# $\Sigma$-$\Delta$ Background Subtraction and the Zipf Law

Antoine Manzanera

ENSTA - Elec. and Comp. Sc. lab
32 Bd Victor, 75015 Paris, France
antoine.manzanera@ensta.fr
http://www.ensta.fr/~manzaner

**Abstract.** The $\Sigma$-$\Delta$ background estimation is a simple non linear method of background subtraction based on comparison and elementary increment/decrement. We propose here some elements of justification of this method with respect to statistical estimation, compared to other recursive methods: exponential smoothing, Gaussian estimation. We point out the relation between the $\Sigma$-$\Delta$ estimation and a probabilistic model: the Zipf law. A new algorithm is proposed for computing the background/foreground classification as the pixel-level part of a motion detection algorithm. Comparative results and computational advantages of the method are commented.

**Keywords:** Image processing, Motion detection, Background subtraction, $\Sigma$-$\Delta$ modulation, Vector data parallelism.

## 1 Introduction

Background subtraction is a very popular class of methods for detecting moving objects in a scene observed by a stationary camera [1] [2] [3] [4] [5]. In every pixel $p$ of the image, the observed data is a time series $I_t(p)$, corresponding to the values taken by $p$ in the video $I$, as a function of time $t$. As only temporal processing will be considered in this paper, the argument $p$ will be discarded. The principle of background subtraction methods is to discriminate the pixels of moving objects (the foreground) from those of the static scene (the background), by detecting samples which are significantly deviating from the statistical behaviour of the time series. To do this, one needs to estimate the graylevel distribution with respect to time, i.e. $f_t(x) = P(I_t = x)$. As the conditions of the static scene are subject to changes (lighting conditions typically), $f_t$ is not stationary, and must be constantly re-estimated. For the sake of computational efficiency, which is particularly critical for video processing, it is desirable that $f_t$ should be represented by a small number of estimates which can be computed recursively. In this paper, we focus on recursive estimation of mono-modal distributions, which means that we assume that the time series corresponding to the different values possibly taken by the background along time, presents one single mode. This may not be a valid assumption for every pixel, but it does not affect the interest of the principle since the technique presented can be extended to multi-modal background estimation.

L. Rueda, D. Mery, and J. Kittler (Eds.): CIARP 2007, LNCS 4756, pp. 42–51, 2007.

The $\Sigma$-$\Delta$ background estimation [6] [7] [8] is a simple and powerful non linear background subtraction technique, which consists in incrementing (resp. decrementing) the current estimate by an constant value if it is smaller (resp greater) than the current sample. Our objective is to discuss the foundations of this method, with regards to statistical estimation. We show the relation between the $\Sigma$-$\Delta$ estimation and the probabilistic model of Zipf-Mandelbrot, and compare it with two other recursive methods: exponential smoothing and Gaussian estimation. Section 2 presents the general framework of recursive estimation. Section 3 presents the particular case of $\Sigma$-$\Delta$ estimation, and provides the full numerical algorithm to compute it in the mono-modal case. Section 4 shows some results and discuss the computational advantages of $\Sigma$-$\Delta$ background subtraction, proposing in particular a complete vector data parallel implementation adapted to the SIMD-within-register framework. Section 5 concludes and presents the possible extension of the primitive algorithm.

## 2 Recursive Estimation

If one should represent $f_t$ by one single scalar estimate, one of the most natural would be the average $M_t$ of the time series $I_t$. The naive recursive computation: $M_t = \frac{1}{t} I_t + \frac{t-1}{t} M_{t-1}$ can be used as initialisation for the small values of $t$, but is not numerically realisable for long series. So one common solution is to use a constant weight (or learning rate) $\alpha \in ]0,1[$ for the current sample: $M_t = \alpha I_t + (1-\alpha) M_{t-1}$. This is sometimes referred to as running average, and corresponds to the recursive implementation of exponential smoothing.

One way of generalising this is to write the updating equation in an incremental form: $M_t = M_{t-1} + \delta_t(I_t)$, where $\delta_t$ is the increment function, depending on the current sample $I_t$. In the case of exponential smoothing, $\delta_t$ is the affine function $\alpha(I_t - M_{t-1})$ (Fig. 1(1)). This linear dependence is not satisfying, since, in most cases, a sample which is far from the average is out of the background model and should have little effect on the estimate updating. This problem can still be addressed in the exponential smoothing framework, by using two distinct constants $\alpha_1$ and $\alpha_2$ such that $\alpha_1 > \alpha_2$, and by defining $\delta_t(I_t) = \alpha_1(I_t - M_{t-1})$ if $I_t$ is in the background model, and $\delta_t(I_t) = \alpha_2(I_t - M_{t-1})$ if $I_t$ is foreground. This results in a discontinuous increment function $\delta_t$, as shown in Fig. 1(2), where the decision background/foreground is done by simply thresholding the absolute difference: The pixel is foreground if $|I_t - M_{t-1}| > Th$. It appears however, that the discontinuity of $\delta_t$ makes the choice of $Th$ critical.

To get a more continuous behaviour, we shall follow [9], who suggests that the weight $\alpha$ attached to the current sample $I_t$ should depend on its probability $f_t(I_t)$. But, as noted by [10], the mere product $\delta_t(I_t) = f_t(I_t) \times \alpha(I_t - M_{t-1})$ suggested by [9] is not usable in practise because of increments too small in general to be numerically operative. A proper way to achieve this, if $\alpha_{max}$ is the maximal desired weight, and as $M_{t-1}$ is the mode of the current distribution $f_t$, is to use:

$$\delta_t = \frac{\alpha_{max} f_t(I_t)}{f_t(M_{t-1})} \times (I_t - M_{t-1}). \tag{1}$$

If we use a Gaussian distribution as density model like in [9], we get the following increment function:

$$\delta_t = \alpha_{max} \times \exp(\frac{-(I_t - M_{t-1})^2}{2V_{t-1}}) \times (I_t - M_{t-1}). \quad (2)$$

The model needs the temporal variance $V_t$. In [9], it is computed as the Gaussian estimation of the series $(I_t - M_t)^2$. But this leads to a double auto-reference in the definitions of $M_t$ and $V_t$, which is prejudicial to the adaptation capability of the algorithm. We recommend rather to compute $V_t$ as the exponential smoothing of $(I_t - M_t)^2$, using a fixed learning rate $\alpha_V$.

One of the interest of computing $V_t$ is that it provides a natural criterion of decision background/foreground through the condition $|I_t - M_t| > N \times \sqrt{V_t}$, with $N$ typically between 2 and 4. Note that the increment function (Fig. 1(3)) is very different from the previous ones, and has a derivative-of-Gaussian shape.

This Gaussian estimation provides some attractive features compared to the exponential smoothing: the update of the estimates depends on the probability of the current sample, and the increment values are globally higher when the background density is more dispersed. Nevertheless, it is less used than exponential smoothing because of the computational load much higher. Now, what does the increment function look like if we take the Zipf law as the probabilistic model ?

## 3 Zipfian Background Estimation

Originally the Zipf law is an empirical principle [11] at the crossroads of linguistic and information theory, stating that, in any sense-making text, the probability of occurrence of the $n^{th}$ most frequent word is $1/n$ the probability of occurrence of the (first) most frequent word. So the Zipf distribution is a hyperbolic decreasing function. Recently, it has been used in several applications of image processing [12], in particular as a model for the distribution of local spatial features. We use it here as a model for (pixel-wise) temporal distribution.

Because of the divergence of the sum $1/n$, the Zipf density function includes a power factor: $1/n^s$, with $s > 1$. The general expression of the continuous symmetric Zipf-Mandelbrot distribution can be written:

$$Z_{(\mu,k,s)}(x) = \frac{(s-1)k^{s-1}}{2(|x-\mu|+k)^s}. \quad (3)$$

In this expression, the parameter $\mu$ represents the mode of the distribution, and $k$ determines its dispersion. The remarkable property of $Z$, taken as the density model $f_t$ of the time series $I_t$ (and then, replacing eq. 3 in eq. 1), is the shape of the increment function $\delta_t$ (Fig. 1(3)), which is close to the Heaviside shaped function: $H_{(\mu,\kappa)}(x) = -\kappa$ if $x < \mu$, $+\kappa$ if $x > \mu$, with $\kappa = \alpha_{max}k^s$. Thus it is naturally related to the $\Sigma$-$\Delta$ modulation, classically used in Analog to Digital Conversion:

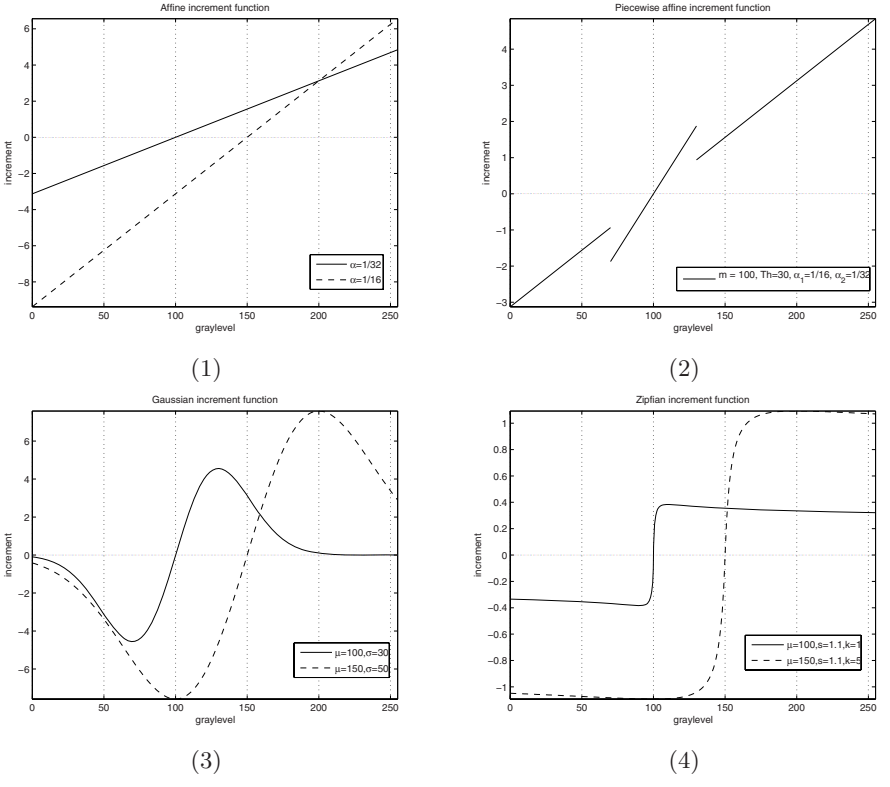


**Fig. 1.** The different increment functions $\delta_t$ associated to the different distribution models: (X axis: graylevel $I_t$, Y axis: increment value $\delta_t(I_t)$). (1) Exponential smoothing (plain: $\alpha = 1/32$; dashed: $\alpha = 1/16$) (1) Bi-level exponential smoothing ($m = 100$, $Th = 30$, $\alpha_1 = 1/16$, $\alpha_2 = 1/32$) (3) Gaussian laws, $\alpha_{max} = 1/4$ (plain: $\mu = 100$, $\sigma = 30$; dashed: $\mu = 150$, $\sigma = 50$) (4) Zipf laws, $\alpha_{max} = 1/4$ (plain: $\mu = 100$, $k = 1$, $s = 1.1$; dashed: $\mu = 150$, $k = 5$, $s = 1.1$).

For every time step $\Delta t$:
If $M_{t-\Delta t} > I_t$ then $M_t = M_{t-\Delta t} - \varepsilon$ ;
If $M_{t-\Delta t} < I_t$ then $M_t = M_{t-\Delta t} + \varepsilon$ ;

Here, the average increment per time unit is $\kappa = \frac{\varepsilon}{\Delta t}$. Digitally, the elementary increment $\varepsilon$ is the least significant bit of the representation, i.e. 1 if the values are integer-coded. Adaptation to the dispersion of the model can then be done by tuning the updating period $\Delta t$: the greater the variance, the smaller $\Delta t$ should be. The following algorithm reproduces such behaviour. The principle is to attach to every pixel, in addition to the mode estimator $M_t$, a dispersion estimator $V_t$. Suppose that $V_t \in ]0, 2^m - 1[$, which means that it is coded on $m$ bits:

For every frame $t$: {
rank = $t$ % $2^m$ ; pow2 = 1 ;
do { pow2 = 2× pow2 ; } while((rank % pow2 == 0) && (pow2 < $2^m$))

If $(V_{t-1} > \frac{2^m}{pow2})$ {
  If $M_{t-1} > I_t$ then $M_t = M_{t-1} - 1$ ;
  If $M_{t-1} < I_t$ then $M_t = M_{t-1} + 1$ ;
  }
$D_t = |I_t - M_t|$ ;
If $(t \ \% \ T_V == 0)$ {
  If $V_{t-1} > \max(V_{min}, N \times D_t)$ then $V_t = V_{t-1} - 1$ ;
  If $V_{t-1} < \min(V_{max}, N \times D_t)$ then $V_t = V_{t-1} + 1$ ;
  }
}

Here $x\%y$ is $x$ modulo $y$. The purpose of the two first lines of the algorithm (which are computed once for all the pixels at every frame) is to find the greatest power of two (pow2) that divides the time index modulo $2^m$ (rank). Once this has been determined, it is used to compute the minimal value of $V_{t-1}$ for which the $\Sigma$-$\Delta$ estimate $M_t$ will be updated. Thus the (log-)period of update of $M_t$ is inversely proportional to the (log-)dispersion: if $V_t > 2^{m-1}$, $M_t$ will be updated every frame, if $2^{m-2} \leq V_t < 2^{m-1}$, $M_t$ will be updated every 2 frames, and so on.

The dispersion factor $V_t$ is computed here as the $\Sigma$-$\Delta$ estimation of the absolute differences $D_t$, amplified by a parameter $N$. Like in Gaussian estimation, we avoid double auto-reference by updating $V_t$ at a fixed period $T_V$. $V_t$ can be used as a foreground criterion directly: the sample $I_t$ is considered foreground if $D_t > V_t$. $V_{min}$ and $V_{max}$ are simply used to control the overflows ; 2 and $2^m - 1$ are their typical values.

Note that the time constants, which represent the period response of the background estimation algorithm, are related here to the dynamics (the number of significant bits) of $V_t$, and to its updating period $T_V$. For the other methods, the time constants were associated to the inverse of the learning rates: $1/\alpha_i$ for the exponential smoothing and $1/\alpha_{max}$ and $1/\alpha_V$ for Gaussian estimation.

## 4 Results

Figure 2 shows the background estimation for all the time indexes, and one particular pixel. This is a difficult case for pixel-level motion detection: an outdoor scene where the background signal (high grass meadow with wind) is corrupted by the passage of two foreground objects. The Boolean condition "$D_t > V_t$" is used as foreground classification.

Figure 3 shows the result for all the pixels, at 4 different time indexes of the classical *Hall* sequence, in which two people are moving in radial motion, i.e. in the direction of the optical axis. This is a difficult case too, since the values in the centre of the moving objects do not change much (aperture problem). For reference, the last row of Figure 3 displays the hand drawn ground truth for the 4 frames.

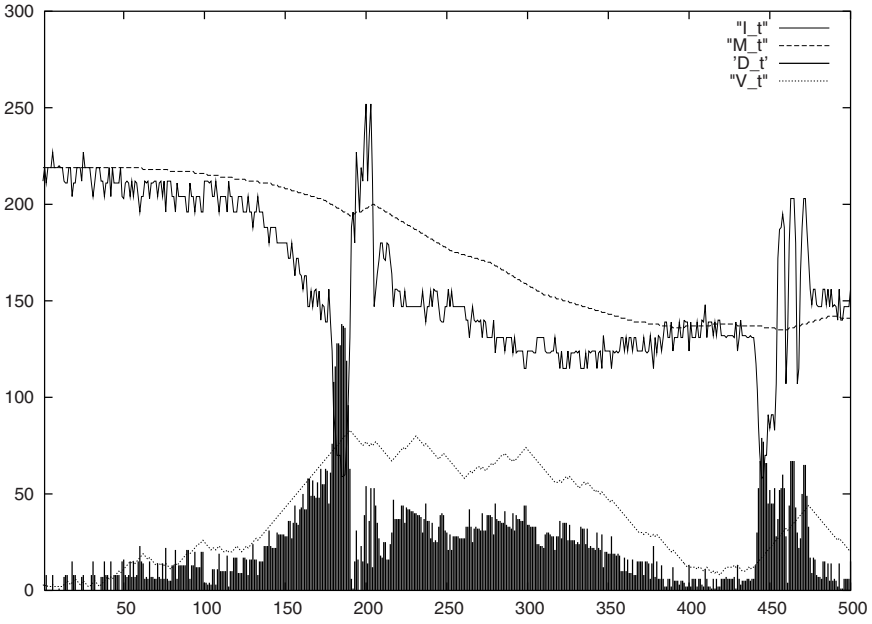


**Fig. 2.** $\Sigma$-$\Delta$ background estimation running on a given pixel. (X axis: time index, Y axis: graylevel). All values are 8-bit coded. Amplification factor $N$ is 2. Variance updating period $T_V$ is 1. Plain line: $I_t$, Dotted line: $M_t$, Impulses: $D_t$, Dashed line: $V_t$.

This ground truth is used for the quantitative evaluation (detection and false alarm rates are averaged on these 4 key frames) shown on Table 1, for different values of the amplification constant N, and of the updating period $T_V$. Those results are resumed on Figure 4, where the 9 $\Sigma$-$\Delta$ algorithms are compared with 6 different Gaussian algorithms. Note that these figures relate to pixel-level methods, and should not be interpreted in absolute, since a simple spatial regularisation appreciably improves the two measures, in all cases.

**Table 1.** (Detection, False alarm) rates for 9 $\Sigma$-$\Delta$ background subtraction algorithms. Measures are averaged on the 4 key frames of the *Hall* sequence.

| | N=1 | N=2 | N=4 |
|---|---|---|---|
| $T_V = 1$ | (0.74,0.25) | (0.62,0.10) | (0.53,0.02) |
| $T_V = 8$ | (0.91,0.38) | (0.87,0.23) | (0.85,0.12) |
| $T_V = 32$ | (0.95,0.45) | (0.94,0.38) | (0.94,0.33) |

The relevance and power of the $\Sigma$-$\Delta$ estimation, as a pixel-level temporal filter, is comparable to that of the Gaussian estimation, whereas its computational cost is even inferior to that of exponential smoothing. Indeed, the algorithm proposed in Section 3 is straightforward to compute in any fixed-point

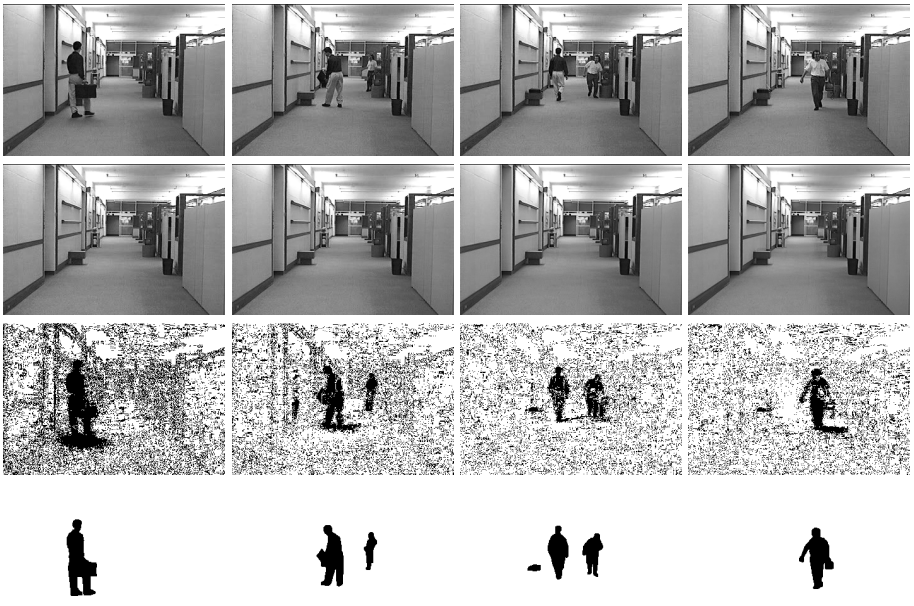

**Fig. 3.** Background subtraction shown at different frames of the *Hall* image sequence. Row 1: Original sequence, frames 38, 90, 170 and 250. Rows 2 and 3: $\Sigma$-$\Delta$ Background and foreground, with N=2, and $T_V = 8$. Row 4: (Fore)ground truth.

arithmetic, using an instruction set limited to: absolute difference, comparison, increment/decrement. Thus, it is well adapted to real-time implementation using dedicated or programmable hardware.

Another important computational property of $\Sigma$-$\Delta$ background subtraction, is that, once chosen the number of bits used to represent the estimates $M_t$ and $V_t$, every computation can be made at full precision without increasing the data width. This allows in particular to make the most of the data parallelism provided by the SIMD-WAR (Single Instruction Multiple Data Within A Register) paradigm, which consists in concatenating many short operands in one single very long variable, and then applying scalar operations on the long variables. This implementation is available on most personal computers, using for example the SSE-2 (Streaming SIMD Extensions 2) instructions of the Intel®C++ compiler [13]. We provide hereunder the vectorised pseudo-code of the $\Sigma$-$\Delta$ background subtraction. Here, a 16-times acceleration is achieved by performing the operations on a 128-bit register made of 16 8-bit data.

```
vmin = 2; vmax = 255; logN = 1; Tv = 4;// Scalar constants definition
// Vector constants definition: creates 128-bit constants
// by concatenating 16 8-bit constants
VMIN = vector16_define(vmin);
VMAX = vector16_define(vmax);
// Sigma-Delta initializations
for(i=0; i<height; i++) {
```

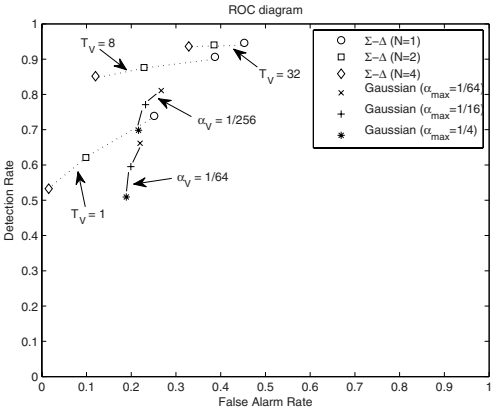


**Fig. 4.** Detection / false alarm rates diagram, for 9 $\Sigma$-$\Delta$ and 6 Gaussian background subtraction algorithms. Measures are averaged on the 4 key frames of the *Hall* sequence.

```
|  for(j=0; j<width/16; j++) {
|  |  I = I(0);// I(0): first image
|  |  M = I; // M(0) = I(0)
|  |  V = VMIN; // V(0) = vmin
|  }
}
for(t=1; t<=tstop; t+=1) {// Time loop*********************************
|  // Computation of the update threshold according to the time index
|  rank = (t%256); pow2 = 1; thres = 256;
|  do { pow2 = pow2*2; thres = thres/2;
|     } while (((rank%pow2)==0)&&(thres>1));
|  TH = vector16_define(thres);// vector variable
|  for(i=0; i<height; i++) {// Space loop------------------------------
|  |  for(j=0; j<width/16; j++) {
|  |  |  // (1) Update of Background M(t)
|  |  |  I = I(t);//loading I(t)
|  |  |  UPDATE = vector16_compare_greater(V,TH);// Comparison (>)
|  |  |  //if V(t-1)>th, update= FF (-1), else update = 0
|  |  |  C1 = vector16_compare_greater(I,M);
|  |  |  //if I(t)>M(t-1), c1= FF (-1), else c1 = 0
|  |  |  C2 = vector16_compare_less(I,M);// Comparison (<)
|  |  |  //if M(t-1)>I(t), c2= FF (-1), else c2 = 0
|  |  |  C1 = vector128_and(C1,UPDATE);// Bit-wise logical AND: Update is
|  |  |  C2 = vector128_and(C2,UPDATE);// effective only if V(t-1) > th
|  |  |  M = vector16_sub(M,C1);//M(t) = M(t-1) - c1
|  |  |  M = vector16_add(M,C2);//M(t) = M(t-1) + c2
|  |  |  // (2) Computation of absolute difference D(t)
|  |  |  MAX = vector16_max(I,M); // max(I(t),M(t))
|  |  |  MIN = vector16_min(I,M); // min(I(t),M(t))
|  |  |  D = vector16_sub(MAX,MIN); // d = |I(t) - M(t)|
|  |  |  // (3) Update of variance V(t): one over Tv frames
```

```
|  |  |  if (t % Tv == 0) {
|  |  |  |  ND = D;  // Difference amplification (Saturated addition)
|  |  |  |  for (k=1;k<=logN;k++) ND = vector16_add_sat(ND,ND);
|  |  |  |  BDEC = vector16_max(ND,VMIN);// Variance is bounded
|  |  |  |  BINC = vector16_min(ND,VMAX);// between Vmin and Vmax
|  |  |  |  C1 = vector16_compare_greater(V,BDEC);
|  |  |  |  //if V(t-1)>max(D(t),Vmin) c1= FF (-1), else c1 = 0
|  |  |  |  C2 = vector16_compare_less(V,BINC);
|  |  |  |  //if V(t-1)<min(D(t),Vmax) c2= FF (-1), else c2 = 0
|  |  |  |  V = vector16_add(V,C1);//V(t) = V(t-1) + c1
|  |  |  |  V = vector16_sub(V,C2);//V(t) = V(t-1) - c2
|  |  |  }
|  |  |  // (4) Computation of Foreground label L(t)
|  |  |  L = vector16_compare_greater(D,V);
|  |  |  //if D(t)>V(t) L(t)= FF, else L(t) = 0
|  |  }
|  }// end of space loop-----------------------------------------
}// end of time loop*********************************************
```

## 5 Conclusion and Extensions

We have proposed a justification of using the $\Sigma$-$\Delta$ estimation as a background subtraction method, based on the use of the Zipf law as a density model. We have proposed an algorithm implementing this method and allowing to adapt the background updating to the temporal dispersion. We have shown the computational advantages of the $\Sigma$-$\Delta$ estimation, illustrated by the vector SIMD implementation.

The limitations of this algorithm - used "as is" in a motion detection system - are inherent to its mono-modal nature: first, one single mode in the density model can be inefficient to discriminate moving objects over a complicated background, and second, one single dispersion estimate, related to one time constant, may not be sufficient for certain kind of motion such as remote objects with radial velocity w.r.t. the optical centre. Nevertheless the basic model can be enriched, either by using a multi-modal Zipfian distribution like it is done in [9] for Gaussian estimation, or by using several time magnitudes, as shown in [8].

## References

1. Karmann, K.P., von Brandt, A.: Moving Object Recognition Using an Adaptive Background Memory. In: Time-Varying Image Processing and Moving Object Recognition, Elsevier, Amsterdam (1990)
2. Toyoma, K., Krumm, J., Brumitt, B., Meyers, B.: Wallflower: Principles and Practice of Background Maintenance. In: Proc. IEEE ICCV, Kerkyra - Greece, pp. 255–261 (1999)
3. Elgammal, A., Harwood, D., Davis, L.: Non-parametric Model for Background Subtraction. In: Proc. IEEE ECCV, Dublin - Ireland (2000)

4. Piccardi, M.: Background subtraction techniques: a review. In: Proc. of IEEE SMC/ICSMC (October 2004)
5. Cheung, S.C., Kamath, C.: Robust techniques for background subtraction in urban traffic video. In: Proc. SPIE Video Com. and Image Proc. San Jose - CA (2004)
6. McFarlane, N., Schofield, C.: Segmentation and tracking of piglets in images. Machine Vision and Applications 8, 187–193 (1995)
7. Manzanera, A., Richefeu, J.: A robust and computationally efficient motion detection algorithm based on $\Sigma$-$\Delta$ background estimation. In: Proc. ICVGIP 2004, pp. 46–51 (December 2004)
8. Manzanera, A., Richefeu, J.: A new motion detection algorithm based on $\Sigma$-$\Delta$ background estimation. Pattern Recognition Letters 28, 320–328 (2007)
9. Stauffer, C., Grimson, E.: Learning patterns of activity using real-time tracking. IEEE Trans. on PAMI 22(8), 747–757 (2000)
10. Power, P., Schoonees, J.: Understanding background mixture models for foreground segmentation. In: Imaging and Vision Computing New Zealand, Auckland, NZ (2002)
11. Zipf, G.: Human behavior and the principle of least-effort. Addison-Wesley, New-York (1949)
12. Caron, Y., Makris, P., Vincent, N.: A method for detecting artificial objects in natural environments. In: Int. Conf. in Pattern Recognition, pp. 600–603 (2002)
13. Intel, C.: Intel®C++ Compiler for Linux Systems - User's Guide (1996-2003) Document number 253254-014