Reducing Frame Rate for Object Tracking

Pavel Korshunov[1] and Wei Tsang Ooi[2]

[1] National University of Singapore, Singapore 119077,
`pavelkor@comp.nus.edu.sg`
[2] National University of Singapore, Singapore 119077,
`ooiwt@comp.nus.edu.sg`

Abstract. Object tracking is commonly used in video surveillance, but typically video with full frame rate is sent. We previously have shown that full frame rate is not needed, but it is unclear what the appropriate frame rate to send or whether we can further reduce the frame rate. This paper answers these questions for two commonly used object tracking algorithms (frame-differencing-based blob tracking and CAMSHIFT tracking). The paper provides (i) an analytical framework to determine the critical frame rate to send a video for these algorithms without them losing the tracked object, given additional knowledge about the object and key design elements of the algorithms, and (ii) answers the questions of how we can modify the object tracking to further reduce the critical frame rate. Our results show that we can reduce the 30 fps rate by up to 7 times for blob tracking in the scenario of a single car moving across the camera view, and by up to 13 times for CAMSHIFT tracking in the scenario of a face moving in different directions.

1 Introduction

Object tracking is a common operation in video surveillance systems. However, given an object tracking algorithm, it is unclear what frame rate is necessary to send. Typically, video is sent at the rate of full video camera capacity, which may not be the best option if network bandwidth is limited.

Previously, we have shown in [1] that frame rate can be significantly reduced without object tracking losing the object. We found that the critical frame rate for a given algorithm depends on the speed of tracked object. The simple way to determine the critical frame rate is to run algorithm on a particular video sequence, dropping frames and noticing which rate causes the algorithm to lose the object. Such approach however is not practical, because objects in real surveillance videos move with different speeds, and the critical frame rate therefore should depend on this parameter. We suggest finding critical frame rate using analysis based on the algorithm's key design elements (specific object detection and tracking mechanism) and measured speed and size of the tracked object.

In this paper, we focus on two tracking algorithms, blob tracking algorithm that relies on frame differencing and foreground object detection by Li *et al.* [2] as well as Kalman filter for tracking; and CAMSHIFT algorithm [3], in which objects are represented as color histograms, and tracking is performed using mean

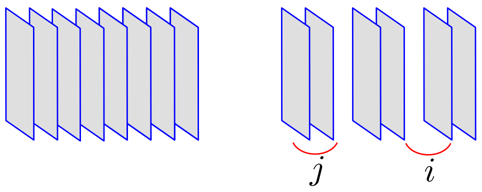


Fig. 1. Dropping i out of $i + j$ frames. i is the drop gap.

shift algorithm. We present an analytical framework formalizing the dependency between video frame rate and algorithms' accuracy. We estimate critical frame rate using analysis with assumption of known speed and size of the tracked object. Guided by the estimation, we slightly modify these tracking algorithms making them adaptive and more tolerant to the videos with even lower frame rate.

In Section 2, we present analysis of the critical frame rate for object tracking. In Section 3, we demonstrate how the dependency between frame rate and accuracy can be estimated specifically for the blob tracking. We also specify the critical frame rate for this algorithm. In Section 4, we present similar analysis for CAMSHIFT tracking. In Section 5, we show how, using our estimations and measurement of speed and size of the tracked object, we can modify these tracking algorithms adapting them to the reduced frame rate. Section 6 ends the paper with conclusion and future works.

2 General Analysis

We degrade temporal video quality by applying the dropping pattern "drop i frames out of $i+j$ frames", where i is *drop gap*, and j is the number of consecutive remaining frames (see Figure 1). Note that the same frame rate can correspond to two different dropping patterns, for instance, dropping 2 out of 3 frames results in the same frame rate as dropping 4 out of 6 frames. The reason for choosing such dropping pattern is because we found that drop gap is more important factor for the performance of the tracking than simply a frame rate. Therefore, instead of critical frame rate, we focus on finding *critical drop gap*, which would determine the corresponding frame rate.

First, we present an estimation of the critical drop gap for an object tracking algorithm without taking into account the specific method of detection and tracking. For simplicity, consider a video containing a single moving object, which can be accurately tracked by the algorithm. We can notice that dropping frames affects the speed of object. Since video is a sequence of discrete frames, the speed of object can be understood as a distance between the centers of object positions in two consecutive frames, which we call inter-frame speed denoted as Δd. Without loss of generality, we can say that for every object tracking algorithm there exists a $\Delta \tilde{d}$ such that, if object moves for a larger distance than $\Delta \tilde{d}$, the algorithm loses it.

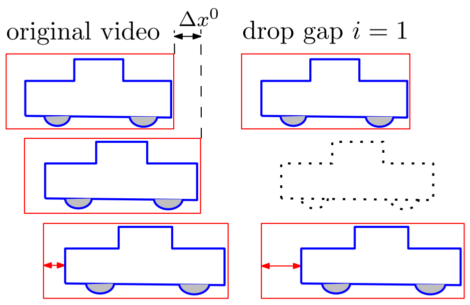


Fig. 2. The schema of the difference between object foreground detection for original video and for video with dropped frames.

Let Δd_0 be the maximal inter-frame speed of the object in the original video, when no frame dropping is applied yet. If we drop frames with drop gap $i = 1$, the new maximum inter-frame speed can be approximated as $\Delta d_1 = 2\Delta d_0$. Then, for general frame dropping pattern, $\Delta d_i = (i + 1)\Delta d_0$. Assume we know the original speed of the object and the algorithm's threshold $\Delta\tilde{d}$. Then, we can compute the maximum number of consecutive frames that can be dropped, i.e., critical drop gap $\tilde{i}$, as

$$\tilde{i} = \frac{\Delta\tilde{d}}{\Delta d_0} - 1. \tag{1}$$

3 Blob Tracking Algorithm

For blob tracking algorithm, due to frame differencing detection, the value $\tilde{i}$ depends on the size and the speed of tracked object. If too many consecutive frames are dropped, the object in the current frame appear so far away from its location in the previous frame that the frame differencing operation results in detecting two separate blobs (see Figure 3(b)). Such tracking failure occurs when the distance between blob detected in the previous frame and blob in the current frame is larger than the size of the object itself. Therefore, this distance is the threshold distance $\Delta\tilde{d}$. To determine its value, we need to estimate the coordinates of the blob center in the current frame, which depend on its location and size in the previous frame.

In this analysis, we assume a single object monotonously moving in one direction. Although this assumption considers only a simplified scenario, many practical surveillance videos include objects moving in a single direction towards or away from the camera view. Also, such movements of the object in camera's view as rotating or only changing in size (when object goes away/towards camera view but does not move sideways) do not have a significant effect on frame differencing object detection. We also assume, without loss of generality, that the object moves from left to right with its size increasing linearly. The assumption allows us to consider only changes in coordinate x, and width w.

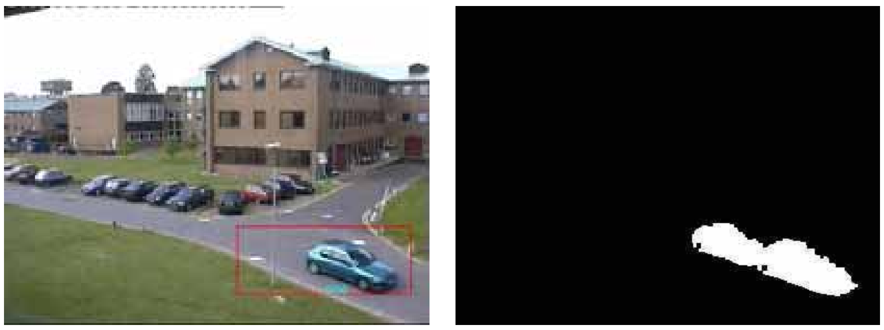

(a) Detected foreground object with drop gap 14 frames. PETS2001 video. (b) Binary mask of the frame in 3(a). Effect of drop gap on frame differencing.

Fig. 3. The foreground object detection based on frame differencing.

Increase/decrease in size is important because when tracked objects approach or move away from the camera, their size changes. In practice, when object moves in both x and y coordinates, the overall critical drop gap would be the minimum of the two values estimated for corresponding coordinates.

Consider the original video when no frames are dropped. We assume the average distance between fronts of the blob when it shifts from the previous frame to the current frame is Δx^0. We consider the front of the object because it is more accurately detected by frame differencing. When frame differencing is used, the resulted detected blob is the union of the object presented in the previous and current frames (see Figure 3(b)). Therefore, when we drop frames, the width of the blob in the frame following after the drop gap will be larger than that in the original video sequence (see Figure 2 for illustration). However, the front of the blob would be detected in the same way as in the original video.

Since frame dropping affects size of the detected object, we consider average change in size as Δw^0. The superscript indicates the size of the drop gap, which is 0 when frames are not dropped. Assume that x_k^0 is x-coordinate of blob's center in *k-th* frame, then, we can estimate its coordinate in the frame $k+i+1$ as following,

$$x_{k+i+1}^0 = x_k^0 + (i+1)\Delta x^0 - (i+1)\frac{\Delta w^0}{2}. \tag{2}$$

If i frames are dropped after frame k, the detected blob in the $k+i+1$ frame is the union of actual object appearing in frames k and $k+i+1$ (as Figure 2 illustrates). Then, the width difference $(w_{k+i+1}^i/2 - w_k^0/2)$ can be approximated as $(i+1)\Delta x^0/2$. Therefore, the blob's center in the $k+i+1$ frame can be estimated as,

$$x_{k+i+1}^i = x_k^i + (i+1)\Delta x^0 - (i+1)\frac{\Delta x^0}{2} = x_k^0 + (i+1)\frac{\Delta x^0}{2}, \tag{3}$$

since $x_k^i = x_k^0$.

As was mentioned, $\Delta\tilde{d} = |x^{\tilde{i}}_{k+\tilde{i}+1} - x^{\tilde{i}}_{k}|$, where $\tilde{i}$ indicates the critical drop gap. The failure of the blob tracking implies that $\Delta\tilde{d} = w^0_k$, where value w^0_k is the width of the blob detected in frame k. Therefore, from equation (3), we obtain $w^0_k = \Delta\tilde{d} = (\tilde{i}+1)\frac{\Delta x^0}{2}$, from which we can find the critical drop gap to be

$$\tilde{i} = \frac{2w^0_k}{\Delta x^0} - 1. \tag{4}$$

In practice, values w^0_k and Δx^0 can be determined by either keeping the history of speed and size of tracked object or by estimating their average values for a particular surveillance site.

In addition to the estimation of the critical drop gap for blob tracking, we can estimate the dependency function between accuracy of the algorithm and video frame rate. Such estimation is possible because of the way drop gap affects the accuracy of the frame differencing object detection algorithm used in blob tracking. We can define blob detection error for a particular frame as the distance between blob centers detected in this frame for the degraded video (with dropped frames) and the original video. Then, the average error, denoted as ϵ_{ij}, is the average blob tracking error for all frames in the video. This ϵ_{ij} function can be used as accuracy metric for the blob tracking depicting the tradeoff between tracking accuracy and video frame rate.

Using equations (2) and (3) we can estimate the blob tracking error for $k+i+1$ frame as following,

$$\left|x^i_{k+i+1} - x^0_{k+i+1}\right| = (i+1)\left|\frac{\left(\Delta x^0 - \Delta w^0\right)}{2}\right| = (i+1)C, \tag{5}$$

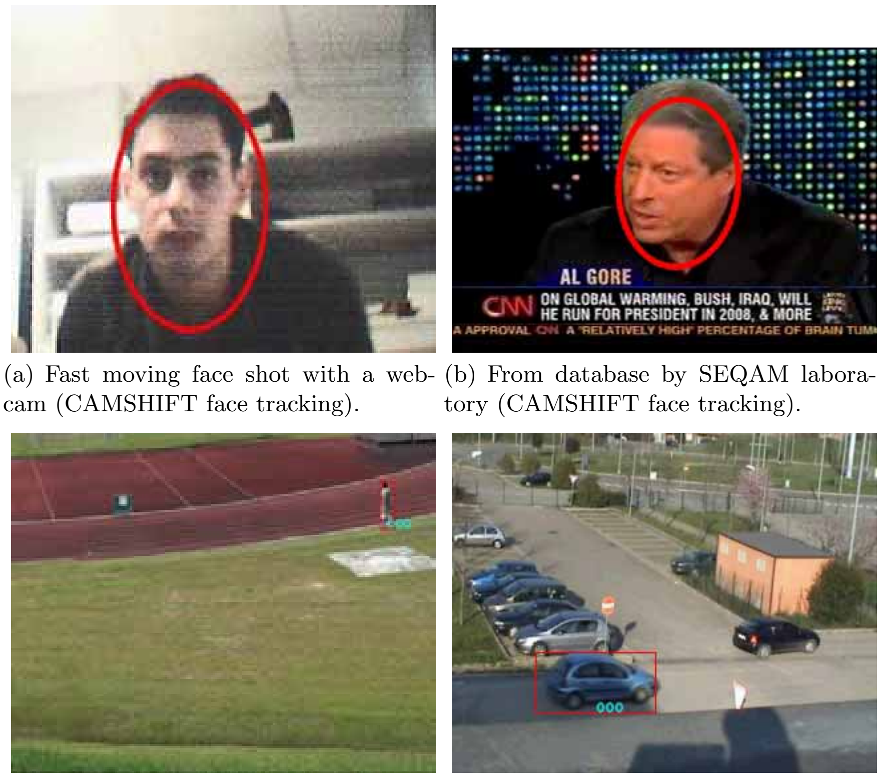
where constant $C \geq 0$ depends on the size and the speed of object in the original video.

Since we apply the dropping pattern "drop i frames out of $i+j$ frames", we need to estimate the blob tracking error for each of the remaining j frames in the video. There is no error in detecting blob for $j-1$ frames that do not have drop gap in front of them, i.e., for these frames, the result of the frame differencing would be the same as in original video with no dropping. Therefore, the average error for all j frames is the error estimated for the frame, which follows the drop gap (equation (5)) divided by j:

$$\epsilon_{ij} = \frac{i+1}{j}\left|\frac{\left(\Delta x^0 - \Delta w^0\right)}{2}\right| = \frac{i+1}{j}C. \tag{6}$$

Note the important property of this function that the average error is proportional to i and inversely proportional to j.

We performed experiments to validate the estimation of the average blob tracking error ϵ_{ij}. We use several videos from ViSOR video database, PETS2001 datasets, as well as videos we shot on campus with a hand-held camera (example screenshots in Figure 3(a), Figure 4(c), and Figure 4(d)). Videos include moving


AL GORE
CNN ON GLOBAL WARMING, BUSH, IRAQ, WILL HE RUN FOR PRESIDENT IN 2008, & MORE
A APPROVAL CNN A 'RELATIVELY HIGH' PERCENTAGE OF BRAIN TUM

(a) Fast moving face shot with a webcam (CAMSHIFT face tracking).

(b) From database by SEQAM laboratory (CAMSHIFT face tracking).

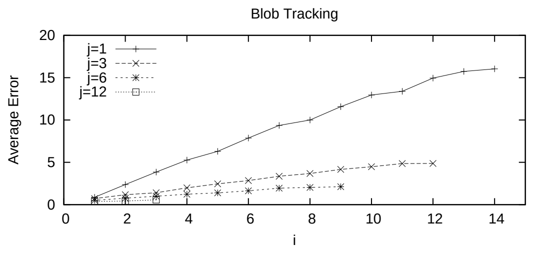
(c) Shot on campus with hand-held camera (blob tracking).

(d) From VISOR video database (blob tracking).

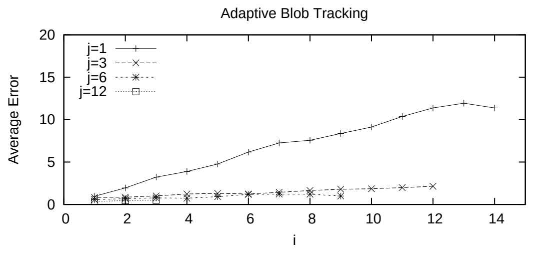
Fig. 4. Snapshot examples of videos used in our experiments.

cars, person on a bicycle and people walking in a distance. We ran blob tracking algorithm on these videos and applied different dropping patterns. We plot the resulted average error against drop gap i when value j is 1, 3, 6, and 12. The results are shown in Figure 5(a) (original video is 158 frames of 384×288, 30 fps) and Figure 6(a) (original video is 148 frames of 320×256, 30 fps).

Figure 5(a) shows the resulted average tracking error plotted against the drop gap i when value j is 1, 3, 6, and 12. It can be noted from the Figure 5(a) that for each fixed value j the average error is proportional to i. Also, average error is inversely proportional to j, as indicated by the angles of each line in the graph (for instance, angle of the line marked as “j=1” is three times larger than the angle of the line “j=3”). Figure 6(a) demonstrates similar results. These experimental results strongly support our analytical estimation of the average error given in the equation (6). The figures do not reflect the critical drop gap value because even for large drop gaps the blob tracking did not lose the track of the car in this test video sequence.



(a)



(b)

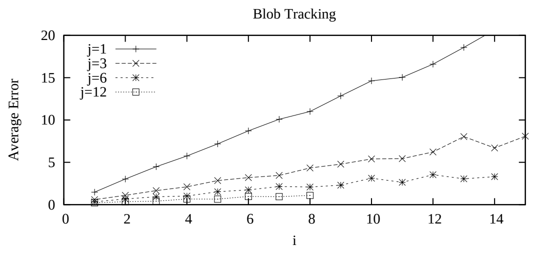
Fig. 5. Accuracy of original and adaptive blob tracking algorithm for PETS2001 video (snapshot in Figure 3(a)).

4 CAMSHIFT Algorithm

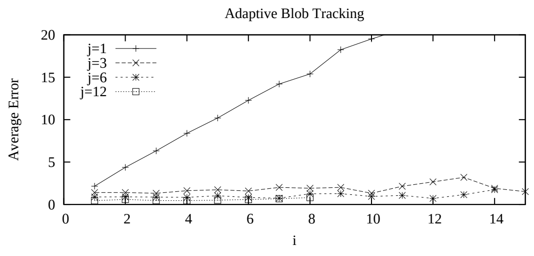
CAMSHIFT object tracking [4] relies on color histogram detection and mean shift algorithm for tracking. The algorithm searches for a given object's histogram inside a subwindow of the current frame of the video, which is computed as 150% of the object size detected in the previous frame. Therefore, if the object, moves between two frames from its original location for a distance larger than half of its size, the algorithm will lose the track of the object. Hence, assuming we drop i frames before frame $k+i+1$, the threshold distance $\Delta\tilde{d} = \frac{w_k^0}{2}$, where w_k^0 is the width of the blob detected in frame k. Since CAMSHIFT does not use frame differencing, drop gap does not have an additional effect on object's size. Therefore, we can estimate the center of the blob after drop gap i using the equation (2) instead of equation (3). Hence, the critical drop gap can be derrived as

$$\tilde{i} = \frac{w_k^0}{2\Delta x^0 - \Delta w^0} - 1. \tag{7}$$

Estimating the average tracking error loses its meaning for CAMSHIFT tracking because it uses a simple threshold for detection of the object in the current frame. If the drop gap of the given frame dropping pattern is less than critical drop gap in equation (7), the algorithm continue tracking the object, otherwise it loses it. And the critical drop gap depends on the changes in speed and size of the object.



(a)



(b)

Fig. 6. Accuracy of original and adaptive blob tracking algorithm for VISOR video (snapshot in Figure 4(d)).

We performed experiments with CAMSHIFT tracking algorithm to verify our analytical estimation of the critical drop gap (equation (7)). We used several videos of a moving face shot with a simple web-cam, videos of talking heads by SEQAM laboratory and some movie clips (example screenshots in Figure 4(a) and Figure 4(b)). Figure 8(a) (original video is 600 frames of 352 $\times$ 288, 30 fps) and Figure 7 (original video is 303 frames of 320 $\times$ 230, 30 fps) show average tracking error vs. drop gap for CAMSHIFT tracking and various frame dropping patterns. Figure 7, corresponding to the video of a talking head (see snapshot in Figure 4(b)), demonstrates that tracking algorithm does not lose the face even when drop gap is 14 frames. The reason is because the face in the video does not move around and is always present in the search subwindow of CAMSHIFT tracker. However, for the experiments shown in Figure 8(a), the video with fast moving head was used (see snapshot in Figure 4(a)). It can be noted that the algorithm does not lose the face until value of drop gap is 8, because for the smaller drop gaps, the face is still within a search subwindow and can be detected by the histogram matching. The fluctuations in the average error for the larger drop gaps appear because the face is either lost by the tracker or, for some large enough gaps, it would move out of the subwindow and move back in, hence the tracker does not lose it. We conducted experiments with more videos and observed that the critical drop gap value is smaller for videos with faster moving faces and larger for videos with slower moving faces. These observations agree with equation (7).

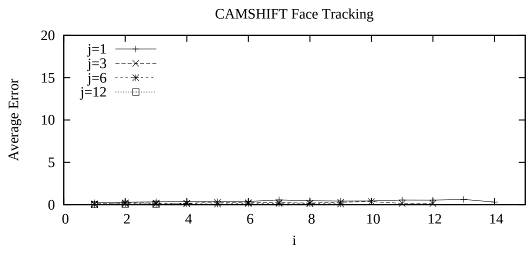


Fig. 7. Accuracy of original and adaptive CAMSHIFT tracking algorithm for video with slow moving face (snapshot in Figure 4(a)).

5 Adaptive Tracking

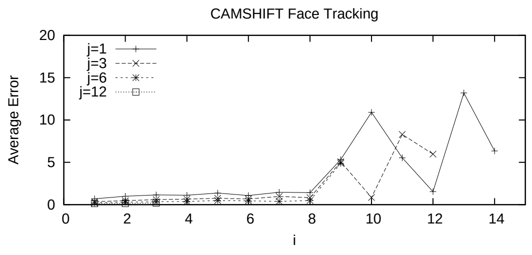
We propose to modify blob tracking and CAMSHIFT algorithms and make them more tolerant to video with low frame rate. We have shown that average error and the critical frame rate of tracking algorithms depend on speed and size of the object in the original video. Therefore, if we record these characteristics for previous frames, the location and the size of object in the frame that follows a drop gap can be approximated. Adjusting to frame dropping in such way allows us to reduce the average error for blob tracking algorithm and increase the critical drop gap for the CAMSHIFT algorithm.

Blob tracking algorithm tracks the detected foreground object using the simplified version of Kalman filter: $x_k = (1-\alpha)x_{k-1} + \alpha z_k$, where x_k and x_{k-1} represent estimated coordinates of the object in the current and previous frames, z_k is the output of the object detector, and $\alpha \leq 1$ is some constant. When $\alpha = 1$, then the tracker trusts the measurement z_k fully and its average error can be estimated by equation (6). In cases when $\alpha < 1$, the accuracy of the tracking against the frame dropping worsens, due to the larger shifts in blobs' centers for videos with high drop gap. We propose using adaptive Kalman filter [5] to make blob tracking more tolerant to the frame dropping. We apply the filter only to the width of the object, because the front is detected correctly by frame differencing (see Figure 2). The filter can be defined as following,

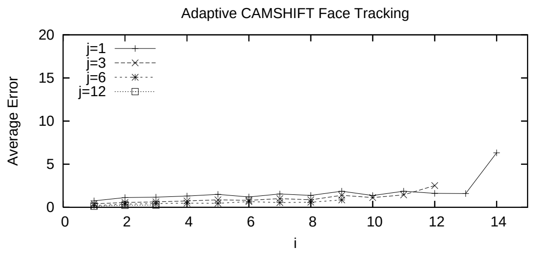
$$\tilde{w_k} = w_{k-1} + K_k\,(w_{k-1} + u_k) \qquad \tilde{P_k} = P_k + Q_k$$
$$P_k = (1 - K_k)\tilde{P_k} \qquad\qquad K_k = \frac{\tilde{P_k}}{(\tilde{P_k} + R_k)},$$

where Q_k and R_k are the process and measurement noise covariances; $\tilde{w_k}$ is the new estimate of the blob's width in the current frame; w_{k-1} is blob's width in the last not dropped frame; u_k is the width measurement provided by the frame-differencing based detector.

Kalman filter depends on correct estimation of the error parameters, Q_k and R_k. By looking at Figure 2, we can set $Q_k = (i\varDelta w^0)^2$, which estimates how big the tracked object should be at frame $k+i+1$ compare to its width before the



(a)



(b)

Fig. 8. Accuracy of original and adaptive CAMSHIFT tracking algorithm for video with fast moving face (snapshot in Figure 4(a)).

drop gap at frame k. R_k is essentially the error of the measurement, i.e., the output of the foreground object detector, therefore, $R_k = (w^i_{k+i+1} - w^0_{k+i+1})^2$.

Since w^i_{k+i+1} can be estimated as $w^0_k + (i+1)\Delta x^0$ and w^0_{k+i+1} as $w^0_k + (i+1)\Delta w^0$, we can approximate $R_k = (i+1)^2(\Delta x^0 - \Delta w^0)^2$. We obtain the values of Δw^0 and Δx^0 by recording the speed of the object and how fast it grows in size using last two available frames.

To compare how adaptive Kalman filter improves the accuracy of blob tracking, we performed the same experiments varying frame dropping pattern. The average error for blob tracking with adaptive Kalman filter is plotted in Figure 5(b) and Figure 6(b), which can be compared to results with original algorithm in Figure 5(a) and Figure 6(a) respectively. We can note that the accuracy of the adaptive blob tracking algorithm is improved for larger drop gaps (larger frame rate reduction). In both figures, Figure 5(b) and Figure 6(b), the angles of the lines in the graph are not inversely proportional to j anymore, giving fundamentally different bound on the average error. All lines with $j > 1$ are almost parallel to x-axis. It means that Kalman filter adapts very well to the drastic changes in speed and size of the object that occur due to the frame dropping. The constant increase in the average error for $j = 1$, is because, for such dropping pattern, all remaining frames are separated by drop gaps. In this scenario, adaptive Kalman filter accumulates the approximation error of object's size and speed. Therefore, the critical frame rate can be achieved with j that is at least equal to 2. If we take $i = 12$, the original frame rate is reduced by 7 times.

We also modified the CAMSHIFT tracking algorithm, adjusting the size of its search subwindow to the frame dropping. We simply increased the subwindow size in the current frame by $i\Delta x^0$, where i is the drop gap. The average error of this adaptive CAMSHIFT algorithm for the video with fast moving face is shown in Figure 8(b). Comparing with the results of original algorithm in Figure 8(a), we can notice that the adaptive tracker performs significantly better for the larger drop gaps. The experiments show that we can drop 13 frames out of 14 with a tradeoff in small average error. It means that CAMSHIFT algorithm, for this particular video sequence, can accurately track the face with frame rate reduced by 13 times from the original. For the news videos of talking heads, where face does not move significantly around, adaptive algorithm performs with exactly the same accuracy results as the original algorithm. Therefore, Figure 7 illustrates essentially both versions of the algorithm, original and adaptive. These experiments demonstrate that by using analysis to modify CAMSHIFT algorithm, we can improve its performance on videos with fast moving faces, while retaining the original accuracy on videos with slow moving faces.

6 Conclusion

In this paper, we use analysis to estimate the tradeoff between accuracy of two common tracking algorithms and video frame rate. Such estimation depends on the speed and size of the tracked object, and therefore, in practice, such measurements of the object need to be taken (for instance, running average of these values during the last few frames). We also show that slight modifications to existing algorithms can significantly improve their accuracy for the video with larger reductions in frame rate. These findings motivate us to use reasoning for determining critical frame rate (not just running many different experiments) for other video analysis algorithms. The findings also encourage the development of the new object tracking algorithms robust to highly degraded video.

References

1. Korshunov, P., Ooi, W.T.: Critical video quality for distributed automated video surveillance. In: Proceedings of the ACM International Conference on Multimedia, ACMMM'05, Singapore (November 2005) 151–160
2. Li, L., Huang, W., Gu, I.Y., Tan, Q.: Foreground object detection from videos containing complex background. In: Proceedings of the ACM International Conference on Multimedia, ACMMM'03, Berkeley, CA, USA (November 2003) 2–10
3. Bradski, G.R.: Computer vision face tracking as a component of a perceptual user interface. In: Proceedings of the Forth IEEE Workshop on Applications of Computer Vision, WACV'98, Princeton, NJ (January 1998) 214–219
4. Boyle, M.: The effects of capture conditions on the CAMSHIFT face tracker. Technical Report 2001-691-14, Department of Computer Science, University of Calgary, Alberta, Canada (2001)
5. Welsh, G., Bishop, G.: An introduction to the kalman filter. In: Proceedings of SIGGRAPH 2001. Volume Course 8., Los Angeles, CA, USA (August 2001)