the feature tracking fails, the initial position of CAMShift is computed by particle filtering, which is better for tracking fast movements. In this paper, the particle filter [12] is used to capture the region of the rapidly moving face. To chase the face area, face detection is applied in a sub window around it to refine the search position, or to reset the tracking process.

Particle filter describes the tracked object as a state vector X_t, and the observation as a vector Z_t. It approximates the probability distribution by a weighted sample set $S = \{(s^i, w^i) \mid i = 1...N\}$, in which each sample represents one hypothetical state of the object, with a sampling weight w [12]. Particle filter consists of three main steps: prediction, weight update, and resample. The prediction step, denoted as the probability $P(X_t \mid Y_{t-1})$, propagates each sample according to a system model. Then, the weight of each sample is updated, based on the observation Z_t, and the likelihood: $w^i = P(Z_t \mid X_t = s^i_t)$. The mean state of an object is acquired by

$$E(S) = \sum_{i=1}^{N} w^i s^i. \tag{16}$$

Finally, the resample step will be applied to eliminate small weight samples, and multiply samples with large weight. Each sample is presented by a rectangle with position (x^i, y^i), and fixed size:

$$S = \{(s^i = (x^i, y^i, w, h), w^i) \mid i = 1...N\}. \tag{17}$$

In this paper, a fast particle weight computation method is proposed, which makes use of the probability distribution image I in Section 2.2. The particle weights are computed as below:

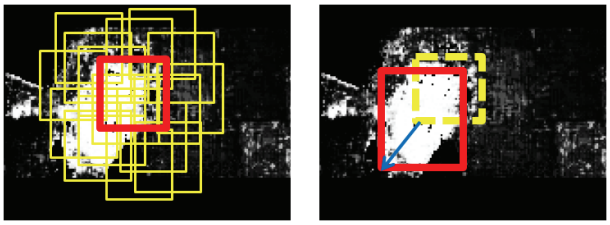


Fig. 6. Combining particle filtering and CAMShift. In the left side, the particle filter is used to get the mean window position. In the right side, the tracked window is shifted to the face region.

$$w^i = \sum_{x=x^i}^{x^i+w} \sum_{y=y^i}^{y^i+h} I(x,y). \tag{18}$$

The mean state or the location of the face is then taken:

$$x_c = \frac{\sum_{i=1}^{N} w^i x^i}{\sum_{i=1}^{N} w^i}; \quad y_c = \frac{\sum_{i=1}^{N} w^i y^i}{\sum_{i=1}^{N} w^i}. \tag{19}$$

The sample weight is a summation of the values in the image region. Therefore, the weight values could be obtained quickly, by creating the integral image of the color probability distribution image. Similar to the combination of optical flow and CAMShift, the mean location (x_c, y_c) acquired in equation (15) is set as the initial region for CAMShift. Fig. 6 shows a demonstration of using Particle filter for tracking, and CAMShift for refining.

Based on the zero-th moment of the distribution image, the scaling is adjusted:

$$s = \alpha\sqrt{M_{00}/256}, \tag{20}$$

where α is the parameter for controlling the scale. If the displacement of two tracked windows in two successive frames is smaller to a specific distance, the face is detected again in long range tracking mode, after which the tracking process is initialized.

5. EXPERIMENTS

5.1. Experimental construction

The proposed tracking system is implemented in a Samsung Galaxy SII Android 4.0 Smartphone, which has an 8 megapixel camera, and a 1.2 GHz dual core ARM Cortex-A9 processor. The application interface for displaying results and capturing camera images is programmed in Java. The tracking system is developed with Android NDK for a better performance. An image sequence with size 640x480 is extracted from the input video, with above 30 FPS.

The proposed method is compared with three latest state-of-the-art trackers, named L1 [1], Sparsity-based Collaborative Model (SCM) [4] and the TMD [8]. Four video sequences are tested. In each sequence, the error is measured using the Euclidian distance of the two center points of the tracked window, and the ground truth. The result shows that our method is more accurate than L1, TMD, or SCM trackers, in terms of appearance changes, and adapts better to abrupt movements.

Table 1. Tested video sequences.

Sequences	Features	Frame length	Frame size
yawbitch	Large pose changes	317	640x480
david	Global illumination changes	766	320x240
Motinas	Fast and abitrary displacement of the face, out of frame movement	447	320x240
fast movement	Abrupt movement of the face and camera	616	640x480

5.2. Tracking with appearance changes

For demonstrating the performance of our proposed model with appearance variation, we perform the tracking with two video sequences. In the first sequence, we test the tracking with appearance changes of the rotated face. The L1 tracker loses the face after frame 71. The TMD and SCM trackers have a divergence problem, when the face appearance varies. However, our proposed method could track the face well.

The appearance changes due to the variation of luminance are tested in the *david* sequence studied in [16]. TMD tracker and our method give good results. However, L1 and SCM trackers fail, under the variation of luminance and pose. The tracking results of several frames are shown in Figs. 7 and 8.

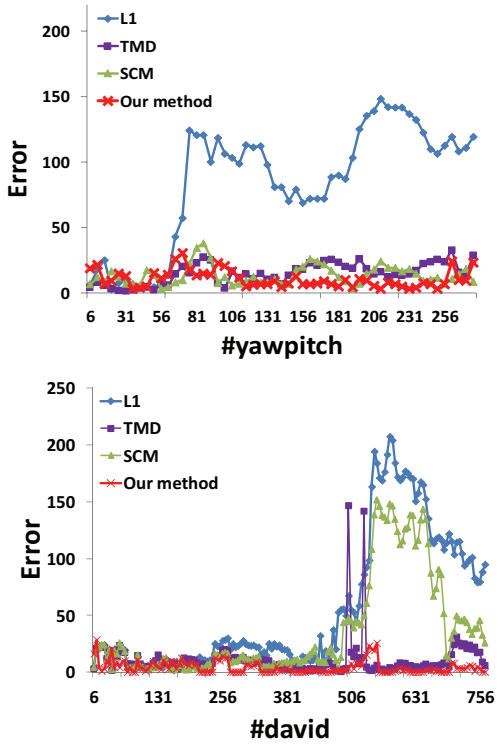
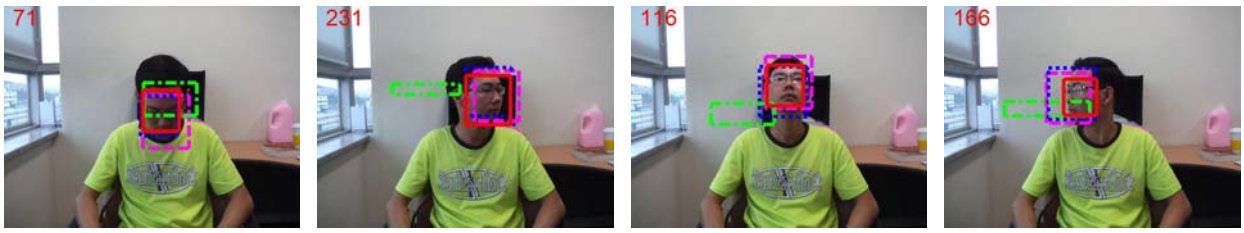
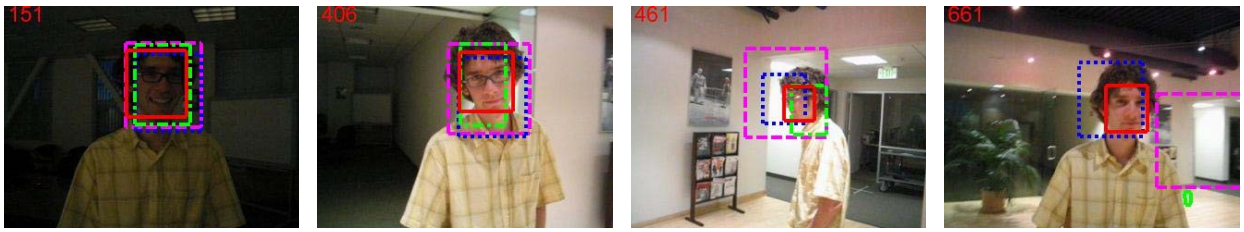


Fig. 7. The tracking error of *yawpitch* and *david* sequence.



(a) Tracking with *yawpitch* sequence.



(b) Tracking with *david* sequence.

ℓ_1 TMD SCM Our method

Fig. 8. Tracking with appearance changes.

5.3. Tracking with abrupt movements

In this section, the adaptation of our model to abrupt movements is described. In the *motinas_emilio_webcam* sequence [17], the tracked face moves fast, and arbitrarily. In addition, the face sometime disappears from the video frame. L1 and SCM methods miss the tracking from the first frames, and after that, cannot reinitialize the tracking. The TMD method has large displacement of the tracking window, because of the wrong detection in the background, when the face moves out of the video frame. Our method can follow the face, and gives the best average tracking error.

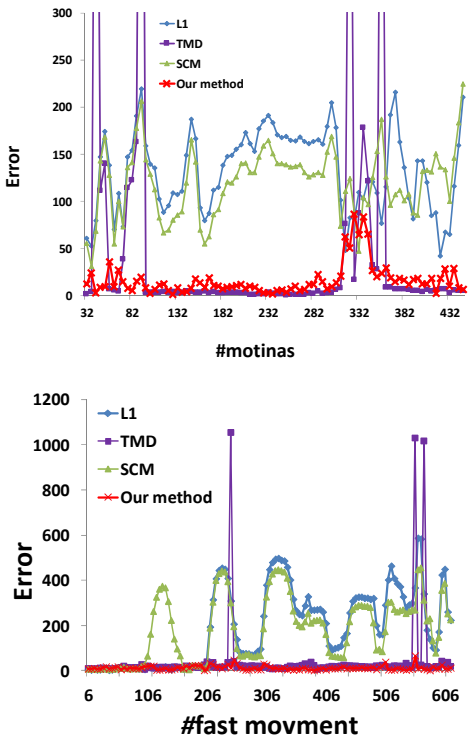


Fig. 9. The tracking error of *motinas* and *fast_movment* movement sequence.

The second sequence includes motion of the tracked face and the smart phone camera. This situation creates large motion and image blur. As we can see in Fig. 10, our method could capture fast motion, while the L1, SCM and TMD trackers miss the face, due to large displacements, and the blur effect.

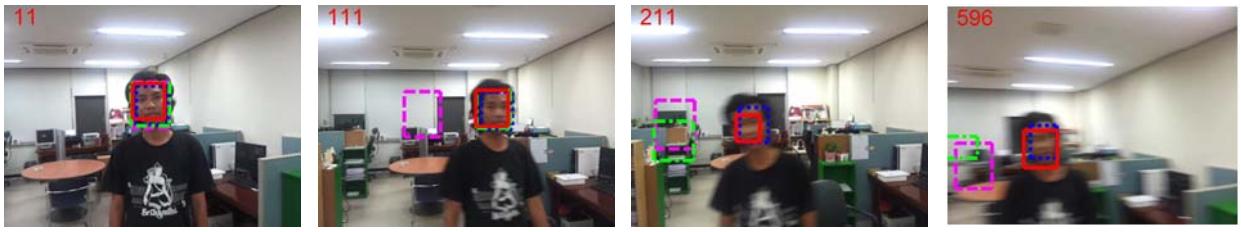
5.4. Performance of tracking modes

Our system implemented on Samsung Galaxy SII can process the tracking video at the speed of 30 FPS. In order to demonstrate more clearly the computation time as well as the order of tracking modes, we observe the *fast movement* sequence which includes both the normal and abrupt moments. Fig. 11 shows the distance of the tracked face between two consecutive frames in the *fast movement* sequence based on the ground truth data. Besides that, the amount of tracking time, and the corresponding tracking modes are displayed in each frame. We can see that the face is detected at frame 270 when the movement is stable and the tracking mode is switched from long-range to short-range. The system also reinitialize the face location after some period of time (each 30 frames in our experiment) at frames having small displacement. Although the detection step has most time-consuming, it doesn't affect the performance of tracking due to the limited number of trigger time.

In most of time, the short-range tracking mode can only deal with an acceptable shifting between frames. However, we observe that, in a special case, the short-range tracking mode can track the face at frame 505 with large displacements. In this case, because the tracking can maintains the facial feature points, the tracking mode is unchanged. With the large movements, the long range tracking mode is employed.

Depending on the size of tracked face and the number of detected feature points, the computation time may vary. In general, the average tracking time of short-range tracking mode is lower than the long-range tracking mode as shown in Fig. 11. The reason is that the long-range tracking incorporates the particle filter framework

(a) Tracking with *motinas_emilio_webcam* sequence.



(b) Tracking with *fast movement* sequence.

Fig. 10. Tracking with abrupt movements.

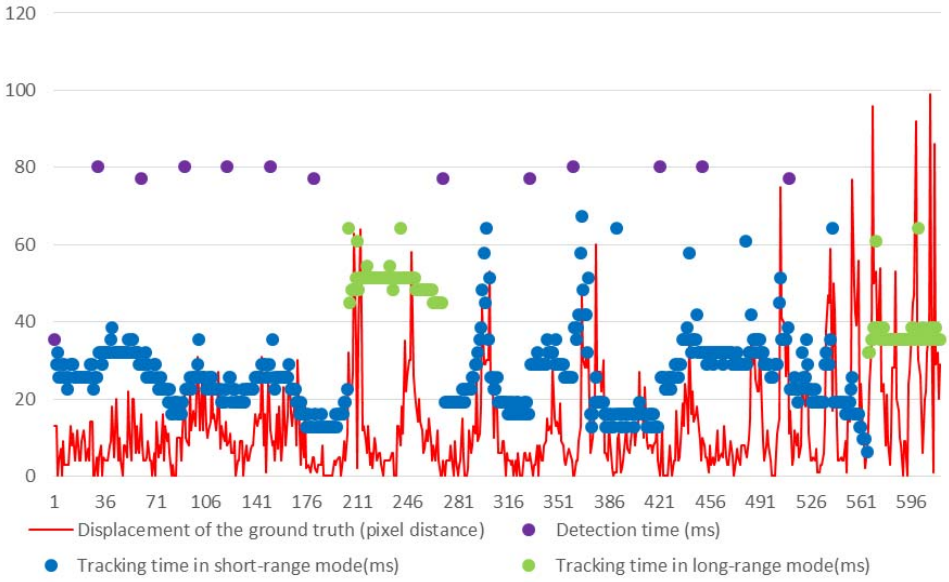


Fig. 11. Computation time of face detection, short-range tracking, and long-range tracking with the displacement of ground truth in each frame of the *fast movement* sequence.

which has larger search range in order to deal with abrupt and fast movements.

6. CONCLUSION

In this paper, we presented a face tracking system that is accurate, fast, and easy to implement. The proposed system utilizes both corner points and color features of the tracking framework of optical flow. Also, CAMShift and optical flow are used to create an adaptive model for unstable tracking situations. The implementation in a mobile device showed attractive performance, with the feasibility of the tracking system for future applications. However, the proposed method currently works only with a single face. Future research may include the issue of real-time multiple face tracking, which could be far more challenging.

REFERENCES

[1] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1830-1837, 2012.

[2] X. Mei and H. Ling, "Robust visual tracking using L1 minimization," *Proc. of IEEE International Conference on Computer Vision*, pp. 1436-1443, 2009.

[3] B. Liu, J. Huang, L. Yang, and C. Kulikowski, "Robust tracking using local sparse appearance model and k-selection," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1313-1320, 2011.

[4] W. Zhong, H. Lu, and M. H. Yang, "Robust object tracking via sparsity-based collaborative model,"

Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1838-1845, 2012.
[5] M. Qing and K. H. Jo, "A novel particle filter implementation for a multiple-vehicle detection and tracking system using tail light segmentation," *International Journal of Control, Automation, and Systems*, vol. 11, no. 3, pp. 577-585, 2013.
[6] S. Avidan, "Ensemble tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261-271, 2007.
[7] H. Nguyen and A. Smeulders, "Tracking aspects of the foreground against the background," *Proc. of IEEE European Conference on Computer Vision*, pp. 446-456, 2004.
[8] Z. Kalal, J. Matas, and K. Mikolajczyk, "Online learning of robust object detectors during unstable tracking," *Proc. of IEEE International Conf. on Computer Vision Workshops*, pp. 1417-1424, 2009.
[9] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple," *Proc. of EEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 511-518, 2001.
[10] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, vol. 2, no. 2, pp. 13-27, 1998.
[11] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proc. of the 7th International Joint Conference on Artificial Intelligence*, pp. 674-679, 1981.
[12] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. on Signal Processing*, vol. 50, no. 2, pp. 174-188, 2002.
[13] M. Pietikäinen and G. Zhao, "Local texture descriptors in computer vision," *Proc. of IEEE International Conference on Computer Vision*, *Tutorial*, 2009.
[14] R. Stolkin, I. Florescu, and G. Kamberov, "An adaptive background model for camshift tracking with a moving camera," *Proc. of International Conference on Advances in Pattern Recognition*, pp. 147-151, 2007.
[15] K. Nickels and S. Hutchinson, "Estimating uncertainty in SSD-based feature tracking," *Image and Vision Computing*, vol. 20, no. 1, pp. 47-58, 2002.
[16] D. 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, 2007.
[17] Surveillance Performance Evaluation Initiative (SPEVI), Single Face Datasets. http://www.eecs.qmul.ac.uk/~andrea/spevi.html

Vo Quang Nhat received his B.S. degree in Information Technology from the University of Science Ho Chi Minh City, Vietnam in 2010, and his M.S. degree in Electronics and Computer Engineering from the Chonnam National University, Korea in 2013. He is currently a Ph.D. student in the department of Chonnam National University, Korea. His interesting studies are in multimedia and image processing, vision tracking, and pattern recognition.

Soo-Hyung Kim received his B.S. degree in Computer Engineering from Seoul National University in 1986, and his M.S. and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology, in 1988 and 1993, respectively. From 1990 to 1996, he was a senior member of research staff in Multimedia Research Center of Samsung Electronics Co., Korea. Since 1997, he has been a professor in the Department of Computer Science, Chonnam National University, Korea. His research interests are pattern recognition, document image processing, medical image processing, and ubiquitous computing

Hyung Jeong Yang received her B.S., M.S. and Ph.D. from Chonbuk National University, Korea. She is currently an associate professor at the Deprtment. of Electronics and Computer Engineering, Chonnam National University, Gwangju, Korea. Her main research interests include multimedia data mining, pattern recognition, artificial intelligence, e-Learning, and e-Design.

Gueesang Lee received his B.S. degree in Electrical Engineering and his M.S. degree in Computer Engineering from Seoul National University, Korea, in 1980 and 1982, respectively. He received his Ph.D. degree in Computer Science from Pennsylvania State University in 1991. He is currently a professor of the Department of Electronics and Computer Engineering in Chonnam National University, Korea. His research interests are mainly in the field of image processing, computer vision and video technology.