Real-time Constant Memory Visual Summaries for Surveillance

Nathan Jacobs and Robert Pless
Department of Computer Science and Engineering
Washington University, St. Louis, MO, 63130, USA
{jacobsn, pless}@cse.wustl.edu

ABSTRACT

In surveillance applications there may be multiple time scales at which it is important to monitor a scene. This work develops online, real-time algorithms that maintain background models simultaneously at many time scales. This creates a novel temporal decomposition of video sequence which can be used as a visualization tool for a human operator or an adaptive background model for classical anomaly detection and tracking algorithms. This paper solves the design problem for choosing appropriate time scales for the decomposition and derives the equations to approximately reconstruct the original video given only the temporal decomposition. We present two applications that highlight the potential of video processing; first a visualization tool that summarizes recent video behavior for a human operator in a single image, and second a pre-processing tool to detect "left bags" in the challenging PETS 2006 dataset which includes many occlusions of the left bag by pedestrians.

Categories and Subject Descriptors

I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding

General Terms

Algorithms

Keywords

change detection, background modeling, video surveillance, video analysis

1. INTRODUCTION

Recently announced plans for security and surveillance involve the use of hundreds to thousands of cameras along thousands of miles of outdoor, natural terrain. While some plans advocate putting all this video data live on the Internet so that the general public can report potential intrusions [1], it is more likely that both computational algorithms and trained human operators will be the key observers. With current technology, feasible system architectures include initial automatic stages of object detection and tracking, followed by a further human-in-the-loop analysis of the video stream.

However, when a single operator is responsible for thousands of cameras instead of dozens, one cannot expect them to remember what the scene typically looks like or how it commonly varies. Instead, it is necessary to provide additional support and visualization tools for the operator to understand the context of the scene. This paper introduces a lightweight, real-time computation that provides one such visualization tool. Additionally, the representation can instead be used as a video pre-processing for other surveillance algorithms—offering explicit control over both how long an object needs to be visible to be considered more than noise, and how long an object must be in the scene before being considered part of the background.

The contribution of this paper is the development of real-time, constant-memory algorithms that use a small collection of low-pass filtered versions of the video to provide effective video pre-processing. We demonstrate the efficacy of these tools in two experimental paradigms; first a traffic scene where the visualization tools immediately present a view of how long each vehicle has been in the scene, and second, an experiment run on the PETS 2006 dataset[1] in a challenging "left bag" detection problem.

2. PRIOR WORK

There has been relatively little work to explicitly include temporal filters of varying extents within video surveillance applications. The most recent work with temporal filters has been in gesture and action recognition; two examples are defining a vector-valued image summarizing recent motion at each pixel in order to recognize a small library of actions [2] and the reverse problem of finding all instances of a given action by searching for a specific spatio-temporal template [3].

Within the surveillance domain, anomaly detection is usually performed with reference to a background model. This background model may be based upon pixel intensity statistics, either a Gaussian mixture model [4], a non-parametric distribution [5], or a predictive model for the time sequence [6]. Alternatively, the model may be based on local estimates of the optic flow [7, 8], or parametric models of the distributions of spatio-temporal derivatives [9]. All of these background models allow for updating based on scene changes, and allow the update rule to weight the recent scene appearance more strongly. While this permits these methods to accommodate to slowly changing backgrounds, the drift rate (how much the more recent frames dominate the background model) must be set ahead of time.

[1] http://www.pets2006.net, April 25th, 2006

VSSN'06, October 27, 2006, Santa Barbara, California, USA.

The current paper illustrates how to maintain a background model simultaneously at multiple time scales. The general approach could apply to many of the above methods; in the following section we develop tools to maintain a very simple background model, and illustrate that the difference of background models with a particular ratio of time scales is an efficient tool (both theoretically and practically) to highlight many features of interest in video streams.

3. TEMPORAL DECOMPOSITION

We create a multi-resolutional temporal decomposition of a video sequence by filtering pixel intensity values. This view-based approach is particularly useful when the camera is static, as is often the case in surveillance applications. The decomposition is constructed by maintaining multiple exponentially-weighted moving averages or, in other words, filtering the sequence with multiple causal low-pass filters, each with different filtering constants.

Given an image sequence $I(x,t)$, where x is the pixel index and t denotes time, we create a set of low-pass filtered sequences $\mathbf{L} = \{L_1, \ldots, L_N\}$ defined by the following recursive equation:

$$L_i(x,t) = \alpha_i L_i(x,t-1) + (1-\alpha_i) I(x,t) \tag{1}$$

for $t > 0$ and $L_i(x,0) = B_x$, an application dependent initialization constant (usually the median background intensity at x).

The filtering constant $\alpha_i \in [0,1]$ determines the amount of the current image $I(x,t)$ to include in $L_i(x,t)$. The set $\mathbf{L}$ of low-pass filtered images depends on the set of filter constants $\mathbf{A} = \{\alpha_1, \ldots, \alpha_N\}$. Selection of $\mathbf{A}$ depends on the video frame-rate and temporal scales of interest.

As an alternative to Equation 1, L_i can be written as the following linear equation:

$$L_i(x,t) = (1-\alpha_i) \sum_{j=1}^{t} \alpha_i^{t-j} I(x,j) + \alpha_i^t B_x. \tag{2}$$

This form of L_i is used as a basis for the image sequence reconstruction process and subsequent methods.

In the remainder of this paper we explore applications of this temporal decomposition. Note that while descriptions are in terms of a temporal decomposition of pixel intensity values it is possible to decompose other signals generated from video sequences, *i.e.* a binary foreground-background detection sequence.

4. SIGNAL RECONSTRUCTION

Given a set of N exponentially-weighted moving average images it is possible to reconstruct the original video exactly by inverting Equation 2 (if the original video is N or fewer frames). This is of little practical value because memory use is unbounded. However, given additional constraints on the form of the signal, it is possible to approximately reconstruct significantly more frames with constant memory. In this section we describe one such constraint that provides an estimate of when a pixel most recently changed and provides intuition for the less computationally intensive method described in Section 5.

In this section we constrain the reconstructed signal at each pixel to be piecewise constant with only two pieces. While not a practical model over long durations it is useful for modeling short-term image changes (*e.g.* a person or vehicle occluding the background).

Specifically, we make the assumption that the signal at each pixel has the following form:

$$I(x,t) = \begin{cases} f_{x,1} & \text{if } t < r_x \\ f_{x,2} & \text{if } t \geq r_x \end{cases}. \tag{3}$$

Reconstructing the signal reduces to determining when the pixel changed r_x and the pixel intensity before and after the change, respectively $f_{x,1}$ and $f_{x,2}$. Estimating the reconstruction reduces to determining these parameters at each pixel. With this signal model, Equation 2 can be simplified so that $\hat{L}_i(x,t) =$

$$\begin{aligned}
&= (1-\alpha_i) \sum_{j=1}^{t} \alpha_i^{t-j} I(x,j) + \alpha_i^t B_x \\
&= (1-\alpha_i) \left[\sum_{j=1}^{r_x-1} \alpha_i^{t-j} f_{x,1} + \sum_{j=r_x}^{t} \alpha_i^{t-j} f_{x,2} \right] + \alpha_i^t f_{x,1} \\
&= \alpha_i^t (1-\alpha_i) \left[\sum_{z=0}^{r_x-2} \alpha_i^{-(z+1)} f_{x,1} + \sum_{z=0}^{t-r_x} \alpha_i^{-(z+r_x)} f_{x,2} \right] \\
&\qquad + \alpha_i^t f_{x,1} \\
&= \frac{1-\alpha_i}{1-\alpha_i^{-1}} \left[\alpha_x^{t-1} f_{x,1} (1-\alpha_x^{1-r_x}) + \alpha_i^{t-r_x} f_{x,2} (1-\alpha_i^{r_x-t-1}) \right] \\
&\qquad + \alpha_i^t f_{x,1} \\
&= f_{x,2} + (f_{x,1} - f_{x,2}) \alpha_i^{t+1-r_x}
\end{aligned} \tag{4}$$

Given the known low-pass filter responses $L_i(x,t)$ we seek the the parameters $f_{x,1}$, $f_{x,2}$, and r_x for the piecewise-constrained response $\hat{L}_i(x,T)$ that most closely approximate the actual signal responses $L_i(x,T)$ in the least-squares sense:

$$\underset{r_x, f_{x,1}, f_{x,2}}{\arg\min} \sum_{i=1}^{N} (L_i(x,T) - \hat{L}_i(x,T))^2 \tag{5}$$

Reconstructing a signal from the temporal decomposition is instructive but in practice performing the reconstruction is computationally intensive—requiring solving Equation 5 for each pixel. In the next section we show a method for extracting useful information from $\mathbf{L}$ without this step.

5. CHANGE DETECTION WITHOUT RECONSTRUCTION

Determining when the most recent change occurred in a video, which we define as determining r_x in Equation 4 for all pixels, is useful for many applications. However, the computationally intensive signal reconstruction step described in Section 4 is unnecessary when an exact estimate of r_x is not needed. In this section we describe a filter, based on a combination of two low-pass filter responses, and a corresponding approximation that is significantly less computationally intensive.

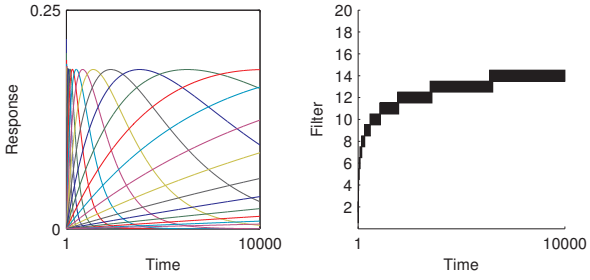
We define the difference of low-pass filter $D_{i,j}(x,t) = L_i(x,t) - L_j(x,t)$ which, if we assume a two-piece piecewise constant signal, can be simplified as follows:

$$D_{i,j}(x,t) = (f_{x,1} - f_{x,2})(\alpha_i^{t+1-r} - \alpha_j^{t+1-r}). \tag{6}$$

We now explore properties of this filter when applied to piecewise constant signals.

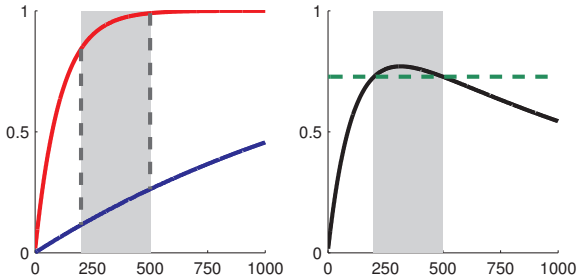
5.1 Estimating Recent Changes

Figure 1 shows the responses of a set of difference of low-pass filters, with α_i and α_j set to consecutive elements of $\mathbf{A} = 1 - \{e^{-1}, e^{-2}, \ldots, e^{-N}\}$, to a unit step function input. Each function $D_{i,i+1}$ is maximal over a continuous temporal region. The insight is that find the maximal $D_{i,i+1}$ response indicates when the step ocurred. Note that this structure is also evident in the responses to real signals that are not perfect unit step functions, see Figure 4.



(a) Response (b) Max Response

Figure 1: Difference of low-pass filter response to a unit step function input ($r_x = 0$). In this example, the filter constants are logarithmically-spaced, *i.e.* $\mathbf{A} = \{\alpha_1, \ldots, \alpha_n\} = 1 - \{e^{-1}, e^{-2}, \ldots, e^{-N}\}$. (a) Notice that a given difference of low-pass filter response is the maximum response for a continuous temporal interval. The particular response that is the maximum (or exceeds a threshold) can be used as an estimate of when the input step occurred. (b) A view of the thresholded response in which each row corresponds to a line in (a). In this example the same constant threshold was used for each response.



(a) Low-pass filter responses (b) Difference of low-pass filter response

Figure 2: Responses of a set of filters automatically derived to detect pixels that changed between 250 and 500 frames ago (gray region). Both plot show responses to a unit step function input at $t = 0$. (a) The responses (red and blue) by the two low-pass filters. The region between the two vertical line corresponds to the temporal region of interest. (b) Shows the difference of the two low-pass filters in (a). As desired, the threshold (green) is exceeded only when $250 < t < 500$.

In Section 5.2 we show the inverse of this operation. Instead of determining temporal intervals from known filter constant and threshold values we show how to design these values given known temporal intervals of interest.

5.2 Designing Filtering Constants

In some applications the range of r_x values of interest is known at deployment time. In such cases the approximation inherent with a generic set of filtering constants can be avoided by designing $\mathbf{A}$ for the specific application requirements. Using this method temporal accuracy is improved without additional runtime cost.

More precisely we want to determine α_1, α_2, and c such that $c < D_{1,2}(x,t)$ when $t_1 \leq t \leq t_2$ (see Figure 2). This yields the following equations:

$$(f_{x,1} - f_{x,2})(\alpha_i^{t_1+1} - \alpha_j^{t_1+1}) = c \quad (7)$$

$$(f_{x,1} - f_{x,2})(\alpha_i^{t_2+1} - \alpha_j^{t_2+1}) = c \quad (8)$$

To find a unique solution we need an additional constraint. Empirically we find choosing α_1 such that $L_1(x,t_2) = 0.99 * f_{x,2}$ (*i.e.* constraining the intersection of the red curve with the right-most dashed line in Figure 2.(a)) leads to nearly maximal differences between $D_{1,2}$ and c over the range $[t_1, t_2]$. We plan to explore the effect of changing k on the robustness to noise in future work.

Using α_1 we can solve numerically to find the non-trivial value of α_2 (the trivial solution is $\alpha_2 = \alpha_1$). With α_1 and α_2 we can directly compute the specific value of c needed to properly threshold the signal.

Note that filters are designed for $f_{x,1} = 0$ and $f_{x,2} = 1$. At runtime, the threshold for $D_{i,j}$ needs to be rescaled according to the actual signal. As an approximation, we use the exponentially weighted-moving average with the largest filtering constant to estimate $f_{x,1}$ and exponentially weighted-moving average with the smallest filtering constant to estimate $f_{x,2}$.

6. SAMPLE APPLICATIONS

Security personnel are often tasked with monitoring multiple video streams. A key question that arises when a new video stream is presented is "How long has that object been there?". This section highlights how tools to answer this question directly can be derived using only the temporal decomposition (and not, for example, rewinding or reviewing the entire video).

We consider two different cases. The first is the visualization of the history of an entire scene that using colors or multiple image to indicate how long each part of the scene has been constant. The second is the local analysis of a single pixel location in an indoor train station scene which is part of the PETS 2006 "left bag" data set. a scene with multiple time scales—moving vehicles, vehicles that have stopped momentarily, vehicles that are waiting at a light, and vehicles that have parked. The ability to evaluate potential threats within a scene may require such knowledge of the status of each car. Maintaining the temporal video decomposition allows this to be computed on demand, in real-time.

As an illustration, using video from a static camera mounted far above an intersection, we create filtered images that separates objects that have been static for a long time from those that are moving. We maintain 4 background models using low-pass filters with alpha values: $\mathbf{A} = \alpha_1, \alpha_2, \ldots, \alpha_4 = 1 - \{e^{-1}, e^{-3}, e^{-5}, e^{-7}\}$.

Figure 3(a) shows three frames of the input sequence, and Figure 3(b) shows the decomposition of each frame—with the different filtered images stacked as a column. These filtered images are created by compositing a background image with information extracted from a set of three difference of low-pass filtered images ($D_{1,2}$, $D_{2,3}$, and $D_{3,4}$). For illustration purposes, we define a background image as the average of all images in the sequence (although in a continuously operating online system, we could use the image L_4 or the low-pass filtered image with the largest filtering constant). Pixel values that differ from background by more than 25 gray values are considered to be foreground. At each frame, each foreground pixel has a largest response in one of the three difference image; that foreground pixel is drawn onto the corresponding filtered image. In the first column, the scene is new, each pixel location just changed from the background, and all objects are therefore drawn on the top image. In the second column, corresponding to frame 60, the bus and opposite cars are shown at the second timescale, while the moving cars show up at the shortest time scale. Finally, in the third column, the bus and cars have still not moved, and are drawn in the filtered image corresponding to

the longest time scale. Figure 3(c) shows a false color image where the color indicates the length of time that each pixel has been static, and summarizes each column of Figure 3(b) in one image.

Left-bag Detection Without hypothesizing a background model it is still possible to estimate when an object appeared in the scene, even in the case of substantial noise and short occlusions. Figure 4(a,b,c) shows several frames of the PETS 2006 "left bag" challenge. Figure 4(d) shows the pixel intensity profile of a pixel (the center of the square in the image) that views the bag. Figure 4(e) shows the $D_{i,i+1}$ (difference of low pass filter images) for $i \in \{1..27\}$, where $\alpha_i = 1 - e^{-(0.5+\frac{i}{2})}$. The piece-wise constant reconstruction of the pixel intensity using *only* the $D_{i,i+1}$ values (following Equation 5) indicates a time that the bag was left that is accurate to a within frame of a human estimate. Note that this accuracy is achieved despite significant noise before the bag is left, and 5 occlusions (people walking in front of the bag) after the bag has been left.

7. DISCUSSION

This paper outlines an online algorithm to maintain a temporal decomposition of a video sequence that highlights changes at different time scales. We illustrate that this decomposition offers new tools for developing visualization and providing context cues for the analysis of surveillance video. These contextual cues will be increasingly important as individual security personnel become responsible for very large numbers of cameras.

There are several parameters in the algorithms we present—while we find that behavior is relatively robust to small changes of these parameters, replacing the ad-hoc choices for these constants is a key next step. Furthermore, extending this idea of maintaining background models at multiple time scales will apply as well the more comprehensive models (including the distribution rather than value of the background).

8. REFERENCES

[1] Associated Press, "Web cams on texas border." New York Times, June 9 2006.

[2] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001.

[3] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes.," in *Proc. International Conference on Computer Vision*, pp. 1395–1402, 2005.

[4] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2246–2252, 1999.

[5] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density for visual surveillance," in *Proceedings of the IEEE*, vol. 90, pp. 1151–1163, July 2002.

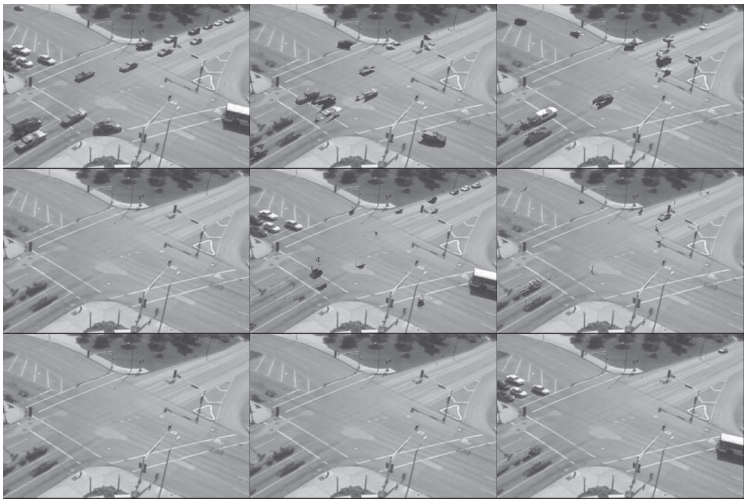
[6] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proc. International Conference on Computer Vision*, pp. 255–261, 1999.

[7] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh, "Background modeling and subtraction of dynamic scenes," in *Proc. International Conference on Computer Vision*, pp. 1305–1312, 2003.

[8] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 302–309, 2004.

[9] R. Pless, J. Larson, S. Siebers, and B. Westover, "Evaluation of local models of dynamic backgrounds," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 73–78, 2003.

(a) Original frames $t = \{1, 60, 150\}$

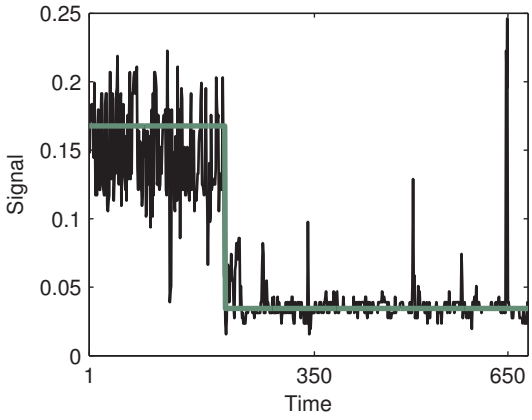


(b) Filtered frames

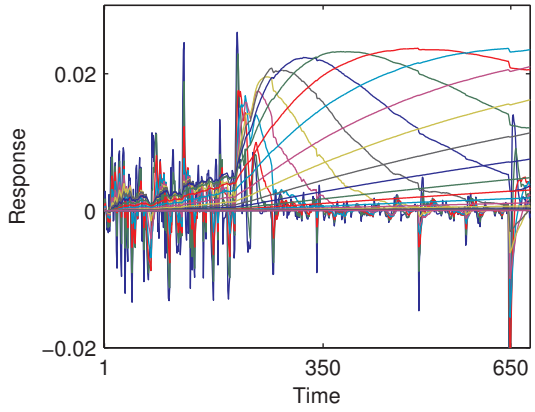
(c) False-color frames

Figure 3: Examples of video frames filtered by thresholding difference of low-pass filter responses at each pixel. (a) Three frames from a video of an intersection. The bus and the cars across the intersection have just arrived and do not move for the duration of the video but other traffic continues to move. (b) The rows correspond to difference of low-pass filters tuned to detect different temporal ranges with lower rows detecting longer temporal ranges. The columns are the image generated for the above original frame. As can be seen, the images of the stationary bus and cars move from the most to the least recent time scale image as time progresses but the non-stationary vehicles remain in the most recent time scale image.

(a) $t = 30$ (b) $t = 250$ (c) $t = 649$



(d)



(e)

Figure 4: An example of reconstruction and change estimation on real video data. (a,b,c) Three example frames from a video in which a suitcase is left in front of a train. (d) The pixel intensity value over time of the pixel highlighted in (a,b,c). Notice the signal is noisy and the package is occluded several time by pedestrians (*e.g.* c). The reconstruction (green) is computed from the exponentially-weighted moving averages at the end of the video and accurately highlights the time the bag was left. This is because the difference of low-pass filter responses over time (e) have the structure expected from Figure 1.