Motion detection: Fast and robust algorithms for embedded systems
Lionel Lacassagne, Antoine Manzanera, Antoine Dupret

To cite this version:
Lionel Lacassagne, Antoine Manzanera, Antoine Dupret. Motion detection: Fast and robust algorithms for embedded systems. International Conference on Image Processing (ICIP), Nov 2009, Le Caire, Egypt. 10.1109/ICIP.2009.5413946. hal-01130889

HAL Id: hal-01130889
https://hal.archives-ouvertes.fr/hal-01130889
Submitted on 12 Mar 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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 Σ∆ 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 Σ∆ 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. Σ∆ BACKGROUND SUBTRACTION

The basic principle of the Σ∆ algorithm is to estimate parameters of the background using Σ∆ 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 \( t \) the following update formulas:

\[
\text{If } d_{t_{i-1}} < f_{t_i}, \text{ then } d_{t_i} = d_{t_{i-1}} - \varepsilon \text{ else } d_{t_i} = d_{t_{i-1}} + \varepsilon
\]

where \( \varepsilon \) is the discretization step (least significant bit) of \( d_t \).

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

2.1. Basical Σ∆ algorithm

Algorithm 1: Basical Σ∆

1. foreach pixel \( x \) do [step #1: \( M_t \) estimation]
   2. if \( M_{t-1}(x) < I_t(x) \) then \( M_t(x) \leftarrow M_{t-1}(x) + 1 \)
   3. if \( M_{t-1}(x) > I_t(x) \) then \( M_t(x) \leftarrow M_{t-1}(x) - 1 \)
   4. otherwise \( M_t(x) \leftarrow M_{t-1}(x) \)
5. foreach pixel \( x \) do [step #2: \( O_t \) computation]
   6. \( O_t(x) = |M_t(x) - I_t(x)| \)
7. foreach pixel \( x \) do [step #3: \( V_t \) update]
   8. if \( V_{t-1}(x) < N \times O_t(x) \) then \( V_t(x) \leftarrow V_{t-1}(x) + 1 \)
   9. if \( V_{t-1}(x) > N \times O_t(x) \) then \( V_t(x) \leftarrow V_{t-1}(x) - 1 \)
10. otherwise \( V_t(x) \leftarrow V_{t-1}(x) \)
    \( V_t(x) \leftarrow \max(\min(V_t(x), V_{\text{max}}), V_{\text{min}}) \)
11. foreach pixel \( x \) do [step #4: \( \hat{E}_t \) estimation]
    12. 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 Σ∆ background \( M_t \) and Σ∆ 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). \( V_{\text{min}} \) and \( V_{\text{max}} \) 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 \( \Sigma \Delta \)**

```plaintext
1 foreach pixel \( x \) with do  [step #1: conditional \( M_t \) update]
   2 if \( \hat{E}_{t-1}(x) = 0 \) then
      3 if \( M_{t-1}(x) < I_t(x) \) then \( M_t(x) \leftarrow M_{t-1}(x) + 1 \)
      4 if \( M_{t-1}(x) > I_t(x) \) then \( M_t(x) \leftarrow M_{t-1}(x) - 1 \)
      5 else \( M_t(x) \leftarrow M_{t-1}(x) \)
   8 foreach pixel \( x \) do  [step #2: \( O_t \) computation]
      9 \( O_t(x) = |M_t(x) - I_t(x)| \)
10 foreach pixel \( x \) do  [step #3: \( V_t \) update]
   11 if \( V_{t-1}(x) < N \times O_t(x) \) then \( V_t(x) \leftarrow V_{t-1}(x) + 1 \)
   12 if \( V_{t-1}(x) > N \times O_t(x) \) then \( V_t(x) \leftarrow V_{t-1}(x) - 1 \)
   13 otherwise \( V_t(x) = V_{t-1}(x) \)
   14 \( V_t(x) \leftarrow \max(\min(V_t(x), V_{\text{max}}), V_{\text{min}}) \)
15 foreach pixel \( x \) do  [step #4: \( E_t \) estimation]
   16 if \( O_t(x) < V_t(x) \) then \( E_t(x) \leftarrow 0 \) else \( E_t(x) \leftarrow 1 \)
```

**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 \( t \); \( \rho \) is the value of the index modulo \( 2^m \) (\( m \) is the number of bits of the representation). \( \pi \) is the value of the greatest power of 2 which divides \( \rho \). Finally, the threshold \( \sigma \) is equal to \( 2^m \) divided by \( \pi \). The result is that pixels \( x \) such that \( V_t(x) > 2^{m-\pi} \) 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**

```plaintext
1 [step #0: variance threshold computation]
2 find the greatest \( 2^k \) that divides \( t \ mod \ 2^m \)
3 set \( \sigma = 2^{m-2^k} \)
4 foreach pixel \( x \) do  [step #1: conditional \( M_t \) estimation]
   5 if \( V_{t-1}(x) > \sigma \) then
      6 if \( M_{t-1}(x) < I_t(x) \) then \( M_t(x) \leftarrow M_{t-1}(x) + 1 \)
      7 if \( M_{t-1}(x) > I_t(x) \) then \( M_t(x) \leftarrow M_{t-1}(x) - 1 \)
      8 else \( M_t(x) \leftarrow M_{t-1}(x) \)
   11 [foreach pixel \( x \) do  [step #2: \( O_t \) computation]
      12 \( O_t(x) = |M_t(x) - I_t(x)| \)
13 foreach pixel \( x \) do  [step #3: update \( V_t \) every \( T_V \) frames]
   14 if \( t \ mod \ T_V = 0 \) then
      15 if \( V_{t-1}(x) < N \times O_t(x) \) then \( V_t(x) \leftarrow V_{t-1}(x) + 1 \)
      16 if \( V_{t-1}(x) > N \times O_t(x) \) then \( V_t(x) \leftarrow V_{t-1}(x) - 1 \)
      17 otherwise \( V_t(x) = V_{t-1}(x) \)
   19 foreach pixel \( x \) do  [step #4: \( E_t \) estimation]
      20 if \( O_t(x) < V_t(x) \) then \( E_t(x) \leftarrow 0 \) else \( E_t(x) \leftarrow 1 \)
```

### 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: \( M^1_t \) and \( V^1_t \) are updated depending on \( \hat{E}^1_{t-1} \). At the high level, it is a conditional spatial update: \( M^0_t \) and \( V^0_t \) are updated depending on \( \hat{E}^0_{t-1} \), the oversampling binary mask of \( \hat{E}^1_t \). The subsampling factor is in the range \( [2, 10] \) and is set accordingingly 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 $3 \times 3$ morphological closing.

3. BENCHMARK

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.

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

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 subsampling and oversampling was set to 8 and the Zipfian $V_t$ update period $T_{V_t}$ was set to 4.

![Fig. 3. Hall sequence, images 38, 91, 170, 251](image)

<table>
<thead>
<tr>
<th>algorithm</th>
<th>38</th>
<th>91</th>
<th>170</th>
<th>251</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma\Delta$</td>
<td>0.495</td>
<td>0.347</td>
<td>0.169</td>
<td>0.282</td>
<td>0.323</td>
</tr>
<tr>
<td>$\Sigma\Delta+$Zipf</td>
<td>0.676</td>
<td>0.600</td>
<td>0.306</td>
<td>0.308</td>
<td>0.487</td>
</tr>
<tr>
<td>Conditional $\Sigma\Delta$</td>
<td>0.424</td>
<td>0.533</td>
<td>0.555</td>
<td>0.590</td>
<td>0.526</td>
</tr>
<tr>
<td>Hierarchical $\Sigma\Delta$</td>
<td>0.644</td>
<td>0.663</td>
<td>0.468</td>
<td>0.415</td>
<td>0.548</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>algorithm</th>
<th>38</th>
<th>91</th>
<th>170</th>
<th>251</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma\Delta$</td>
<td>0.811</td>
<td>0.657</td>
<td>0.372</td>
<td>0.596</td>
<td>0.609</td>
</tr>
<tr>
<td>$\Sigma\Delta+$Zipf</td>
<td>0.830</td>
<td>0.728</td>
<td>0.547</td>
<td>0.449</td>
<td>0.639</td>
</tr>
<tr>
<td>Conditional $\Sigma\Delta$</td>
<td>0.754</td>
<td>0.764</td>
<td>0.530</td>
<td>0.385</td>
<td>0.608</td>
</tr>
<tr>
<td>Hierarchical $\Sigma\Delta$</td>
<td>0.816</td>
<td>0.827</td>
<td>0.686</td>
<td>0.582</td>
<td>0.728</td>
</tr>
</tbody>
</table>

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 $M_{1t}$ 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
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


