Motion detection: Fast and robust algorithms for embedded systems

Lionel Lacassagne, Antoine Manzanera, Antoine Dupret

MOTION DETECTION: FAST AND ROBUST ALGORITHMS FOR EMBEDDED SYSTEMS

L. Lacassagne

IEF/AXIS – Digiteo Labs – Université Paris Sud

A. Manzanera

UEI – ENSTA – Paris Tech

ABSTRACT

This article introduces a new hierarchical version of a set of motion detection algorithms called $\Sigma\Delta$. These new algorithms are designed to preserve as much as possible the computational efficiency of the basical $\Sigma\Delta$ estimation, in order to target real-time implementation for low power consumption processors and embedded systems.

***Index Terms*—** Motion detection, Sigma-Delta filtering, Embedded systems, Real-Time implementation.

1. INTRODUCTION

The growing interest for developing fully automatic video surveillance systems has recently renewed the interest for fast and reliable motion detection algorithms. Such algorithms must partition the pixels of every frame of the image sequence into two classes: the *background*, corresponding to pixels belonging to the static scene (label: 0), and the *foreground*, corresponding to pixels belonging to a moving object (label: 1).

A motion detection algorithm must discriminate the moving objects from the background as accurately as possible, without being too sensitive to the sizes and velocities of the objects, or to the changing conditions of the static scene. For long autonomy and discretion purposes, the system must not consume too much computational resources (energy and circuit area) [1]. As it involves a great amount of data - like any image processing module - the motion detection is certainly the most computationally demanding function of a video surveillance system.

Background subtraction techniques have been the object of much attention for years [2]. Recently, we have proposed a new type of methods based on $\Sigma\Delta$ estimation [3]. These methods are very attractive from a computational point of view since they work on any size fixed-point arithmetic using only comparison, increment and absolute difference, while being as robust as other mono-modal statistical estimation methods (e.g. Gaussian estimation), whose computation is much more costly.

Different modified versions of the basical $\Sigma\Delta$ algorithm have been proposed since then. The purpose of this paper is to review and compare them and also to introduce a new hierarchical version.

2. $\Sigma\Delta$ BACKGROUND SUBTRACTION

The basic principle of the $\Sigma\Delta$ algorithm is to estimate parameters of the background using $\Sigma\Delta$ modulation, which is a very common tool in analog-to-digital conversion: Considering a time-varying signal f_t (continuous or discrete), we estimate a discrete signal d_t by quantizing the time indexes $\{t_i\}_{i\in\mathbb{N}}$, and then performing at every time index i the following update formulas:

If $d_{t_{i-1}} < f_{t_i}$ then $d_{t_i} = d_{t_{i-1}} - \varepsilon$ else $d_{t_i} = d_{t_{i-1}} + \varepsilon$ where ε is the discretization step (least significant bit) of d_t.

In $\Sigma\Delta$ background subtraction, the input signal is the value of every pixel over time I_t, from which we compute the first $\Sigma\Delta$ background estimator M_t. Then the values of the absolute differences $|I_t - M_t|$ are used to compute the second $\Sigma\Delta$ background estimator V_t, which is a parameter of dispersion.

2.1. Basical $\Sigma\Delta$ algorithm

Algorithm 1: Basical $\Sigma\Delta$

foreach *pixel* x **do** [step #1: M_t estimation]
- **if** $M_{t-1}(x) < I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) + 1$
- **if** $M_{t-1}(x) > I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) - 1$
- **otherwise** $M_t(x) \leftarrow M_{t-1}(x)$

foreach *pixel* x **do** [step #2: O_t computation]
- $O_t(x) = |M_t(x) - I_t(x)|$

foreach *pixel* x **do** [step #3: V_t update]
- **if** $V_{t-1}(x) < N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) + 1$
- **if** $V_{t-1}(x) > N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) - 1$
- **otherwise** $V_t(x) \leftarrow V_{t-1}(x)$
- $V_t(x) \leftarrow max(min(V_t(x), V_{max}), V_{min})$

foreach *pixel* x **do** [step #4: $\hat{E}_t$ estimation]
- **if** $O_t(x) < V_t(x)$ **then** $\hat{E}_t(x) \leftarrow 0$ **else** $\hat{E}_t(x) \leftarrow 1$

In the basical version (Alg. 1), the $\Sigma\Delta$ background M_t and $\Sigma\Delta$ variance V_t are updated every frame, according to the comparison with the current image I_t and current absolute difference O_t respectively. N is an amplification factor for V_t, allowing then to compute the motion label $\hat{E}_t$ by simply comparing O_t and V_t (typical values of N are between 1 and 4). V_{min} and V_{max} are two parameters used to control the overflow of V_t that could happens if some pixels are saturated (due to sensor over-exposition). Their typical values are 2 and $2^m - 1$ respectively (where m is the number of bits of the representation). Note that this *clipping* is a modification not present in the original version [3].

2.2. Improved algorithm: *conditional* $\Sigma\Delta$

Algorithm 2: Conditional $\Sigma\Delta$

foreach *pixel* x *with* **do** [step #1': **conditional** M_t update]
  if $\hat{E}_{t-1}(x) = 0$ **then**
    if $M_{t-1}(x) < I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) + 1$
    if $M_{t-1}(x) > I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) - 1$
    otherwise $M_t(x) \leftarrow M_{t-1}(x)$
  else
    $M_t(x) \leftarrow M_{t-1}(x)$
foreach *pixel* x **do** [step #2: O_t computation]
  $O_t(x) = |M_t(x) - I_t(x)|$
foreach *pixel* x **do** [step #3: V_t update]
  if $V_{t-1}(x) < N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) + 1$
  if $V_{t-1}(x) > N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) - 1$
  otherwise $V_t(x) \leftarrow V_{t-1}(x)$
  $V_t(x) \leftarrow max(min(V_t(x), V_{max}), V_{min})$
foreach *pixel* x **do** [step #4: $\hat{E}_t$ estimation]
  if $O_t(x) < V_t(x)$ **then** $\hat{E}_t(x) \leftarrow 0$ **else** $\hat{E}_t(x) \leftarrow 1$

The conditional version (Alg. 2 and Fig. 1) uses relevance feedback from the estimated position of the moving objects at the previous frame, given by $\hat{E}_{t-1}$. It consists in updating the $\Sigma\Delta$ background M_t and/or the variance V_t only for the pixels x considered background (i.e. where $\hat{E}_{t-1}(x) = 0$). It prevents moving object from integrating the background and/or modifying the noise variance [4].

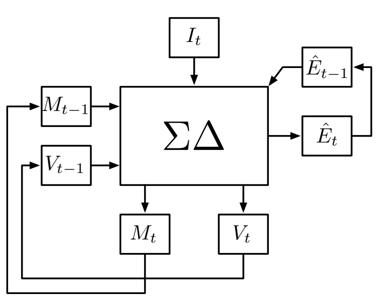

Fig. 1. conditional $\Sigma\Delta$

2.3. Zipfian estimation

The Zipfian version (Alg. 3) [5] is based on the relation between the $\Sigma\Delta$ estimation and the statistical estimation, using a Zipf-Mandelbrot distribution, which implies that the updating frequency of the background should be proportional to the dispersion of the distribution (variance). In that version, we first compute a threshold which varies according to the frame index t: ρ is the value of the index *modulo* 2^m (m is the number of bits of the representation). π is the value of the greatest power of 2 which divides ρ. Finally the threshold σ is equal to 2^m divided by π. The result is that pixels x such that $V_t(x) > 2^{m-k}$ will be updated every 2^{k-1} frames, for $k \in \{1, m\}$. To avoid auto-reference, the variance V_t is updated using a constant period T_V (usually a power of 2 between 1 and 64). T_V, like the amplification parameter N, can be automatically adjusted using a simple noise estimation method, which consists in counting the number of isolated pixels in the estimated labels $\hat{E}_t$.

Algorithm 3: Zipfian estimation

[step #0: variance threshold computation]
find the greatest 2^p that divides $(t \mod 2^m)$
set $\sigma = 2^m / 2^p$
foreach *pixel* x **do** [step #1": **conditional** M_t estimation]
  if $V_{t-1}(x) > \sigma$ **then**
    if $M_{t-1}(x) < I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) + 1$
    if $M_{t-1}(x) > I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) - 1$
    otherwise $M_t(x) \leftarrow M_{t-1}(x)$
  else
    $M_t(x) \leftarrow M_{t-1}(x)$
[**foreach** *pixel* x **do** [step #2: O_t computation]
  $O_t(x) = |M_t(x) - I_t(x)|$
foreach *pixel* x **do** [step #3": update V_t every T_V frames]
  if $t \mod T_V = 0$ **then**
    if $V_{t-1}(x) < N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) + 1$
    if $V_{t-1}(x) > N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) - 1$
    otherwise $V_t(x) \leftarrow V_{t-1}(x)$
    $V_t(x) \leftarrow max(min(V_t(x), V_{max}), V_{min})$
foreach *pixel* x **do** [step #4: $\hat{E}_t$ estimation]
  if $O_t(x) < V_t(x)$ **then** $\hat{E}_t(x) \leftarrow 0$ **else** $\hat{E}_t(x) \leftarrow 1$

2.4. New hierarchical algorithm

The hierarchical algorithm (Fig. 2) is a bi-level version of $\Sigma\Delta$ filtering. Each $\Sigma\Delta$ block implements the basic algorithm #1 of the algorithm #3. Both blocks are using conditional update. At the low level it is a conditional *temporal* update: M_t^1 and V_t^1 are updated depending on $\hat{E}_{t-1}^1$. At the high level, it is a conditional *spatial* update: M_t^0 and V_t^0 are updated depending on $\tilde{E}_t^0$, the oversampling binary mask of $\hat{E}_t^1$. The subsampling factor is in the range $[2, 10]$ and is set accordingly to the "size" of the clutter noise. Finally, a morphological post-processing is applied in two steps. The first one

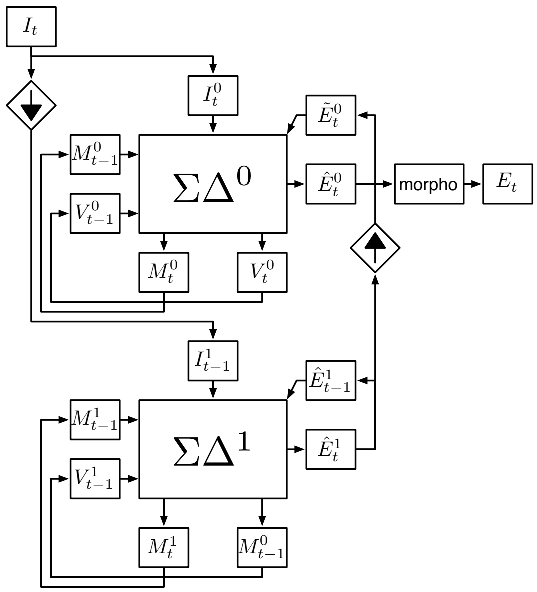

Fig. 2. hierarchic $\Sigma\Delta$

removes stand-alone pixels that are considered as noise, the second one is a 3×3 morphological closing.

3. BENCHMARK

Fig. 3. *Hall* sequence, images 38, 91, 170, 251

In order to evaluate the impact of the modifications on the performance of these algorithms, a RoC analysis has been done with the *Hall* sequence (Fig. 3) than can be considered as a *difficult* sequence because of the radial movement of non-rigid objects (people). The Ground Truth has been drawn for 4 images of that sequence. Given TP the True Positive, TN the True Negative, FP the False Positive and FN the False Negative, we compute the Matthews Correlation Coefficient (Eq. 1) instead of accuracy or product of TP ratio by TN ratio because the two classes (motion and background) are of very different size. It returns a value between -1 (perfect inverse segmentation) and $+1$ (perfect segmentation) while 0 signifies a wrong segmentation.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (1)$$

A set of 32 algorithms (combinations of parameters) has been evaluated. Figures (Tab. 1) are provided for only four of them: $\Sigma\Delta$ is the basic algorithm (Fig. 4), $\Sigma\Delta$+Zipf (Fig. 5) is the basic algorithm with Zipfian estimation, Conditional $\Sigma\Delta$ (Fig. 6) is the best mono-level algorithm with conditional update (with or without Zipfian estimation) and Hierarchical $\Sigma\Delta$ (Fig. 7)is the best two-level algorithm with conditional update. For this benchmark, the decimation factor for sub-sampling and oversampling was set to 8 and the Zipfian V_t update period T_V was set to 4.

algorithm	38	91	170	251	average
MCC ***without*** *morphological post processing*					
$\Sigma\Delta$	0.495	0.347	0.169	0.282	0.323
$\Sigma\Delta$+*Zipf*	0.676	0.600	0.366	0.308	0.487
Conditional $\Sigma\Delta$	0.424	0.533	0.555	0.590	0.526
Hierarchical $\Sigma\Delta$	0.644	0.663	0.468	0.415	0.548
MCC ***with*** *morphological post processing*					
$\Sigma\Delta$	0.811	0.657	0.372	0.596	0.609
$\Sigma\Delta$+*Zipf*	0.830	0.728	0.547	0.449	0.639
Conditional $\Sigma\Delta$	0.754	0.764	0.530	0.385	0.608
Hierarchical $\Sigma\Delta$	0.816	0.827	0.686	0.582	0.728

Table 1. Results: MCC scores for 4 $\Sigma\Delta$ algorithms with/without morphological post processing

Considering first, the results *without* post morphological processing, each evolution has better results than the previous one. The best *conditional* version is obtained with Zipfian estimation combined with the conditional update of M_t and V_t. The best *hierarchical* version is obtained with the best *conditional* version combined with a conditional update of M_t^1 at low level. Considering then the results *with* morphological post processing, all results are in progression except for image # 170 that corresponds to radial movement of the first person. Both visual and numerical results enforce the use of morphological post-processing (Fig. 8) to remove remaining noise. Another benchmark, not presented here, has been done on a sequence with cars. The results were better but harder to differentiate, as such a kind of sequence is easier to segment.

4. CONCLUSION

We have presented a new hierarchical and conditional motion detection algorithm based on an evolution of previous $\Sigma\Delta$ algorithms. Preliminary results show better (visual and

Fig. 4. basic $\Sigma\Delta$

Fig. 5. $\Sigma\Delta$ + Zipf

quantitative) performances for difficult sequences with radial movement and non-rigid object. As its complexity remains low this algorithm is well suited for very light embedded systems. Future work will consider other *difficult* sequences with the presence of clutter like snow, rain or moving trees.

5. REFERENCES

[1] L. Lacassagne; A. Manzanera; J. Denoulet; A. Mérigot, "High performance motion detection: Some trends toward new embedded architectures for vision systems," *JRTIP*, october 2008.

[2] M. Piccardi, "Background subtraction techniques: a review," in *Conference on Systems, Man and Cybernetics*. IEEE, 2004, vol. 4, pp. 3099–3104.

[3] J. Richefeu A. Manzanera, "robust and computationally efficient motion detection algorithm based on sigma-delta background estimation," in *ICVGIP*. IEEE, 2004.

[4] L. Lacassagne J. Denoulet, G. Mostafaoui and A. Mérigot, "Implementing motion markov detection on general purpose processor and associative mesh," in *CAMP*. IEEE, 2005.

[5] A. Manzanera, "Sigma-delta background subtraction and the zipf law," in *CIARP*. LNCS, 2007, vol. 28-2, pp. 42–51.

Fig. 6. conditional $\Sigma\Delta$

Fig. 7. Hierarchical $\Sigma\Delta$

Fig. 8. Hierarchical $\Sigma\Delta$ + Morpho