Détection de mouvement par capteur intelligent

M. Julien Richefeu & M. Antoine Manzanera
École Nationale Supérieure de Techniques Avancées
32, Boulevard Victor
75015 Paris
Julien.Richefeu at ensta.fr

Résumé

Cet article présente le portage de trois méthodes de détection du mouvement, temps réel, dans une séquence vidéo sur un système de vision à base de rétine numérique. Basé sur une architecture massivement parallèle, ce capteur intelligent permet d'effectuer des opérations élémentaires y compris pendant la phase d'acquisition de l'image. Les trois méthodes de détection du mouvement, ainsi que leur implémentation sur la rétine sont ensuite présentées. Pour chaque méthode, un effort d'explication est fait, et l'algorithme complet est fourni explicitement.

Mots Clef

détection du mouvement, système temps réel, rétine artificielle, moyenne récursive, morphologie mathématique, modulation Σ-Δ.

Abstract

This article presents the implementation of three motion detection methods, real-time, on a video sequence in vision system based on a numerical retina. Based on a massivelly parrallel architecture, this smart sensor is able to execute elementary operations during the image acquisition phase. These three motion detection methods and their implementation on the retina are then presented. For each method a particular explanation effort is made and the complete algorithm is explicitely given.

Keywords

motion detection, real-time system, artificial retina, recursive mean, mathematical morphology, Σ-Δ modulation.

1 Introduction

Alors que les systèmes classiques de vision artificielle séparent complètement capture et traitement de l'image, ces deux fonctions s'avèrent en réalité intimement associées dans les systèmes visuels biologiques. Le caractère fortement submicronique des techniques d'intégration CMOS actuelles permet justement d'introduire des circuits de traitement de l'information au sein de chaque pixel d'un imageur, qui devient alors une rétine artificielle.

La rétine numérique [10] [8] est un circuit combinant fonctions d'acquisition et de traitement des images. Elle est constituée d'une grille bidimensionnelle de "cellules", chaque cellule étant formée de l'association d'un élément photosensible (photorécepteur), et d'un élément de calcul (processeur). Du point de vue algorithmique, la rétine numérique est une machine massivement parallèle de type réseau d'automates cellulaires, caractérisée par des éléments de calcul de capacité limitée (en terme de puissance de calcul et de mémorisation). Dans un système de vision, la rétine est associée à un hôte appelé cortex, typiquement composé d'un processeur scalaire (microprocesseur basse puissance DSP) et/ou d'un circuit de logique programmable (FPGA).

Le cortex a deux fonctions : (1) contrôler la rétine, en lui envoyant les séquences de commandes; (2) effectuer les tâches de haut niveau du problème de vision à partir des informations extraites de la rétine.

Un système de vision à base de rétine programmable doit permettre d'atteindre de grandes performances en terme de vitesse, grâce au parallélisme massif de la rétine, et aussi en terme de consommation d'énergie, de par la réduction drastique du flux d'information au sein du système.

L'utilisation d'un tel système nous amène à une réflexion de fond sur les méthodes à employer : Comment sortir de l'information du circuit rétinien ? Sous quelles formes ? Quels sont les descripteurs appropriés à de tels calculs ? Comment adapter les algorithmes d'analyse d'images ? Comment mettre en place les mécanismes de communication entre la rétine et le cortex ?

De par ses avantages, le couple (rétine-cortex) est particulièrement adapté aux systèmes de vision portables, mobiles, autonomes. Cela implique que la perception du mouvement doit être une caractéristique forte du système à base de rétine. Les applications sont nombreuses et variées : systèmes de télésurveillance, capteurs de robots mobiles, systèmes de poursuite automatique, caméras compressives...

Nous présentons ici une étude sur l'implantation sur rétine programmable de trois algorithmes différents de détection du mouvement dans un contexte de caméra fixe. Les contraintes liées à ce système ont orienté les choix algorithmiques vers des méthodes récursives afin de réduire le coût en terme de mémoire utilisée et vers des méthodes disposant d'une grande adaptabilité (globale (temporelle) pour les deux premières et locale (spatiotemporelle) pour

la dernière méthode).
Après avoir présenté brièvement la rétine programmable et le cadre général du problème de détection du mouvement, nous verrons trois exemples de ces techniques, leurs principes et leurs intérêts puis une étude de leur implémentation rétinienne.
Nous présenterons d'abord une technique basée sur une moyenne récursive classique, puis une nouvelle méthode de différentiation fondée sur le filtrage morphologique oublieux, une combinaison d'opérations linéaires et morphologiques. Nous verrons son principe et son implémentation sur système à base de rétine numérique. Nous présenterons ensuite une technique s'appuyant sur une analogie de la modulation Σ-Δ en électronique. Ce filtrage très simple dans son principe nous permet d'aborder des problèmes de traitement d'images de plus haut niveau. Enfin, nous donnerons quelques exemples de techniques d'adaptation automatique d'un niveau de seuillage et de traitements spatiaux et nous conclurons sur les perspectives qu'apportent un tel système et ses performances.

2 Algorithmique de la rétine numérique

Comme nous l'avons vu précédemment les rétines sont adaptées à des applications portables, mobiles ou autonomes mais la petite capacité des éléments de calcul, et en particulier leur faible mémoire (typiquement quelques dizaines de bits par processseur cellulaire) constitue un réel défi pour la perception du mouvement. Cependant, parce que dans un système à base de rétine l'algorithmique débute dès l'acquisition et se poursuit dans la manière de coder une donnée et de l'extraire, c'est toute la méthodologie de conception des algorithmes qui se trouve transformée, en particulier par l'adaptation très fine à la dynamique tonale et temporelle rendue possible par la rétine programmable.
La rétine est une grille de processeurs à topologie régulière dotée d'une architecture massivement parallèle. Chaque cellule de la rétine exécute à chaque cycle d'horloge la même instruction élémentaire : soit une opération booléenne entre deux de ses registres mémoire; soit une translation élementaire vers un de ses quatre voisins.
L'acquisition d'une image numérique en n niveaux de gris s'effectue sur la rétine grâce à la lecture multiple du photorécepteur qui fournit $(n-1)$ coupes binaires au cours du temps (Figure 1). En effet, le photorécepteur (photodiode) se décharge d'autant plus vite (croissance hyperbolique) qu'il reçoit de photons. On effectue alors pour chaque pixel, un comptage sur $log(n)$ bits du nombre de coupes où le pixel vaut 1.
La technique de comptage la plus rapide sur le circuit consiste à utiliser un *compteur de Gray*. En effet, le principe d'un tel compteur est que les codages de deux nombres consécutifs ne diffèrent que d'un seul bit et donc chaque incrément ne coûte qu'un seul XOR. Si l'image est représentée par ses coupes $(C_i)_{1\leq i\leq n-1}$ et si l'on note

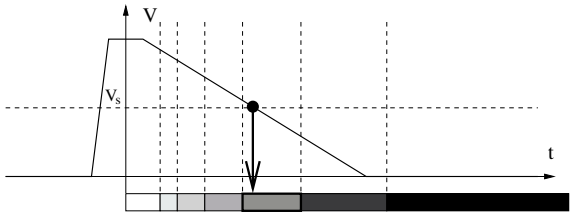


Figure 1: Acquisition d'un niveau de gris par la lecture à différents instants au cours de la décharge de la photodiode.

$Z(p)$ la plus grande puissance de 2 qui divise un naturel p, le codage de Gray du niveau de gris est défini par ses chiffres binaires $(G_i)_{0\leq i\leq log(n-1)}$ avec :

$$G_j = \bigoplus_{Z(i)=2^j} C_i \qquad (1)$$

Où $\bigoplus$ représente l'imparité. On peut en outre passer facilement d'un codage Gray au codage naturel comme le montre la figure 2.

//Acquisition de l'image et codage en code Gray
Pour $(i = 1; i < n; i++)$
{
//Seuillage du niveau i
$X_i = ACQ(i)$
//Calcul de l'index: plus grande puissance de 2 qui divise i
$index = 0$
$x = i$
Tantque $((x\%2) == 0)$ **faire** $\{index++; x = x/2\}$
$G_{index} = XOR(G_{index}, X_i)$
}
//Passage en code naturel
$N_{log(n)} = G_{log(n)}$;
Pour $(i = log(n) - 1; i \geq 0; i--)$
$X_i = XOR(G_i, X_{i+1})$;

Figure 2: Algorithme d'acquisition d'une image en niveau de gris.

Le fait que l'algorithmique intervient aussi précocement dans les rétines programmables permet naturellement d'effectuer des traitements pendant l'acquisition de l'image, en effectuant des calculs à chaque ensemble de niveaux. Le cadre formel le plus adapté est donc celui des opérations morphologiques, grâce à l'équivalence entre morphologie en niveau de gris et morphologie ensembliste sur chaque ensemble de niveau. En outre, on peut intervenir sur la dynamique de l'image, en modifiant les instants de lecture de la photodiode (compression logarithmique par exemple) ou encore en adaptant les instants de lecture à la répartition des niveaux de gris (i.e. égalisation d'histogramme).

3 Présentation du problème général de détection du mouvement

Le défi de la détection du mouvement dans le cas de capteurs fixes tient dans la capacité d'effectuer une bonne segmentation des objets en mouvement indépendamment de leurs tailles, de leurs vélocités, ou de leurs contrastes par rapport au fond.
Dans un tel contexte, l'élaboration des algorithmes de détection du mouvement se fait en tenant compte des contraintes suivantes :

- Le système doit être en mesure de fonctionner sans intervention humaine pendant un long moment, et être capable de prendre en compte des changements graduels ou soudains tels que les variations d'illumination ou la présence de nouveaux objets statiques dans la scène. Le système doit donc être *temporellement adaptatif*;

- Le système doit être capable d'éliminer tout mouvement inintéressant tels que le bruit résultant d'un cours d'eau ou de hautes herbes agitées par le vent. Il doit être robuste à de petits mouvements du capteur. Il doit donc y avoir une estimation *locale* de la confiance dans la valeur estimée du fond;

- Le système doit être temps réel, compact et basse consommation. Les algorithmes ne doivent pas consommer trop de ressources, en terme de *temps de calcul* et de *capacité mémoire*.

Les deux premières conditions impliquent que les mesures statistiques de l'activité temporelle doivent être effectuées localement en chaque pixel, et constamment remises à jour. Cela exclu toutes les approches utilisant des modèles simples de différence trame à trame, de seuillage global et les méthodes utilisant une moyenne comme seuil de décision.
Un grand nombre d'algorithmes de détection du mouvement a déjà été proposé. Nous pouvons les classer en quatre grandes catégories en fonction des types de calculs inter-trame effectués, c'est-à-dire en fonction de la façon dont la différentiation temporelle est menée.
La première est basée sur le *calcul d'un gradient temporel* : une mesure de vraisemblance du mouvement est mesurée par le changement instantané calculé entre chaque trame consécutive [4]. Ces méthodes sont naturellement adaptatives aux changements d'environnements, mais sont aussi dépendantes de la vitesse et de la taille des objets en mouvements. Ce défaut peut être minimisé en utilisant des combinaisons de filtres spatiotemporelles mais au prix d'une complexité croissante.
La seconde catégorie est basée sur des techniques de *différence au fond* [7] [19] [5] [12] [18] qui utilisent une image de référence (le fond), représentant les éléments stationnaires de la scène. Ici la mesure de vraisemblance du mouvement est la différence entre la trame courante et le fond. Ces méthodes sont moins dépendantes de la vitesse et de la taille des objets. Cependant, l'adaptation aux environnements dynamiques est une tâche bien plus ardue, pénalisant la détection des mouvements de faible amplitude (des objets petits ou très lents ou très peu contrastés).
La troisième approche est basée sur le calcul de la *vitesse locale apparente* (flot optique) [2] qui est utilisé en entrée de la segmentation spatiale [13]. Ces méthodes engendrent des informations riches mais sont généralement plus complexes à calculer et sont aussi plus sensibles à la fiabilité des résultats issus du flot optique. Ainsi un compromis doit-être trouvé entre le calcul du flot optique et la précision de la segmentation.
Enfin, plus récemment, des filtres morphologiques ont été employés [6] [16] [1] afin d'analyser des séquences vidéos. En utilisant un élément structurant spatiotemporel, une amplitude de variation locale peut-être calculée comme mesure de vraisemblance du mouvement. Une telle mesure peut être utile pour détecter de petites amplitudes de mouvement, mais comme elle est sensible au bruit spatial, elle est souvent intégrée sur des régions entière en utilisant des opérateurs connexes.
Nous allons maintenant présenter les trois méthodes de détection du mouvement (moyenne récursive, gradient morphologique oublieux et moyenne Σ-Δ) que nous avons implantées sur la rétine numérique.

4 Moyenne récursive

4.1 Principe

Le premier exemple présenté est une technique de détection du mouvement basée sur un calcul d'estimation du fond par moyenne récursive [7]. Soit I_t, l'image acquise à l'instant t, M_t le fond courant et $\alpha \in [0, 1]$ une constante; le fond est calculé récursivement à l'aide de la formule suivante :

$$M_t = \alpha I_t + (1 - \alpha) M_{t-1} \quad (2)$$
$$= \alpha(I_t - M_{t-1}) + M_{t-1} \quad (3)$$

En pratique, pour éviter de multiplier les erreurs d'arrondis dues aux multiplications par α, on utilise la seconde formule. La Figure 3 montre les résultats du calcul de fond récursif pour un pixel particulier correspondant à une zone de passage d'un objet en mouvement. On remarque que la moyenne suit le signal selon une croissance ou une décroissance exponentielle.

4.2 Implémentation

A des fins d'implantation rapide, on limite la division à un simple décalage en prenant $\alpha = 2^{-k}$. Ainsi pour une image acquise en n bits, l'intervalle de valeur pour lequel le calcul ait un sens est $\alpha \in [2^{-(n-1)}, 1]$. La multiplication par α revient alors à effectuer un décalage à droite de k bits.

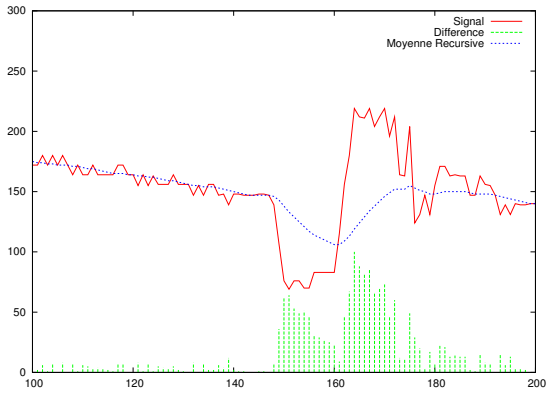


Figure 3: Calcul du fond par moyenne récursive. La courbe rouge correspond à la séquence originale, la courbe pointillée bleue au fond et la courbe en tiret vert à la différence au fond ($\alpha = 16$).

Notons que grâce au bit de signe, lorsqu'on ajoute un nombre positif inférieur à 2^k, l'effet est nul alors que lorsqu'on ajoute un nombre négatif quelconque, on décrémente toujours au moins de 1. Nous menons donc le calcul une trame sur deux en ajoutant $\alpha * (X - M)$ et une trame sur deux en soustrayant $\alpha * (M - X)$ (voir figure 4). Ce calcul est mathématiquement identique mais permet de tenir compte du problème lié aux arrondis. Le fait d'intervertir le sens de la différence une trame sur deux permet statistiquement de pouvoir ajuster au mieux le calcul du fond. En effet, à chaque instant t, la probabilité que l'image acquise X soit supérieure à la moyenne récursive M est égale à celle que X soit inférieure à M.

Pour chaque trame t {
  si ((t mod 2)=0) alors{
    X = X - M ;
    M = M + α* X ;
  }**sinon**{
    X = M - X ;
    M = M - α* X ;
  }
}

Figure 4: Algorithme de calcul de la moyenne récursive.

Le calcul d'une différence signée codée en complément à deux (S étant le bit de signe), majoritairement utilisée par la moyenne récursive, sur le circuit rétinien s'établit comme suit pour des images codées sur la rétine sur 8 bits :

```
//Difference signee
S=1;
for (i=0;i<=7;i++) {
      A = XNOR(X_i,M_i);
      B = AND(X_i,NOT(M_i));
      X_i = XOR(A,S);
      A = AND(A,S);
      S = OR(A,B);
}
S=NOT(S);
```

Avec A, B des variables temporaires et X_i et M_i des vecteurs correspondants aux valeurs de X et M sur la rétine.
Le calcul effectif de la moyenne M_i s'effectue alors comme suit, sachant que A et B sont ici aussi des variables temporaires et que S représente le bit de signe :

```
//Calcul de la moyenne M en ajoutant X
//multipliee par 2^(-k) a l'ancienne
//valeur de la moyenne,
//on introduit le bit de signe

C=0;
for (i=0;i<=7;i++) {
  if (i<=7-k) {
    A = XOR(M_i,X_i+k);
    B = AND(M_i,X_i+k);
    }else{
    A = XOR(M_i,S);
    B = AND(M_i,S);
    }
  M_i = XOR(A,C);
  A = AND(A,C);
  S = OR(A,B);
}
```

Le code rétinien correspondant au calcul de la valeur absolue de X (S est toujours le bit de signe) s'écrit comme suit :

```
//Valeur Absolue
B = AND(S,X_0);
for (i=1;i<=7;i++) {
   X_i = XOR(X_i,B);
   A  = AND(S,X_i);
   B = OR(B,A);
}
```

Enfin, afin de décider quels sont les pixels en mouvement, un seuillage global de la valeur absolue de la différence entre X et M doit être effectué. Le code suivant nous montre comment effectuer le seuillage de l'image X (codée sur la rétine) par rapport à un seuil global th (codé sur le cortex) :

```
// Seuillage de X a la valeur th
T=1;
for (i=0;i<=7;i++) {
   if (th[i])==0) {
      T=OR(T,X_i)
   }else{
      T=AND(T,X_i)
   }
}
```

Avec T l'image binaire représentant le résultat seuillé ($T = 1$ si et seulement si $X \geq th$).

5 Gradient temporel morphologique oublieux

5.1 Principe

Le second exemple présenté est un opérateur différentiel basé sur un filtre hybride combinant opérations linéaires et morphologiques [14]. Il estime récursivement une amplitude de variation en chaque pixel. Cette méthode permet de détecter efficacement les mouvements de faible amplitude (objets lents, petits ou faiblement contrastés) avec une relativement bonne résistance au bruit grâce à la partie linéaire du filtre.
En utilisant le paramètre α, un nombre réel compris entre 0 et 1, la dilatation (resp. érosion) temporelle oublieuse M_t (resp. m_t) est définie comme le montre la Figure 5. Comme pour la moyenne récursive classique, l'inverse de α a la dimension du temps. La sémantique de $M_t(x)$ (resp. $m_t(x)$) est alors la valeur maximale (resp. minimale) (estimée) observée au pixel x pendant les $1/\alpha$ dernières trames. Pour α tendant vers un, M_t (resp. m_t) tend vers I_t, et pour α tendant vers zéro, M_t (resp. m_t) tend vers la valeur maximale (resp. minimale) observée durant toute la séquence.

Initialisation
Pour chaque pixel x:
$M_0(x) = m_0(x) = I_0(x)$
Pour chaque trame t
Pour chaque pixel x:
$M_t(x) = \alpha I_t(x) + (1-\alpha) max(I_t(x), M_{t-1}(x))$
$m_t(x) = \alpha I_t(x) + (1-\alpha) min(I_t(x), m_{t-1}(x))$
Pour chaque trame t
Pour chaque pixel x:
$\Gamma_t(x) = M_t(x) - m_t(x)$

Figure 5: Algorithme de calcul des opérateurs morphologiques oublieux.

Le gradient temporel morphologique oublieux Γ_t représente ainsi une mesure de vraisemblance du mouvement. Grâce à l'intégration récursive sur plusieurs trames, ce filtre permet de détecter les mouvements dont l'amplitude se trouve en dessous de la discrétisation spatiotemporelle.
La Figure 6 montre les résultats du filtrage morphologique oublieux pour le même pixel de la figure 3. Les courbes correspondant à l'érosion morphologique oublieuse et à la dilatation morphologique oublieuse forment une sorte d'enveloppe autour du signal et rendent ainsi compte des changements, y compris les plus faibles. La figure 7 illustre les différents opérateurs morphologiques oublieux et leurs homologues classiques sur une séquence connue.

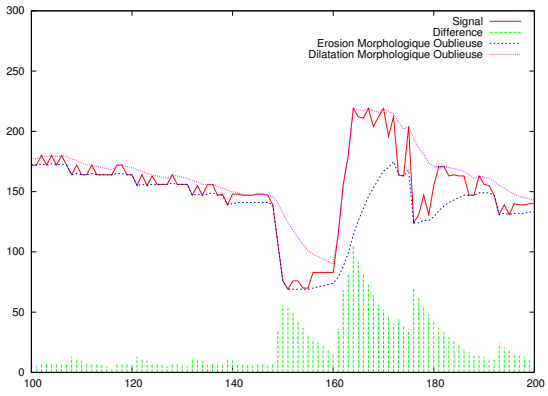


Figure 6: Filtrage morphologique oublieux. La courbe rouge correspond à la séquence originale, la courbe violette au maximum oublieux, la courbe pointillée bleue au minimum oublieux et la courbe en tiret vert le gradient morphologique oublieux ($\alpha = 8$).

5.2 Implémentation

Afin de mener le calcul du gradient morphologique oublieux sur la rétine numérique, nous reprenons le principe du filtre précedent. En effet, le calcul des minima et maxima revient à calculer deux moyennes récursives en utilisant en plus un outil de comparaison. La constante α a la même dimension que dans le calcul de la moyenne récursive. De plus, ici on connaît le signe des différences entre l'image acquise X et les minima m et maxima M . En effet, $X - M$ et $m - X$ sont négatifs et l'on peut donc effectuer les opérations sans problèmes d'arrondis comme le montre la figure 8.
La comparaison entre l'image acquise X, le maxima MAX et le minima MIN consiste à utiliser deux variables E et F ; E (resp. F) vaut 1 si et seulement si X est inférieure à MIN (resp. X est supérieure à MAX). On utilise deux bascules D et G afin de savoir si le bit précédent était déjà inférieur ou supérieur à celui de X. Dans le code rétine suivant, A, B et C sont des variables temporaires :

```
// Comparaison entre X, MAX et MIN
D=0; E=0; F=0; G=0;
for (i=7;i>=0;i--) {
   A = AND(NOT(X_i),MIN_i);
   B = AND(X_i,NOT(MIN_i));
   C = AND(A,NOT(D));
   E = OR(E,C);
   A = OR(A,B);
   D = OR(A,D);
   A = AND(X_i,NOT(MAX_i));
   B = AND(NOT(X_i),MAX_i);
   C = AND(A,NOT(G));
   F = OR(F,C);
   A = OR(A,B);
   G = OR(A,B); }
```

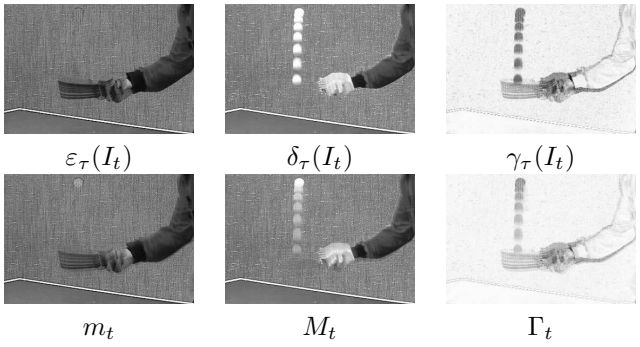


Figure 7: Application des opérateurs morphologique oublieux (en bas), comparés aux opérateurs morphologiques classiques (en haut), calculés à la trame $t = 19$ de la séquence "Tennis" (Berkeley). Pour comparaison, les opérateurs morphologiques classiques utilisent pour élément structurant le segment dans la dimension temporelle $\tau = [-8, 0]$; et les opérateurs morphologiques oublieux $\alpha = 1/9$. Les gradients sont affichés en mode vidéo inversé.

Pour chaque trame t {
M = MAX(X,M) ;
m = min(X,m) ;
Y = m - X ;
X = X - M ;
m = m - $\alpha * Y$;
M = M + $\alpha * X$;
X = M - m :
}

Figure 8: calcul du gradient morphologique oublieux

Il ne nous reste plus qu'à mettre à jour les valeurs de MIN et MAX en fonction des valeurs respectives de E et F. On peut noter que c'est le seul moment où MIN peut décroître et MAX peut croître puisqu'en effet dans la suite nous ne faisons que soustraire des nombres négatifs au minimum et ajouter des nombres négatifs au maximum. Ici encore B et C sont des variables temporaires :

```
// Calcul effectif de MIN et MAX
for (i=0;i<=7;i++) {
   B = AND(E,X_i);
   C = AND(NOT(E),MIN_i);
   MIN_i = OR(B,C);
   B = AND(F,X_i);
   C = AND(NOT(F),MAX_i);
   MAX_i = OR(B,C);
}
```

Enfin, de la même manière que nous l'avons fait pour la moyenne récursive, il nous faut seuiller la différence $X = MAX - MIN$ en chaque pixel afin de décider quels sont ceux pour lesquels un mouvement est détecté.

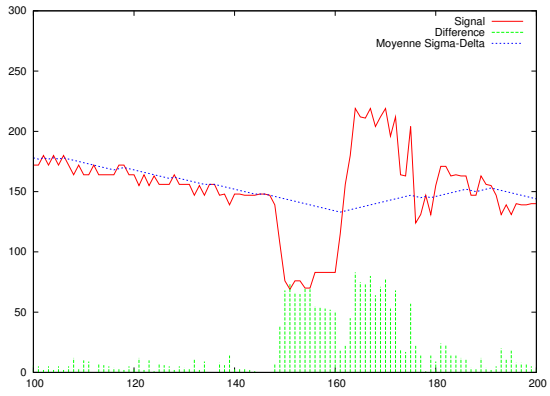


Figure 9: Filtrage Σ-Δ. La courbe rouge correspond à la séquence originale, la courbe pointillée bleue à la moyenne Σ-Δ et la courbe en tiret vert à la différence à la moyenne.

6 Moyenne $\Sigma - \Delta$

6.1 Principe

Dans le dernier algorithme [9], l'estimation du fond est vue comme la simulation d'une conversion numérique d'un signal analogique en utilisant la modulation Σ-Δ qui utilise seulement des comparaisons et des incrémentations/décrémentations élémentaires (Table 1(1). Comme la précision de la modulation Σ-Δ est limitée aux signaux dont la dérivée temporelle absolue est inférieure à un, l'erreur de modulation est proportionnelle au taux de variation du signal, correspondant ici à la mesure de vraisemblance en chaque pixel. Nous utilisons donc les différences absolues entre I_t et M_t comme premier estimateur: la différence Δ_t (Table 1(2)).
On utilise aussi ce filtre pour calculer la variance temporelle du pixel, représentant sa mesure d'activité, utilisée pour décider si le pixel est plutôt mobile ou fixe. Donc le second estimateur V_t (Tableau 1(3)) utilisé dans la méthode a la dimension d'un écart-type temporel calculé comme le filtre Σ-Δ de la séquence des différences Δ_t. Comme l'on s'intéresse aux pixels dont le taux de variation est significativement supérieur à leur activité temporelle, on applique le filtre Σ-Δ à N fois les différences non nulles.
La Figure 9 montre les résultats du filtrage Σ-Δ, toujours sur le même pixel. On peut remarquer comment le comportement non linéaire du filtre lui confère une plus grande robustesse.
Finalement, la détection au niveau pixel est effectué par comparaison entre Δ_t et V_t (Tableau 1(4)).
La figure 10 montre le résultat de cette méthode pour une trame prise dans une scène de traffic urbain.

6.2 Implémentation

La première étape de la moyenne Σ-Δ consiste à effectuer la différence signée entre l'image acquise X et la moyenne M. Ensuite, comme pour le calcul des minima et maxima du gradient morphologique oublieux, on regarde le signe

Pour chaque trame t
pour chaque pixel x:
$\Delta_t(x) = M_t(x) - I_t(x)$

(1)

Initialisation
pour chaque pixel x:
$M_0(x) = I_0(x)$
Pour chaque trame t
pour chaque pixel x:
if $\Delta_t(x) < 0$, $M_t(x) = M_{t-1}(x) + 1$
if $\Delta_t(x) > 0$, $M_t(x) = M_{t-1}(x) - 1$

(2)

Initialisation
pour chaque pixel x:
$V_0(x) = 0$
Pour chaque trame t
pour chaque pixel x tel que $\Delta_t(x) \neq 0$:
if $V_{t-1}(x) < N \times |\Delta_t(x)|$, $V_t(x) = V_{t-1}(x) + 1$
if $V_{t-1}(x) > N \times |\Delta_t(x)|$, $V_t(x) = V_{t-1}(x) - 1$

(3)

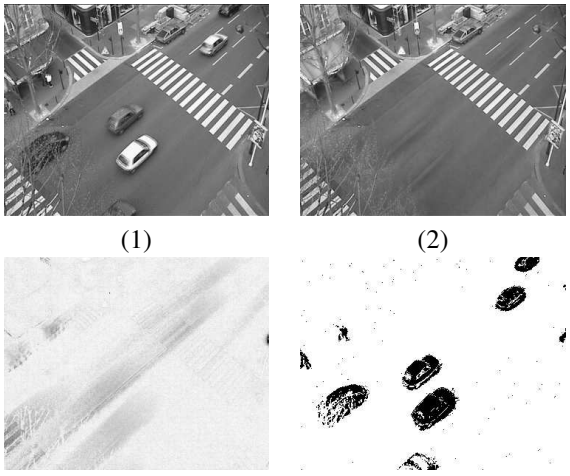
pour chaque trame t
pour chaque pixel x:
si $|\Delta_t(x)| < V_t(x)$
alors $D_t(x) = 0$
sinon $D_t(x) = 1$

(4)

Table 1: L'estimation du fond par filtre Σ-Δ : (1) Calcul de la différence entre l'image originale et la moyenne. (2) Calcul de la moyenne Σ-Δ. Σ-Δ (mesure de vraisemblance du mouvement). (3) Calcul de la variance Σ-Δ définie par la moyenne Σ-Δ de N fois les différences non nulles. (4) Calcul des étiquettes de mouvement par comparaison entre la différence et la variance.

de Δ afin de savoir si X est supérieure, inférieure ou égale à M. C'est le rôle que vont avoir les variables E et F dans le code suivant :

```
// Comparaison entre X et M
D=0;
E=0;
F=0;
for {i=7;i>=0;i--) {
  A = AND(M_i,NOT(X_i));
  B = AND(X_i,NOT(M_i));
  G = AND(A,NOT(D));
  E = OR(E,G);
  G = AND(B,NOT(D));
  F = OR(F,G);
  A = OR(A,B);
  D = OR(D,A);
}
```



(1) (2) (3) (4)

Figure 10: Résultat de l'algorithme pour une séquence de traffic urbain. (1) I_t (2) M_t. (3) V_t (visualisé avec une égalisation d'histogramme). (4) D_t (N=2).

Avec D une bascule et G une variable temporaire. A la fin de ce calcul E vaut 1 si et seulement si $X < M$ et F vaut 1 si et seulement si $M < X$. Il nous reste à mettre à jour la moyenne à l'aide des valeurs de E et F. Ainsi on décremente M de E et on incrémente M de F :

```
// Mise a jour de M
// decrementation de E
C = AND(NOT(M_0),E);
M_0 = XOR(M_0,E);
for {i=1;i<=7;i++) {
  M_i = XOR(M_i,C);
  C = AND(C,M_i);
}
// incrementation de F
C = AND(M_0,F);
M_0 = XOR(M_0,F);
for {i=1;i<=7;i++) {
  M_i = XOR(M_i,C);
  C = AND(C,NOT(M_i));
}
```

A la différence des deux méthodes précédentes, on utilise ici uniquement un seuillage local, correspondant au résultat de la comparaison entre $|\Delta_t|$ et V_t.

7 Adaptation et traitements spatiaux

Pour les deux premiers algorithmes (moyenne récursive et gradient morphologique oublieux), nous utilisons un seuil global (i.e. le même pour tous les pixels). La valeur de ce seuil doit nécessairement être ajustée dynamiquement si l'on souhaite que la détection soit adaptative.

Nous utilisons le dialogue rétine-cortex pour adapter dynamiquement ce seuil en fonction d'une estimation de la distribution spatiale des différences. Pour réduire au maximum le flux d'information, nous utilisons une somme sur une image binaire, en appliquant récursivement l'ajustement suivant : après avoir seuillé l'image à la valeur τ, on calcule sur la rétine l'ensemble des points isolés (voir code rétine ci-dessous), puis on somme sur le cortex l'ensemble ainsi obtenu. Si la proportion de points isolés est inférieure à un taux fixé ρ (typiquement de l'ordre de 1%), on décrémente τ, sinon on l'incrémente.

```
// Detecteur de Points isoles
 P = NOT(P)
 P1 = AND(P.Nord, P.Est)
 P2 = P1.Sud
 P1 = AND(P1, P2.Ouest)
 P2 = AND(P.Nord, P.Sud)
 P2 = AND(P2.Est, P2.Ouest)
 P1 = AND(P1, P2)
 D = AND(NOT(P), P1)
```

La justification de cet ajustement repose sur l'hypothèse qu'un point isolé est forcément du bruit et donc que le comptage de tels points fournit une estimation de ce bruit.
Un tel mécanisme peut également être utilisé dans le cas du troisième algorithme (moyenne Σ-Δ) pour positionner automatiquement le paramètre N du calcul de V_t.
Enfin, il convient de préciser que de telles mesures globales peuvent êtres calculées efficacement sur la rétine en utilisant des opérateurs intégraux analogiques tels que des mesures de courant consommé [3].
Un fois la détection purement temporelle D_t calculée, on la régularise en exploitant les corrélations spatiales grâce au filtrage suivant : on élimine les petites composantes connexes de D_t en utilisant l'ouverture par reconstruction :

$$L_t^{(0)} = Rec^{D_t}(\varepsilon_{B_\lambda}(D_t)) \tag{4}$$

où ε est l'érosion morphologique, B_λ (l'élément structurant), est une boule de rayon λ et $Rec^Y(X)$ est la reconstruction géodésique de X dans Y.
Finalement, on effectue une confirmation temporelle en calculant une autre reconstruction :

$$L_t = Rec^{L_t^{(0)}}(L_{t-1}^{(0)}) \tag{5}$$

L_t représente l'étiquette finale, le résultat de la détection. Sa sémantique est : les objets "plus gros que" λ qui apparaissent sur deux trames consécutives.
Pour effectuer une érosion par la boule élémentaire 4-connexe sur la rétine, il suffit de prendre la conjonction des quatre voisins de chaque pixel P :

```
// Erosion 8-connexe
P = AND(P,P.Nord)
P = AND(P,P.Est)
P = AND(P,P.Sud
P = AND(P,P.Ouest)
```

	ALGORITHME		
	Acquisition		
INS	262		
REG	9		
	MOYREC	*MORPHO*	Σ-Δ
INS	88	303	312
REG	18	36	28
	SEUIL GLOBAL		
INS	8		
REG	9		
	FILTRAGE SPATIAL		
INS	66		
REG	5		
	TOTAL		
	MOYREC	**MORPHO**	Σ-Δ
INS	**424**	**639**	**640**
REG	**23**	**41**	**32**

Figure 11: Tableau comparatif des trois algorithmes (**MOYREC** pour moyenne récursive, **MORPHO** pour gradient temporel morphologique oublieux et Σ-Δ pour la moyenne Σ-Δ) présentés en terme de temps de calcul (nombre d'instructions (**INS**)) et d'utilisation de l'espace mémoire (nombre de registres utilisés (**REG**)). Ainsi pour une rétine numérique cadencée à 1 Mhz (fréquence d'instructions), le temps de calcul par trame nécessaire pour toute la détection est inférieur à 0,45 ms pour la moyenne récursive et inférieur à 0,65 ms pour le gradient temporel morphologique oublieux et pour la moyenne Σ-Δ.

Et enfin, ce dernier code permet d'effectuer la reconstruction du résultat P dans la détection courante (F) :

```
// Reconstruction
P = AND(P,F);
for (i=0;i<N;i++) {
  P = OR(P,P.Ouest);
  P = OR(P,P.Est);
  P = OR(P,P.Sud);
  P = OR(P,P.Nord);
  P = AND(P,F);
}
```

Le modèle de rétine existant actuellement étant une machine parallèle purement synchrone, on utilise dans la reconstruction un nombre fixe d'itérations (N assez grand) au lieu d'une véritable relaxation. Néanmoins, la prochaine génération de rétine intègrera des opérateurs asynchrones qui permettront le calcul de la reconstruction géodésique en temps quasi constant.

8 Conclusion

Nous avons présenté trois exemples de calcul de détection du mouvement sur système de vision à base de rétine

numérique. Nous avons mis en avant cette méthodologie particulière de conception d'algorithmes sur un tel système qui prend en compte la quantité limitée de mémoire disponible ainsi que les contraintes du calcul cellulaire.
Une évaluation comparative de la qualité de détection des différents algorithmes reste à faire. Néanmoins nous avons noté expérimentalement les faits suivants :

- La méthode basée sur le calcul d'une moyenne récursive est moins coûteuse mais moins adaptative et moins précise en terme de localisation;
- La méthode basée sur les opérateurs de morphologie oublieux offre une plus grande capacité de détection mais est moins précise dans la localisation des objets détectés;
- La méthode basée sur l'estimation Σ-Δ est plus adaptative localement et temporellement, la localisation est plus précise mais elle est plus sensible aux modifications brutales du fond.

Actuellement, nous étudions la possibilité de combiner ces différentes méthodes afin de tirer parti au mieux des possibilités de chaque filtre.
Un autre exemple d'implémentation d'algorithmes se trouve en [15] où l'on montre comment le calcul de points d'intérêt peut être mené de façon optimale sur de telles architectures.

References

[1] V. Agnus, C. Ronse, and F. Heitz. Spatio-temporal segmentation using morphological tools. In *15th ICPR*, pages 885–888, Barcelona, Spain, September 2000.

[2] S.S. Beauchemin and J.L. Barron. The computation of optical flow. In *ACM Computing Surveys*, volume 27(3), pages 434–467, September 1995.

[3] T.M. Bernard and F. Paillet, Output methods for an associative operation of programmable artificial retinas, *IEEE/RJS Int. Conf. on Intelligent Robots and Systems*, 752–757, September 1997.

[4] P. Bouthemy and P. Lalande. Recovery of moving objects masks in an image sequence using local spatiotemporal contextual information. *Optical Engineering*, 32(6):1205–1212, June 1993.

[5] S-C.S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. In *IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Nice, France, 2003.

[6] E. Decenciere Ferrandiere, S. Marshall, and J. Serra. Application of the morphological geodesic reconstruction to image sequence analysis. *IEE Proceedings - Vision, Image and Signal Processing*, 144(6):339–344, December 1997.

[7] K.P. Karmann and A. Von Brandt, Time-varying image processing and moving object recognition, chapter Moving Object Recognition Using an Adaptative Background Memory, Elsevier, 1990.

[8] T. Komuro, I. Ishii, M. Ishikawa and A. Yoshida, A digital vision chip specialized for high-speed target tracking, *IEEE Trans. on Electron Devices*, 50(1):191–199, 2003.

[9] A. Manzanera and J. Richefeu. A robust and computationally efficient motion detection algorithm based on Σ-Δ background estimation. In *ICVGIP*, Kolkata, India, 2004.

[10] F. Paillet, D. Mercier and T.M. Bernard, Second Generation Programmable Artificial Retina, *Proc. IEEE ASIC/SOC Conf.*, 304–309, September, 1999.

[11] M. Pic, L. Berthouze, and T. Kurita. Active background estimation: Computing a pixel-wise learning rate from local confidence and global correlation values. *IEICE trans. Inf. & Syst.*, E87-D(1):1–7, January 2004.

[12] M. Piccardi. Background subtraction techniques: a review. In *Proc. IEEE SMC/ICSMC*, October 2004.

[13] F. Ranchin and F. Dibos. Moving objects segmentation using optical flow estimation. Technical report, UPD Ceremade, December 2003. http://www.ceremade.dauphine.fr/CMD/preprints03/0343.pdf.

[14] J. Richefeu and A. Manzanera. A new hybrid differential filter for motion detection, In *ICCVG*, Warsaw, Poland, 2004.

[15] J. Richefeu and A. Manzanera, Morphological dominant points detection and its cellular implementation, In *ISSPA*, Paris, France, 2003.

[16] P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE trans. on Image Processing*, 7(4):555–580, April 1998.

[17] J. Serra, Image analysis and mathematical morphology, New York, Academic Press, 1982.

[18] C. Stauffer and W.E.L. Grimson, Learning patterns of activity using real time tracking, *IEEE trans. on PAMI*, 22(8):747–757, August 2000.

[19] K. Toyoma, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *ICCV*, pages 255–261, Kerkyra, Greece, 1999.

[20] L. Vincent. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. *IEEE trans. on Image Analysis*, 2(2):176–201, April 1993.