Speed Up Temporal Median Filter for Background Subtraction

Mao-Hsiung Hung, Jeng-Shyang Pan
Innovative Information Industry Research Center (IIIRC),
Shenzhen Graduate School, Harbin Institute of Technology
Shenzhen, China
E-mail: mhhung0502@gmail.com, jspan@cc.kuas.edu.tw

Chaur-Heh Hsieh
Dept. of Computer and Communication Engineering
Ming Chuan University
Taoyuan, Taiwan
E-mail: harrishsieh@gmail.com

***Abstract*—Temporal median filter is one of most popular background subtraction methods. However, median operation is very time-consuming which limits its applications. This paper presents a fast algorithm to reduce the computation time of the temporal median operation. By utilizing the characteristics of high correlation of adjacent frames, the fast algorithm designs a simple mechanism to check whether the median of the current frame is equal to that of the previous frame. The proposed algorithm reduces the computing frequency of median operations significantly, and the experimental results indicate it is much faster than the existing algorithms.**

Keywords- background subtraction, temporal median filter, fast algorithm

I. INTRODUCTION

Background subtraction is an important step in many computer vision applications such as video surveillance, people counting, human gesture recognition, moving object detection and tracking, traffic monitoring, and video indexing and retrieval. Background subtraction detects moving objects from the difference between the current frame and a background image. To obtain accurate detection of moving objects, the background image must be a representation of the scene with no moving objects and must be updated regularly so as to adapt to the varying lighting conditions and geometry settings [1]. Many background subtraction methods have been proposed in the literatures such as running Gaussian average, temporal median filter, mixture of Gaussians, kernel density estimation, etc. The major problems exist in these methods are either computation expensive or memory expensive [1]. With the rapid advance of memory technology, memory cost is getting less critical in recent years.

Temporal median refers the median of previous frames in a video sequence to establish a statistical background model for background subtraction. Lo and Velastin [2] first presented the temporal median background update technique for congestion detection system of underground platform. Cucchiara et al. [3] pointed out that temporal median filter provides an adequate background model which immediately reflects sudden scene change. Temporal median filter offers acceptable accuracy while achieving a high frame rate and having limited memory requirements [1]. Therefore, it has become one of most popular background subtraction methods [4-8].

The temporal median of the k-th frame, $I_{\text{Med}}(x,y,k)$, is obtained by

$$I_{\text{Med}}(x,y,k) = \text{Med}\left(I(x,y,k-1), I(x,y,k-2), ..., I(x,y,k-N)\right) \quad (1)$$

where $I(x,y,k-1),\ldots, I(x,y,k-N)$ denote pixel values located at (x,y) over the previous N frames of the k-th frame and Med(.) means the median operation. The methods of the median operation are categorized into sort-based and selection-based. Sort-based median operation is the simplest method. It sorts input data and then picks the middle-order element from the sorted sequence, which requires significant computational complexity. Selection-based median methods do not need to sort input data to determine the median, so they are more efficient than sort-based. Histogram selection [9-10] and divide-and-conquer selection [11] are good examples of the selection-based median methods.

However, the selection-based method still consumes considerable time, so that it is hard to satisfy real-time requirement for some applications, especially for high video quality applications. This paper presents a fast algorithm to further speed up the temporal median operation, which is based on histogram selection. Because pixel data are very high correlated frame by frame, it has high possibility that the two medians of the consecutive frames are equal. We present a scheme to check the median repetition between two consecutive frames. The experimental results indicate our proposed method reduces the computation significantly, and it far exceeds the real-time requirement.

II. PROPOSED METHOD

A. Median Selection Based on Histogram

For the convenience of the representation, we redefine the temporal neighborhood of the pixel at (x, y) of the k-th frame as

$$D_k = \{d_{k-1}, d_{k-2}, \ldots, d_{k-N}\} \quad (2)$$

where the data set D_k stores the previous N pixel data (d_{k-1}, $d_{k-2},\ldots, d_{k-N}$) located at (x,y). Here we assume that the pixel value is in the range of 0 and 255. O_{Mid} means the middle order of the N data, i.e. $O_{\text{Mid}}=(N-1)/2$ (N must be odd). Then, the median selection based on the histogram is described as the following function.

978-0-7695-4180-8/10 $26.00 © 2010 IEEE
DOI 10.1109/PCSPA.2010.79

Function: m = medhist(D)
// Input: D stores the previous N pixel data
// Output: m returns the median of D

hn = hist(D) // hist(.) returns the histogram of the data set
$csum$= 0 // $csum$ means the cumulative function of the histogram
for i = 0 to 255
    if $hn[i]$>0 then
        $csum$+= $hn[i]$
        if $csum \geq O_{\text{Mid}}$ then break
    end if
end for
return i

The above histogram selection first calculates the histogram of the input data set. Then, the cumulative function of the histogram is evaluated by incrementing index from 0 to 255. When the cumulative function reaches the middle order, the current index is the median of the data set required.

B. *Median Selection Based on Histogram and Repetition Checking*

To develop the fast algorithm of the histogram selection, we first design a lower bound and an upper bound of the cumulative function at the median, denoted by lb and ub. By slightly modifying the above histogram selection scheme, we obtain the following function to evaluate the median as well as the two bounds of a data set.

Function: $\{m, lb, ub\}$ = medhist_bnd(D)
// Input: D stores the previous N pixel data
/* Ouput: m returns the median of D. lb and ub respectively returns the lower bound and upper bound of the cumulative function at the median. */

hn = hist(D) // hist(.) returns the histogram of the data set
$csum$ = 0 // $csum$ is the cumulative function of the histogram
for i = 0 to 255
    if $hn[i]$>0
        $csum$ += $hn[i]$
        if $csum \geq O_{\text{Mid}}$ then
        $lb = csum - hn[i]+1$, $ub = csum$
            break
        end if
    end if
end for
return i, lb, ub

Similar to the original histogram selection, when the cumulative function of the histogram ($csum$) reaches the middle order, lb can be obtained by lb=$csum$–$hn[i]$+1, where $hn[i]$ is the value of the indexed histogram and ub is equal to $csum$. Fig. 1 shows an example of cumulative function of histogram, and obviously the middle order satisfies the relation of $lb \leq O_{\text{Mid}} \leq ub$. We apply the relation to develop the repetition checking scheme.

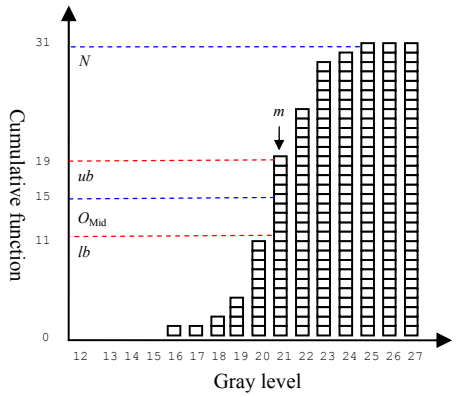


Figure 1. An example of cumulative function of histogram

The data set of D_k contains pixel data within the previous N frames of the k-th frame, as shown in Eq.(2). The next data set of D_{k+1} for (k+1)-th frame is as

$$D_{k+1} = \{d_k, d_{k-1}, \ldots, d_{k-N+1}\} \tag{3}$$

where d_k is the pixel data of the k-th frame. The difference of D_{k+1} and D_k is just d_k and d_{k-N}, which implies D_{k+1} and D_k are highly correlated. Thus, it has high possibility that the medians of the two data sets are equal, which we call median repetition. Consequently, the proposed repetition checking of the median between the two consecutive frames has promise to greatly reduce the median operation in temporal direction.

The temporal median filter with histogram selection and repetition checking is implemented by the following function.

Function: $\{m_{k+1}, lb_{k+1}, ub_{k+1}\}$ = medhist_repchk(D_k, d_k, m_k, lb_k, ub_k)
/* Input: D_k stores the previous N pixel data of the k-th frame. d_k is the pixel data of the k-th frame. m_k, lb_k and ub_k are respectively the median, lb and ub of D_k. */
/* Ouput: m_{k+1} returns the median of D_{k+1}. lb_{k+1} and ub_{k+1} respectively return lb and ub of D_{k+1}. */

insert d_k into D_k and delete d_{k-N} from D_k to obtain D_{k+1}
$\{tf, lb_{k+1}, ub_{k+1}\}$ = repchk($d_{k-N}, d_k, m_k, lb_k, ub_k$) //call function repchk(.) for repetition checking
if tf then
    $m_{k+1} = m_k$ // if repetition checking is true
else
    $\{m_{k+1}, lb_{k+1}, ub_{k+1}\}$ = medhist_bnd(D_{k+1}) // if repetition checking is false

end if
return m_{k+1}, lb_{k+1}, ub_{k+1}

Given the data and parameters of the k-th frame, d_{k-N}, d_k, m_k, lb_k and ub_k, the repetition checking algorithm first calculate the parameter of the next frame, lb_{k+1} and ub_{k+1}, according to the relations of the values of d_{k-N}, d_k and m_k. The relation of d_{k-N} and m_k contains three cases: "less than", "equal to" and "greater than". The relation of d_k and m_k also include the same three cases. Thus, the two relations generate nine permutations, which can be utilized to calculate lb_{k+1} and ub_{k+1} from lb_k and ub_k. For examples, when the deleted element d_{k-N} and the inserted element d_k are less than the previous median of m_k, lb and ub are unchanged, i.e. $lb_{k+1}= lb_k$ and $ub_{k+1}= ub_k$. When d_{k-N} is less than m_k and the d_k is equal to m_k, lb is decreased by 1 but ub is unchanged, i.e. $lb_{k+1}= lb_k-1$ and $ub_{k+1}= ub_k$. Similarly, the other seven conditions are used to update the next lower bound and upper bound. Finally, if $lb_{k+1} \le O_{\text{Mid}} \le ub_{k+1}$, the median of the next frame is equal to that of the current frame; i.e., $m_{k+1}= m_k$. The repetition checking is implemented by the following function.

Function: $\{tf, lb_{k+1}, ub_{k+1}\}$ = repchk(d_{k-N}, d_k, m_k, lb_k, ub_k)
/* Input: d_{k-N} and d_k respectively denote the deleted and inserted element for the next data set. m_k means the previous median. lb_k and ub_k are the previous bounds of the cumulative function at the median. */
/* Output: tf returns 1 if the current median is equal to the previous median, otherwise tf returns 0. */

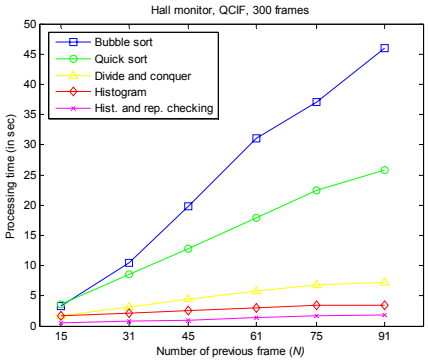
if $d_{k-N} < m_k$ and $d_k < m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k$
if $d_{k-N} < m_k$ and $d_k = m_k$, then $lb_{k+1} = lb_k - 1$, $ub_{k+1} = ub_k$
if $d_{k-N} < m_k$ and $d_k > m_k$, then $lb_{k+1} = lb_k - 1$, $ub_{k+1} = ub_k - 1$
if $d_{k-N} = m_k$ and $d_k < m_k$, then $lb_{k+1} = lb_k + 1$, $ub_{k+1} = ub_k$
if $d_{k-N} = m_k$ and $d_k = m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k$
if $d_{k-N} = m_k$ and $d_k > m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k - 1$
if $d_{k-N} > m_k$ and $d_k < m_k$, then $lb_{k+1} = lb_k + 1$, $ub_{k+1} = ub_k + 1$
if $d_{k-N} > m_k$ and $d_k = m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k + 1$
if $d_{k-N} > m_k$ and $d_k > m_k$, then $lb_{k+1} = lb_k$, $ub_{k+1} = ub_k$

if $lb_{k+1} \le O_{\text{Mid}}$ and $O_{\text{Mid}} \le ub_{k+1}$, then tf=1 else tf=0
return tf, lb_{k+1}, ub_{k+1}

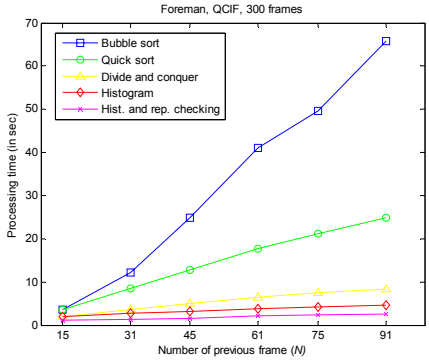
III. EXPERIMENTAL RESULT

In the experiments, we evaluate our proposed method with five MPEG-4 standard testing videos. Among those videos, "Akiyo" is with low motion, "Hall monitor" and "Container" are with middle motion, and "Foreman and Coast guard" are of high-motion videos. The formats of the videos consist of CIF (352×288) and QCIF (176×144) with frame rate of 30 Hz. Each video contains 300 frames. The testing bench uses Visual C++ with Intel Core2 Duo at 2.53GHz. We compare the processing times of our method with existing four methods, and the results are listed in Table I and Table II for CIF and QCIF, respectively. The methods for comparison contain two sort-based methods and two selection-based methods. The former are bubble sort and quick sort, and the later are divide-and-conquer and histogram selection. Our proposed method is based on histogram selection and repetition checking, denoted by hist. & rep. checking. The average processing time of the five videos is listed in the last second column. The time ratio with respect to the proposed method is listed in the last column. The time ratio is defined as the average processing time of a method divided by that of our method.

The experimental results indicate that the proposed method performs significantly faster than other method for all cases. Moreover, the proposed fast algorithm can achieve 53 fps (=300/5.609) for CIF and 305 fps (=300/0.985) for QCIF that greatly excesses the real-time requirement. The proposed method performs 1.92 times and 2.31 times faster than the histogram method for CIF and QCIF, respectively. The results prove that the repetition checking effectively reduces the computing frequency of the temporal median operation.



(a) Hall monitor



(b) Foreman

Figure 2. The time complexity analysis of various algorithms

To investigate the computational complexity of the five methods, we measure the processing times under six different previous frame numbers (N) including 15, 31, 45, 61, 75 and 91. Fig. 2(a) and Fig. 2(b) show the experimental results for hall monitor and foreman sequences respectively. The results indicate that the processing times of the two sort-based methods increase approximately linearly with the increase of N, and those of the other three methods slightly rise with the increase of N. Comparing to the other four methods, the proposed method provides the least increasing rate over previous frame numbers.

IV. CONCLUSION

In this paper, we have presented a fast algorithm to speed up temporal median filter for background subtraction. This algorithm is mainly based a novel repetition checking scheme. The lower and upper bounds of the cumulative function of the histogram are developed for the check of median repetition. Due to the high correlation of the pixel data between consecutive frames, the repetition checking effectively reduces the computing frequency of the median operation. The results indicate that the proposed algorithm performs approximate two times faster than the histogram selection, which is the state-of-the-art. Because our algorithm far exceeds the real-time requirement, it makes the temporal median filter much more applicable.

REFERENCES

[1] M. Piccardi, "Background subtraction techniques: a review," in Proc. of IEEE International Conference on Systems, Man and Cybernetics, Vol. 4, pp.3099--3104, (2004)

[2] B.P.L Lo and S.A. Velastin, "Automatic congestion detection system for underground platforms," Proc. ISIMP 2001, pp. 158-161, May (2001)

[3] R. Cucchiara, C. Cranna, M. Piccardi and A. Prati, "Detecting moving objects, ghosts and shadows in video streams," IEEE Trans on Pattern Anal. and Machine Intell., Vol. 25, No. 10, pp. 1337--1442, (2003)

[4] B. Gloyer, H. K. Aghajan, K. Y. Siu, and T. Kailath, "Video-Based Freeway Monitoring System Using Recursive Vehicle Tracking," Proc. SPIE Symp. Electronic Imaging: Image and Video Processing, (1995)

[5] Cheung, S.-C., Kamath, C. Robust techniques for background subtraction in urban traffic video, Video Communications and Image Processing, SPIE Electronic Imaging, San Jose, Jan. (2004)

[6] R. Cutler and L. Davis, "View-Based Detection and Analysis of Periodic Motion," in Proceedings Fourteenth International Conference on Pattern Recognition, Vol. 1, pp. 495--500, Aug (1998)

[7] Q. Zhou and J. Aggarwal, "Tracking and classifying moving objects from videos," in Proceedings of IEEE Workshop on Performance Evaluation of Tracking and Surveillance, (2001)

[8] N.J.B. McFarlance and C.P. Schofield, "Segmentation and tracking of piglets in images," Machine Vision and Applications, Vol.8 pp.187-193, (1995)

[9] T. S. Huang, G. J. Yang and G. Y. Tang, "A Fast Two-Dimensional Median Filtering Algorithm," IEEE Trans. on Acoustics, Speech, And Signal Processing, Vol. Asp-27, No. 1, pp. 13--18, Feb. (1979)

[10] S. Perreault and P. Hebert, "Median Filtering in Constant Time," IEEE Trans. on Image Processing, Vol. 16, No. 9, pp. 2389--2394, Sept. (2007)

[11] S. Dasgupta, C. H. Papadimitriou and U. V. Vazirani, Algorithms, McGraw-Hill Higher Education, (2007)

TABLE I. COMPARISON OF PROCESSING TIMES OF VARIOUS METHODS FOR CIF VIDEOS (IN SEC)

Methods	Akiyo	Hall monitor	Container	Foreman	Coast guard	Average	Time ratio
Bubble sort	33.125	47.593	47.000	52.156	57.062	47.387	8.45
Quick sort	35.359	35.922	35.343	35.515	37.188	35.865	6.39
Divide-and-conquer	8.875	15.094	14.203	16.25	18.078	14.500	2.59
Histogram	9.203	10.094	10.391	12.454	11.765	10.781	1.92
Hist. & rep. checking	4.359	4.609	5.219	6.859	7.000	5.609	1.0

TABLE II. COMPARISON OF PROCESSING TIMES OF VARIOUS METHODS FOR QCIF VIDEOS (IN SEC)

Methods	Akiyo	Hall monitor	Container	Foreman	Coast guard	Average	Time ratio
Bubble sort	7.765	10.453	10.500	12.515	13.500	10.947	11.11
Quick sort	8.328	8.593	8.343	8.547	8.750	8.512	8.64
Divide-and-conquer	1.688	3.125	2.641	3.656	3.906	3.003	3.05
Histogram	1.906	2.157	2.172	2.688	2.453	2.275	2.31
Hist. & rep. checking	0.688	0.75	0.844	1.344	1.297	0.985	1.0