A Feature-Based Adaptive Model for Realtime Face Tracking on Smart Phones

Quang Nhat Vo and GueeSang Lee*

Department of Electronics and Computer Engineering,
Chonnam National University, Kwangju, South Korea
vqnhat@gmail.com, gslee@chonnam.ac.kr

Abstract. We propose a face tracking method that is extremely fast and applicable to implementation on smart phones. An adaptive model is built to make the tracking robust with variety of situation such as scaling, pose changes, and abrupt movements. For dealing with the scaling and appearance changes, we incorporate the Lukas and Kanade's optical flow and the CAMShift in which both color and corner point features are used to achieve the high accuracy. Based on the feature tracking, the model adapts with abrupt movements of tracked face and mobile camera by a failure detection method. The system then utilizes the particle filter and CAMShift to catch up the fast motion. Based on the tracked region, we detect the face in order to reinitialize the tracking process. The proposed method shows the high performance against the recent ones as well as achieving the realtime requirement on smart phones.

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

1 Introduction

Tracking on mobile devices has specific characteristics comparing with tracking on personal computers. Firstly, we have to consider both the motion of the object and the camera which might cause large changes in appearance and scale. On the other hand, abrupt movements and shaking of the camera also create the blur effect and significant displacements of the object. Secondly, the lack of computing resource makes the tracking difficult to achieve the realtime performance. There are a lot of tracking algorithms proposed in recent time. We could assign them to two main approaches, the generative and the discriminative models.

In the first group, generative tracking methods try to find the most similar region to the target within a local image area. Recent researches start to apply the sparse representation to visual tracker 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 a good performance under some conditions, they are time consumption and quite sensitive to appearance changes.

* Corresponding author.

J.-K. Kämäräinen and M. Koskela (Eds.): SCIA 2013, LNCS 7944, pp. 630–639, 2013.
© Springer-Verlag Berlin Heidelberg 2013

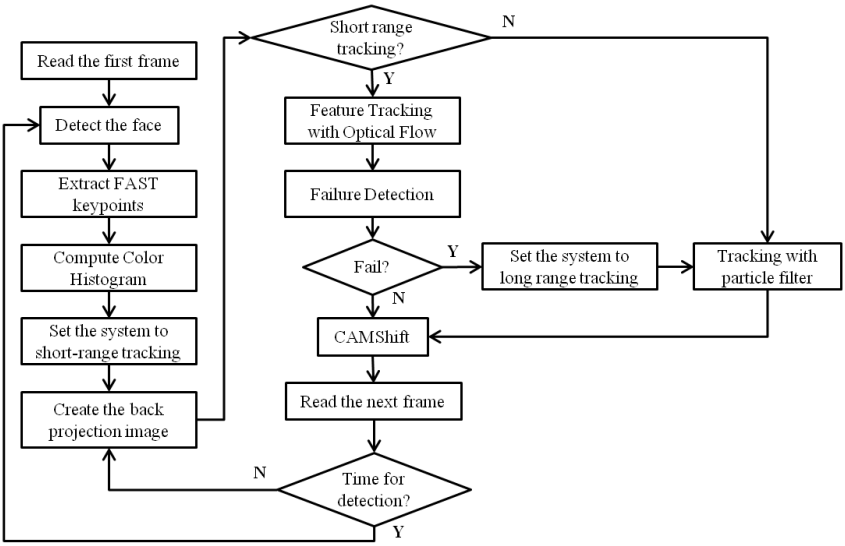


Fig. 1. Our proposed tracking model

In the second group, the discriminative methods try to train a binary classification which distinguishes the target object from the background [5-7]. Their similar point is that 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 feature. Different types of feature and classification are proposed. Gabor filter is applied for extracting the texture of tracked object and background regions [6]. The ensemble tracking [5] uses the local orientation histogram and pixel colors as object features. The classification is a combination of weak classifiers trained by AdaBoost. While most of update mechanisms adapt the classifier to the new object appearance and remove the old one, Kalal et al [7] proposes a long term Tracking-Modeling-Detection (TMD) framework that online trains the classifier for all appearance of the object.

When put into practice, the above algorithms are too heavy for installing on mobile devices. In the other aspect, they also have problems under some tracking situations such as abrupt movements and significant pose changes. In this research, we propose an adaptive model for realtime face tracking on smart phones. The entire process is illustrated in the Fig. 1. In our tracking system, we apply the face detection method of Viola and Jone [11] to locate the face beforehand. As long as the face area is specified, features including corner points and color histogram are extracted. At the first time, 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 [8] and optical flow [9] framework. The tracking model adapts with abrupt movements of the face and the mobile camera by detecting the failure of corner point tracking. Because the feature tracking fails under the blur effect and large displacement of the face, the system employs the particle filter [10] and CAMShift for keep chasing the face. As long as the face is stable in video frames, it is detected to reset

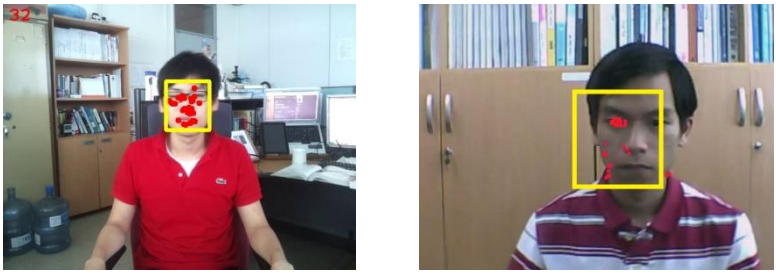
the tracking. In both tracking modes, the role of CAMShift is to refine the search position which is given by particle filter and the feature tracking.

The rest of this paper is organized as follows. We first present the collaboration of CAMShift and optical flow for appearance and scale change in Section 2. In Section 3, the failure detection method is described. In Section 4, we present the using of particle filter and CAMShift for tracking the abrupt movements. The experiment results are explained in Section 5. Finally, section 6 gives the conclusion.

2 Face Tracking with Optical Flow and CAMShift

2.1 FAST Corner Feature Tracking

For the purpose of quickly detecting the feature points in the face region, we take the advantage of FAST corner detection [12]. As showed in Fig. 2a, these points mostly appear at the eyes, the nose, and the mouth and would be tracked between frames by using optical flow method [9].



(a) (b)

Fig. 2. Feature tracking. (a) extracted corners in the face region (b) the divergence problem.

Based on tracked corner points, we adjust the scale by computing the distance of all pairs of these points and get the average value. Scaling is defined by 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 i-th feature point. We assume that the scale does not change too much between two consecutive frames. Therefore, we only update the tracked window size when the value of s is in the range [0.8, 1.2].

The tracked window could be centered at the mean position of tracked corner points. However, it might cause the divergence problem as the mean position deviates from the face center because of the unbalanced distribution of corner points. Therefore, we apply CAMShift to further improve the tracking accuracy.

2.2 Color Back Projection Image

Color back projection image presents the probability that a color pixel belongs to the tracked object. We apply the color background modeling technique [13] to stress the color values which appear in the object but not in the background, and suppress the color values which appear both in the object and the background. A probability function was proposed:

$$P(O \mid C) = \frac{P(C \mid O)P(O)}{P(C \mid O)P(O) + P(C \mid B)P(B)} \tag{3}$$

where $P(C|B)$ and $P(C|O)$ are the color model of the tracked object and the background. $P(O)$ 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.

Color model is built from the color histogram of object and background region. We choose the *UV* color channels in *YUV* color system to reduce the impact of luminance changes. On the other hand, *YUV* is the default color system of Android camera so that it does not take time for color conversion. The probability value of a specific pixel color C_{uv} is computed.

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

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

Fig. 3. Color Probability Distribution Image created in the region around tracked window

For tracking, we only need to create the back projection image in a region surrounding current tracked window. This is based on the assumption 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 computation time. Fig.3 presents an example of color probability distribution image computed by equation (3).

2.3 Tracking Refinement with CAMShift

The idea of CAMShift is to shift the search window in previous frame to the centroid of the distribution in next frame. The summation of probability value 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.

$$x_c = M_{10} / M_{00} \ , \ y_c = M_{01} / M_{00} \tag{5}$$

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

with I is the back projection image. We utilize the result of feature tracking. 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 with 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.

$$m_x = \frac{\sum_{i=1}^{N} x_i}{N} \ , \ m_y = \frac{\sum_{i=1}^{N} y_i}{N} \tag{6}$$

   with N is the number of corner points and (x_i, y_i) is the coordinate of i-th corner point.
3. Find the center of the tracked window with equation (5).
4. Center the tracked window with the position calculated from step 3 and iterate step 3 until convergence or reaching a small number of iterations.

With the support of tracked corner points, we only need to repeat the CAMShift operator in a small number of times. Moreover, the combination of CAMShift and optical flow will prevent the search window move to the other regions such as the neck or background which has the similar color with the face.

3 Failure Detection

Under rapid movements, the corner points in the face region will be moved to the wrong locations and, hence, create the bad initial search position for CAMShift. In order to remove the failed tracked points, we measure the error of local patches in two consecutive frames. The sum of square differences [14] is used in three color channels of the YUV color system:

$$SSD_Y(p^t) = \sum_{m,n} [Y_{t-1}(p_x^{t-1} + m, \ p_y^{t-1} + n) - Y_t(p_x^t + m, \ p_y^t + n)]^2 \tag{7}$$

$$SSD_U(p^t) = \sum_{m,n} [U_{t-1}(p_x^{t-1} + m,\ p_y^{t-1} + n) - U_t(p_x^t + m,\ p_y^t + n)]^2 \tag{8}$$

$$SSD_V(p^t) = \sum_{m,n} [V_{t-1}(p_x^{t-1} + m,\ p_y^{t-1} + n) - V_t(p_x^t + m,\ p_y^t + n)]^2 \tag{9}$$

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

$$\begin{aligned} &SSD^* = \alpha SSD_Y + \beta SSD_U + \gamma SSD_V \\ &\alpha + \beta + \gamma = 1 \wedge \alpha, \beta, \gamma > 0 \end{aligned} \tag{10}$$

Tracked corner points with image patches having the SSD^* value which surpasses a specific threshold would be eliminated from the tracking list. We perform a further failure checking by employing the 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:

$$\left\| p^{t-1} - q^{t-1} \right\| < \lambda \tag{11}$$

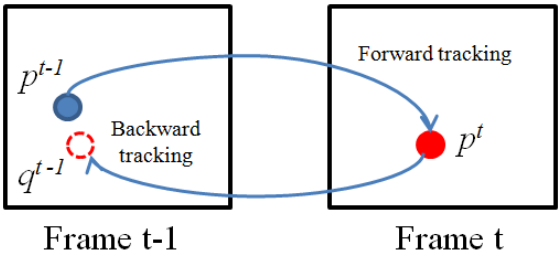


Fig. 4. Forward and Backward Feature Tracking for Failure Detection

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

$$\begin{cases} N_t < T \\ or\ N_t < N_{t-1} / M \end{cases} \tag{12}$$

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

4 Face Tracking with Particle Filter and CAMShift

When the feature tracking fails, the initial position of CAMShift is computed by particle filter method which is better in tracking fast movements. In our work, particle filter [10] is used to catch up the region of the face moving rapidly. By chasing the face area, face detection could apply within a sub window around it to refine the position and reset the tracking process. In our work, each sample is presented by a rectangle with position (x^i, y^i) and fixed size:

$$S = \{ (s^i = (x^i, y^i, w, h), w^i) \mid i = 1...N \} \tag{13}$$

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

$$w^i = \sum_{x=x^i}^{x^i+w} \sum_{y=y^i}^{y^i+h} I(x, y) \tag{14}$$

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

$$x_c = \frac{\sum_{i=1}^{N} w^i x^i}{\sum_{i=1}^{N} w^i}; \; y_c = \frac{\sum_{i=1}^{N} w^i y^i}{\sum_{i=1}^{N} w^i} \tag{15}$$

Each sample weight is a summation of value in an image region. Therefore, the weight values could be taken quickly by creating the integral image of the color probability distribution image. Similar with 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. 5 shows a demonstration of using particle filter for tracking and CAMShift for refining.

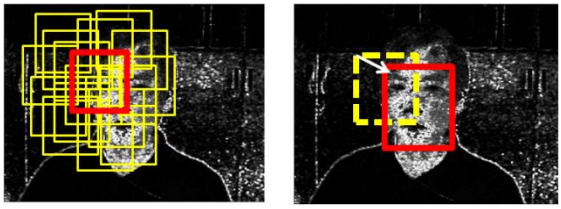


Fig. 5. The combination of particle and CAMShift. In the left side, particle filter is used to get the mean window position. In the right side, the tracked window is shifted to the face region.

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

$$s = \alpha \sqrt{M_{00} / 256} \tag{16}$$

with α is the parameter for controlling the scale. In long range tracking mode, the face is detected again if the displacement of two tracked windows in two successive frames is smaller than a specific distance. After that, we initialize the tracking process.

5 Experiments

We implement our tracking system with Samsung Galaxy SII Android 4.0 Smartphone. This phone 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 on Java. The tracking system is developed with Android NDK to achieve better performance. Image sequence with size 640x480 is extracted and tracked continuously. Our method could achieve above 30 FPS.

We compare our method with three latest state-of-the-art trackers named L1 [1], the TMD [7], and Sparsity-based Collaborative Model (SCM) [4]. Three video sequences are tested. In each sequences, the error is measured using the Euclidian distance of two center points of tracked window and the ground truth. The result shows that our method is more accurate than L1, TMD, and SCM trackers in term of appearance changes and adapts better with abrupt movements.

The tracking results of several frames are showed in Fig. 7. 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 the divergence problem when the face appearance varies. However, our proposed method could track the face well. The second sequence tests the tracking with abrupt movements. It includes the motion of tracked face and the smart phone camera. As we can see in Fig. 7, our method could catch up the fast motion while the L1, SCM and TMD trackers miss the face due to large displacements and the blur effect. The translation, scaling and luminance changes are tested in the *david* sequence studied in [15]. TMD tracker and our method give good results. However, L1 and SCM trackers fail under the varying of luminance and pose.

Table 1. The average tracking errors

	L1 tracker	TMD tracker	SCM tracker	Our method
yawpitch	84.44	14.72	13.39	**10.90**
fast movement	192.49	42.34	182.70	**12.06**
david	55.62	10.63	36.00	**4.25**

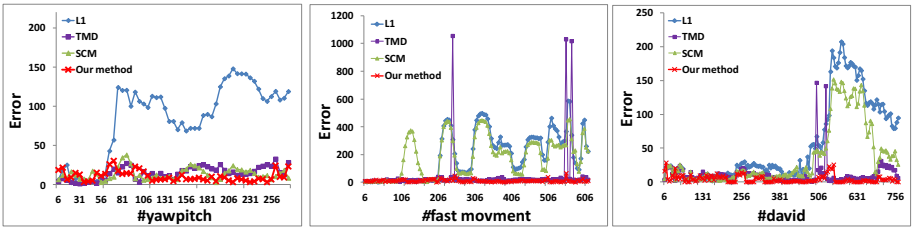
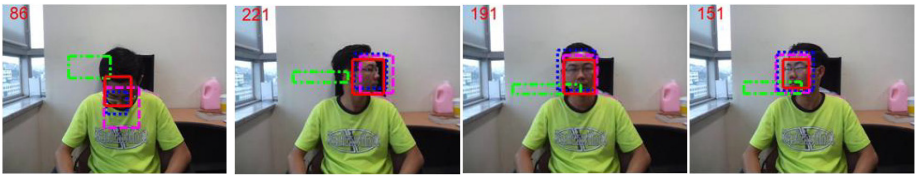


Fig. 6. The tracking error of each sequence



(a) Tracking with *yawpitch* sequence.

(b) Tracking with *fast movment* sequence.

(c) Tracking with *david* sequence.

Fig. 7. Tracking results of tested algorithms on three video sequences

6 Conclusion

This paper presents a face tracking system that is accuracy, fast, and easy to implement. Our system utilizes both keypoints and color feature on the tracking framework of optical flow, CAMShift, and particle flow to get a better tracking result. The implementation in Samsung Android smart phone shows an attractive performance and the feasibility of our tracking system for future applications. However, our method currently only works with single face. Our future research is to extend our method to realtime multiple faces tracking which would be a very challenging problem.

Acknowledgements. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technologies(2012-047759) and the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2013-H0301-13-3005).

References

1. Bao, C., Wu, Y., Ling, H., Ji, H.: Real time robust L1 tracker using accelerated proximal gradient approach. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1830–1837 (2012)
2. Mei, X., Ling, H.: Robust visual tracking using ℓ1 minimization. In: ICCV, pp. 1436–1443 (2009)
3. Liu, B., Huang, J., Yang, L., Kulikowsk, C.: Robust tracking using local sparse appearance model and k-selection. In: CVPR, pp. 1313–1320 (2011)
4. Zhong, W., Lu, H., Yang, M.H.: Robust Object Tracking via Sparsity-based Collaborative Model. In: CVPR, pp. 1838–1845 (2012)
5. Avidan, S.: Ensemble tracking. IEEE Trans. PAMI 29(2), 261–271 (2007)
6. Nguyen, H.T., Smeulders, A.W.M.: Tracking aspects of the foreground against the background. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3022, pp. 446–456. Springer, Heidelberg (2004)
7. Kalal, Z., Matas, J., Mikolajczyk, K.: Online learning of robust object detectors during unstable tracking. In: IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1417–1424 (2009)
8. Bradski, G.R.: Computer Vision Face Tracking for Use in a Perceptual User Interface. Intel. Technology Journal 2(2), 13–27 (1998)
9. Lucas, B.D., Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI 1981), pp. 674–679 (1981)
10. Sanjeev Arulampalam, M., Maskell, S., Gordon, N., Clapp, T.: A Tutorial on Particle Filters for OnlineNonlinear/Non-Gaussian Bayesian Tracking. IEEE Transactions on Signal Processing, 174–188 (2002)
11. Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple. In: Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 511–518 (2001)
12. Pietikäinen, M., Zhao, G.: Local Texture Descriptors in Computer Vision (presentation). In: ICCV (2009)
13. Stolkin, R., Florescu, I., Kamberov, G.: An adaptive background model for camshift tracking with a moving camera. In: Proc. International Conference on Advances in Pattern Recognition, pp. 147–151 (2007)
14. Nickels, K., Hutchinson, S.: Estimating uncertainty in SSD-based feature tracking. IVC 20(1), 47–58 (2002)
15. Ross, D., Lim, J., Lin, R.S., Yang, M.H.: Incremental Learning for Robust Visual Tracking. The International Journal of Computer Vision, Special Issue: Learning for Vision (2007)