Face Tracking in Video Using an Integrated Color-based and Moment-based Particle Filter

Report Author	Group Partner
Tianxiao Zhao	Feiyang Liu
tzh@kth.se	liuf@kth.se

Abstract

In this project, we achieve in face tracking in video using an integrated particle filter which combines a color-based observation model and a moment-based observation model. We first utilize a Haar feature-based cascade classifier to acquire the initial state of the face target, and then generate particles, propagate them and measure their similarities with the target for both the color-based and moment-based model. Systematic resampling is applied to select particles with replacement based on their weights. Finally, a final estimation of the face target is obtained by fusing the posterior states from the color-based and moment-based observation models. Experimental results of videos show that this integrated particle filter works well in the single face case and outperforms the trackers with a single observation model.

1. Introduction

The task of face tracking basically focuses on how to locate a face or several faces in front of a camera in real-time. In recent years, research work in this field is booming as face tracking is increasingly applied in a wide range of applications, such as mobile games, video surveillance and smart rooms [1-2]. Besides, a number of projects related to face tracking are started and proceeded by many companies, such as Microsoft, Tobii and Adobe.

In this regard, this report studies a non-parametric approach for face tracking, namely the particle filter, which has been largely used in many related researches. The implementation in this report mainly rests on the work of Junxiang et al. [3], and we further improve both of the observation models, the fusion method and the initialization process.

1.1 Contribution

In this report, several contributions are made as listed below:

- We use a Haar feature-based cascade classifier to automatically obtain the initial state of targets, instead of manually selecting targets on the first frame of video;
- We improve the color histogram observation model, the moment invariants observation model and the fusion method mentioned in [3] respectively;
- We compare the performances of particle filters with different observation models and present the robustness of the integrated particle filter.

1.2 Outline

Section 2 briefly reviews related work for face tracking using particle filters in particular. Section 3 explains the method and its implementation in detail. Section 4 presents some experimental results and compares them to draw some preliminary conclusions. Section 5 ends the report with final conclusions.

2. Related Work

The task of face tracking in video usually could be separated into two steps. A face detection algorithm is required to capture the state of targets in the first place. And then a face tracker works for generating a trajectory of tracking based on the face's motion in subsequent frames. Generally, approaches [4] to solving face tracking problem can be classified into three categories based on the tracked target types: point tracking, kernel tracking and silhouette tracking. Point tracking represents detected targets by points while kernel tracking realizes it by using a rectangular template or an elliptical shape with an associated RGB color histogram; in silhouette tracking, information in the form of appearance density and shape models is usually applied.

In practice, particle filter (PF) has been proved to be an effective approach in face tracking and state estimation. Nummiaro et al. [5] proposed a particle filter using a linear prediction model and an observation model based on color histogram for face tracking; Okuma et al. [6] presented a boosted particle filter which could take advantages of the strength of mixture particle filters and Adaboost algorithm; Shen et al. [7] implemented an integration of color cue (color histogram) and shape cue (ellipse model) for particle filter based tracking; Junxiang et al. [3] combined color histogram and moment invariants as two observation models and designed a particle filter for tracking based on them. It appears that particle filters which could incorporate multiple observation models or develop themselves under the framework of ensemble learning turn out to perform in a more robust manner and have become a focus in this research field.

In this project, our method belongs to kernel tracking since it uses color histogram and moment invariants to represent a region of interest respectively. Also, it uses a probabilistic approach (particle filter) to estimate the state of moving faces and upgrades itself into an integrated version for better robustness.

3. Integrated Color-based and Moment-based PF

3.1 Face Detection

Face detection in this section is used for obtaining the initial state of face targets in videos. Some papers [5, 8] have proposed an easier way to locate the targets on the first frame of the video manually, e.g. by drawing a rectangle region which covers a face target immediately after a video is loaded. It is more efficient but less accurate since the size of the targets may affect the selection process to some degree. Here, we use a boosted classifier, namely Haar feature-based cascade classifier, to automate the face detection step. This classifier could succeed in detecting a face region and returning its current state in the form of a 1×4 vector which contains the upper-left coordinates and the size of its bounding box when an image is shown. *Fig. 1* shows the results (bounding boxes and their states) after face detection.

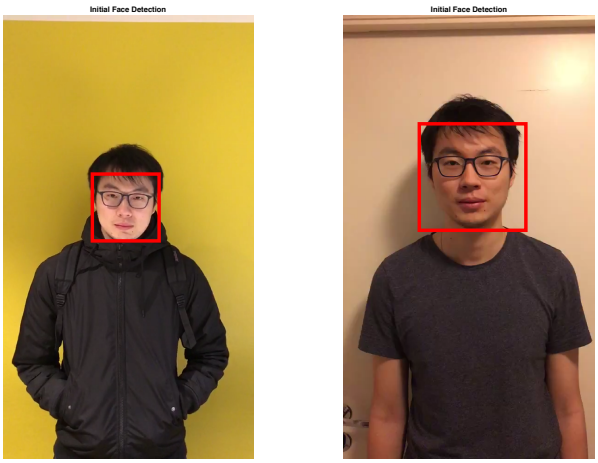

Fig. 1: Face detection results under different illumination conditions with a bounding box selecting out the face region. The initial state of the left one is [194, 344, 145, 145] while that of the right one is [164, 235, 231, 231].

3.2 Prediction Model

After the initial state of a face is acquired, we have to define a model to represent how the face target moves on the following frames. First, we can expand the state of the target from a 1×4 vector into a 1×7 vector shown below since we want to take the scaling factor into consideration:

$$s = \{x, y, \dot{x}, \dot{y}, H_x, H_y, \alpha\} \tag{1}$$

where (x, y) specify the upper-left coordinates of the bounding box, $(\dot{x}, \dot{y})$ the velocity of targets, (H_x, H_y) the width and height of the bounding box, α the scale parameter. Then we can define the prediction model as followed:

$$\begin{cases} x_{t+1} = x_t + \dot{x}_t + N(0, \Sigma_R) \\ y_{t+1} = y_t + \dot{y}_t + N(0, \Sigma_R) \\ \dot{x}_{t+1} = \dot{x}_t + N(0, \Sigma_R) \\ \dot{y}_{t+1} = \dot{y}_t + N(0, \Sigma_R) \\ H_{x,t+1} = (1 + \alpha_t) \cdot H_{x,t} + N(0, \Sigma_R) \\ H_{y,t+1} = (1 + \alpha_t) \cdot H_{y,t} + N(0, \Sigma_R) \\ \alpha_{t+1} = \alpha_t + N(0, \Sigma_R) \end{cases} \tag{2}$$

Σ_R here represents the covariance of the process noise. The prediction model above assumes that the target propagates with a constant velocity and the scale parameter simply changes the size of the bounding box.

3.3 Observation Model

The observation model is used for measuring the similarity between a potential hypothesis and the target given a frame of video. Here, we will derive two observation models based on color histogram and moment invariants respectively.

3.3.1 Color-based Model

Once a predicted state s_t is obtained in the next frame, we could calculate the likelihood of the region within the bounding box being the true face target using color histogram. We then extract the image region of the bounding box and assign each pixel value into corresponding color bins in different RGB channels. At the same time, we assign a small weight to those pixels that are far away from the region's center, and a large weight to those close to the center, so that the similarity could be mainly decided by the center part of the extracted image. The weighting function (Fig. 2) is

$$k(r) = \begin{cases} 1 - r^2, & r \leq 1 \\ 0, & otherwise \end{cases} \tag{3}$$

where r is a normalized distance from the region's center c to a certain pixel p.

$$r = \frac{\|p - c\|}{\sqrt{{H_x}^2 + {H_y}^2}} \tag{4}$$

Then each pixel gives weighted vote to its corresponding color bins and finally the color distribution, denoted as $\boldsymbol{p}$, can be obtained after counting votes in each bin. Here, we use 8×8×8 bins in RGB space and could reshape it into a 1×24 flatten vector with the form of $\boldsymbol{p} = \{p_1^R, \dots p_7^R, p_1^G, \dots p_7^G, p_1^B, \dots p_7^B\}$.

A regular way to compute similarity of two states is called the Bhattacharyya distance. We first calculate Bhattacharyya coefficient, which is defined as

$$\rho(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\vec{\boldsymbol{p}} \cdot \vec{\boldsymbol{q}}} \tag{5}$$

From equation (5), we can see that ρ will be maximal when $\vec{p}$ and $\vec{q}$ could match perfectly and will decrease as they become more dissimilar. And we will take the Bhattacharyya distance as a similarity measure as defined below:

$$d = \sqrt{1 - \rho(\boldsymbol{p}, \boldsymbol{q})} \tag{6}$$

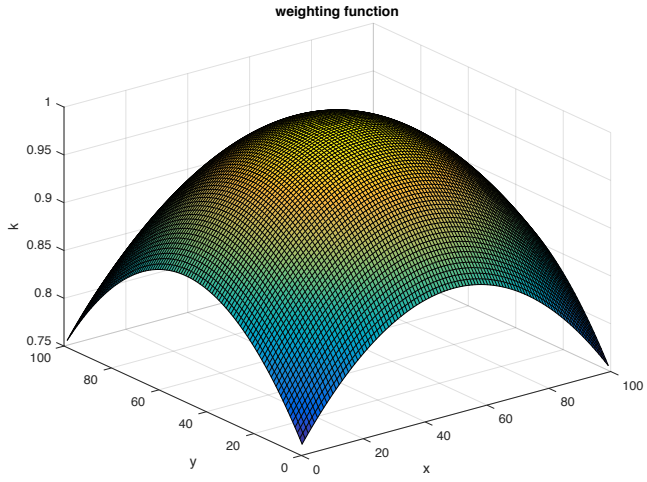

Fig. 2: Weighting function in three dimension

3.3.2 Moment-based Model

Since color histogram neglects spatial information when measuring similarities, a moment-based observation model is introduced to complement the color-based model and enhance the robustness of the face tracker.

For an image $I(x, y)$, the moment of order $(p + q)$ can be expressed as

$$m_{pq} = \sum_x \sum_y x^p y^q \, I(x, y) \tag{7}$$

The centroid of the image is defined as

$$\mu_{pq} = \sum_x \sum_y (x - \frac{m_{10}}{m_{00}})^p (y - \frac{m_{01}}{m_{00}})^q \, I(x, y) \tag{8}$$

Then the normalized central moment is defined as

$$\eta_{pq} = \mu_{pq} / {\mu_{00}}^{(p+q+2)/2} \tag{9}$$

Further, these normalized central moments can be rearranged into seven Hu moment invariants [9], which are invariant with respect to translation, rotation and scaling.

$$\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02} \\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4{\eta_{11}}^2 \\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2 \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
\phi_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{10}$$

Similarly, these seven moment invariants could be placed in a 1×7 vector for each color channel, and could be reshaped into a 1×21 flatten vector when the three vectors are grouped together, e.g. $\boldsymbol{\phi} = \{\phi_1^R, \dots \phi_7^R, \phi_1^G, \dots \phi_7^G, \phi_1^B, \dots \phi_7^B\}$. Then we can use this vector to calculate its likelihood of being a perfect match to the target.

But since these moment invariants don't share the same unit, any mathematical addition or subtraction will be meaningless. Therefore, we use the following standardization to get rid of those units in the first place and measure the distance of moment vector $\boldsymbol{\phi}$ and target vector $\boldsymbol{\phi}_T$:

$$d = \frac{1}{21} \cdot sum(abs[(\boldsymbol{\phi}_T - \boldsymbol{\phi})/(\boldsymbol{\phi}_T + \boldsymbol{\phi})]) \tag{11}$$

As we can see in (11), a minor distance will be presented when $\boldsymbol{\phi}$ is very similar with the target whereas the distance will increase as $\boldsymbol{\phi}$ becomes more dissimilar.

3.4 Particle Filter

In this section, we will use particle filter to estimate the state of the face target. The algorithm in this case is generally illustrated in the table below:

Tab. 1: Particle filter algorithm used in face tracking

Particle Filter
1. Initialize the face target and its initial state. Initialize N particles randomly with equal weights;
2. Propagate each particle based on our prediction model;
3. Update weights for all particles based on our observation model;
4. Estimate the posterior state of the target in current frame;
5. Update the target combining with the posterior state (optional);
6. Resample the particles based on their weights. Go back to step 2.

3.4.1 Initialization

We use the face detection method mentioned above in section 3.1 to obtain the face target. As for the particles, we randomly generate N particles with equal weights 1/N in the beginning, and the range of these particles is set manually. When the range parameters are set to be large, particles tend to spread widely in the image, and may take longer time for them to gather around a particular hypothesis.

3.4.2 Particle Propagation

The particles are described in the form of (1), and are propagated according to prediction model (2). Because of the resampling step, the particles will be gradually grouped around the target after several iterations, see Fig. 3. The covariance of the process noise here could decide how wide the particles spread in the current frame: If the target tends to move in a more predictable way, then decreasing the covariance could lower the risk of losing tracking of the target (may cause the system less robust); if the target often changes its moving direction, then a relative larger covariance could help particles to capture the target timely.

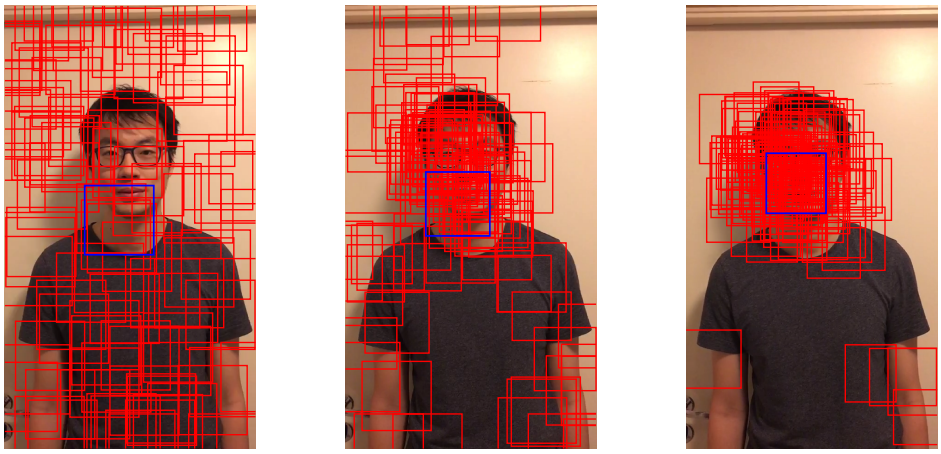

Fig. 3: Particle Propagation affected by resampling (iteration = 1, 3, 6 from left to right; the red rectangles represent particles while the blue rectangle is the posterior state)

3.4.3 Weight Update

When particle propagation is finished, we will turn to the observation models to calculate the likelihood of each particle being the target. The likelihood is directly decided by the distance representing the similarity between the target and candidates. The equation is shown below:

$$\omega_i = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{{d_i}^2}{2\sigma^2}\right), i = 1, 2, \dots, N \tag{12}$$

After normalization, each particle will be assigned a weight according to its distance. The larger that distance is, the more dissimilar the particle is with the target. Thus, it will get a lower weight, and vice versa. Besides, the value of σ will influence the performance of the particle filter. As we know, when σ is given a large value, the distribution will be a short-flat Gaussian, where higher weights will be given to particles with even a large distance. This means particles may spread out and survive more easily. On the other hand, a small σ will lead to a compact group of particles which will converge to a certain hypothesis quickly. Both cases are faced with target-losing problem due to particle deprivation for a large σ and the compactness of particles for a small σ.

Since we possess two observation models in our implementation, each particle will simply have two weights, one from the color-based model and the other from the moment-based model. We will illustrate how to fuse the two weights in section 3.5.

3.4.4 Estimation

After particle propagation and weight update, the posterior state will be estimated using the weighted mean state of all particles. The equation is shown below:

$$s_E = \sum_{i=1}^{N} \omega_i \cdot s_i \tag{13}$$

Since the particles could represent multimodal distribution, we generally choose a threshold of weight and eliminate those particles with weights lower than that threshold when estimating the posterior state using (13). In this way, the problem could be handled appropriately.

3.4.5 Target Update

The face target may change slightly due to e.g. illumination changes over time. Therefore, we need to adapt the color histogram or moment invariants by taking the estimated posterior state into consideration. Here, we use the balance filter as in (14). $\boldsymbol{p_{T,t}}$ denotes the histogram or the moment invariants of the target at time t, and $\boldsymbol{p_{E,t}}$ denotes that of the posterior state at time t; α is a parameter which decides how much we trust on the weighted estimation.

$$\boldsymbol{p_{T,t+1}} = (1-\alpha)\cdot \boldsymbol{p_{T,t}} + \alpha \cdot \boldsymbol{p_{E,t}} \tag{14}$$

3.4.6 Resampling

Some particles may propagate towards wrong direction and appear far off the target. In this case, we need to filter out these particles and keep alive those that could stay around the target. Here, we apply the systematic resampling to resample the particles at time $t-1$ with replacement and redistribute particles at time t. Particles with higher weights stand a larger chance to survive, and may duplicate itself several times in the new particle set. The skeleton of the systematic resampling is shown in Tab. 2. Compared to vanilla resampling, this method is less computationally demanding and could decrease variance to some extent.

Tab. 2: Skeleton of systematic resampling

Systematic Resampling
1. Calculate cumulative weights of N particles. Generate a random number $\boldsymbol{r}$;
2. Find the first particle whose cumulative weight is above $\boldsymbol{r}$, and place it into the new particle set;
3. Update $\boldsymbol{r}$: $\boldsymbol{r} = \boldsymbol{r} + \boldsymbol{1/N}$. If already N particles in the new particle set, continue to step 4; otherwise, go back to step 2;
4. Assign an equal weight, which is 1/N, for all new particles.

3.5 Fusion Method

After obtaining two estimated posterior states from color-based and moment-based particle filters, we need to deal with the problem of how to fuse them together properly. A common solution is to assign different weights two these two posterior states and then calculate their weighted sum as the final estimation of the face target. We first derive the distances representing the similarity from the two posterior states, and then calculate weights using the equations [3] below:

$$\begin{cases} w_{clr} = \dfrac{\exp(-\tau \cdot d_{clr})}{\exp(-\tau \cdot d_{clr}) + \exp(-\tau \cdot d_{mnt})} \\ w_{mnt} = \dfrac{\exp(-\tau \cdot d_{mnt})}{\exp(-\tau \cdot d_{clr}) + \exp(-\tau \cdot d_{mnt})} \end{cases} \tag{15}$$

d_{clr} and d_{mnt} are denoted as the distance of posterior state from the color-based model and that from the moment-based model. τ is a constant of attenuation. The final state is decided by:

$$s_F = w_{clr} \cdot s_{E,clr} + w_{mnt} \cdot s_{E,mnt} \tag{16}$$

4. Experimental Results

In our implementation, we separately generate 50 particles for the color-based model and 50 for the moment-based model in the beginning, and integrate the posterior states into a final estimation. We load a video taped by author's mobile phone, and the face in that video moves and turns its moving direction in a relatively frequent way. We use this video to test the integrated particle filter and compare its error performances with the color-based particle filter and moment-based one respectively.

Fig. 4 shows how different trackers behaves at different positions. It appears that the moment-based particle filter performs better dynamically and often gives responses timely when the target changes its moving direction. The color-based tracker is however more stable but responds slowly. The integrated version takes the advantages of the above two trackers and surely outperforms the other two in general. This could also be observed in Fig. 5.

Fig. 5 depicts how error changes as the target moves in the video. Here, we use the Haar feature-based cascade classifier to obtain the true position of the face, and denote the Euclidean distance between the true and the estimated position as the error of the trackers. We can observe that there exist three peaks in Fig. 5. The first one is caused by the fact that initial particles spread out widely in the frame and may take several iterations to converge. And when the target changes its moving direction, the process noise will be responsible for capturing the target since the prediction model assumes a constant velocity. The errors will increase abruptly when this assumption is violated to a considerable degree. Therefore, we can know the last two peaks in Fig. 5 occur when the target stops and moves in an opposite direction.

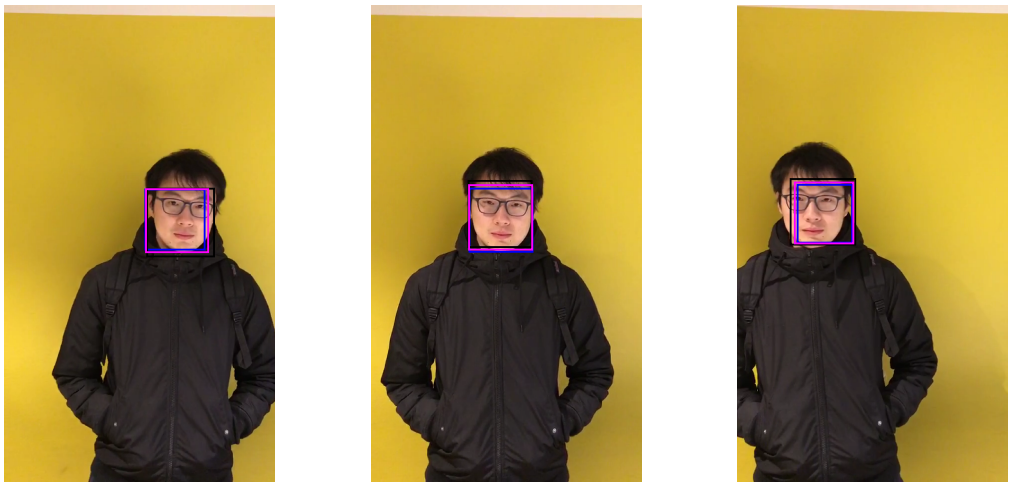

Fig. 4: Results of different trackers (pink: integrated PF; blue: color-based PF; black: moment-based PF)

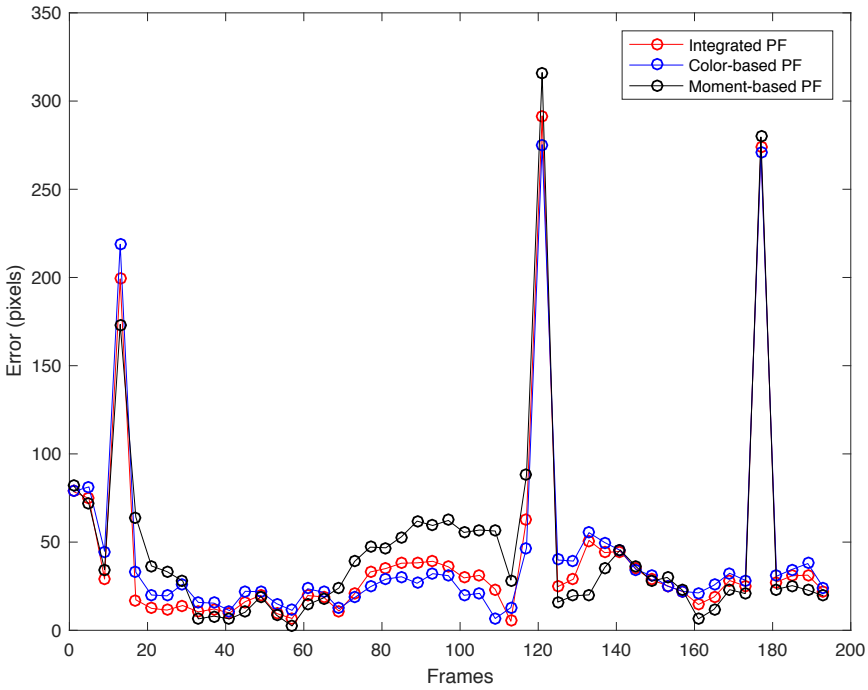

Fig. 5: Error performances (red: integrated PF; blue: color-based PF; black: moment-based PF)

5. Summary and Conclusions

This report mainly illustrates an implementation of face tracking using an integrated particle filter which combines the color-based and moment-based observation model in order to achieve better robustness and dynamical properties. The experimental results show that the algorithm works well. But still further researches are required to improve its performance, e.g. observation model based on normalized cross correlation can be used as a similarity measure. Besides, more tests on multi-faces should be included to enhance its reliability.

References

[1] Li, P. (2008). An adaptive binning color model for mean shift tracking. *IEEE transactions on circuits and systems for video technology*, *18*(9), 1293-1299.

[2] Dornaika, F., & Davoine, F. (2006). On appearance based face and facial action tracking. *IEEE transactions on circuits and systems for video technology*, *16*(9), 1107-1124.

[3] Junxiang, G., Tong, Z., & Yong, L. (2009, October). Face tracking using color histograms and moment invariants. In *Broadband Network & Multimedia Technology, 2009. IC-BNMT'09. 2nd IEEE International Conference on* (pp. 519-523). IEEE.

[4] Chau, D. P., Bremond, F., & Thonnat, M. (2013). Object Tracking in Videos: Approaches and Issues. *arXiv preprint arXiv:1304.5212*.

[5] Nummiaro, K., Koller-Meier, E., & Van Gool, L. (2003). An adaptive color-based particle filter. *Image and vision computing*, *21*(1), 99-110.

[6] Okuma, K., Taleghani, A., De Freitas, N., Little, J. J., & Lowe, D. G. (2004, May). A boosted particle filter: Multitarget detection and tracking. In *European Conference on Computer Vision* (pp. 28-39). Springer Berlin Heidelberg.

[7] Shen, C., Van den Hengel, A., & Dick, A. (2003). Probabilistic multiple cue integration for particle filter based tracking. *Australian Pattern Recognition Society*.

[8] Xu, X., & Li, B. (2005, July). Head tracking using particle filter with intensity gradient and color histogram. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on* (pp. 888-891). IEEE.

[9] Razali, M. T., & Adznan, B. J. (2006). Detection and classification of moving object for smart vision sensor. In *Information and Communication Technologies, 2006. ICTTA'06. 2nd* (Vol. 1, pp. 733-737). IEEE.