Bertrand LASSERRE

Daniel DE LA FUENTE LEGASA

SYSTEME DE TELESURVEILLANCE VIDEO

Avril - Juin 98

Résumé des auteurs :		
Auteurs	**:**	Bertrand LASSERRE Daniel DE LA FUENTE LEGASA
Responsables	**:**	P. KADIONIK Y. BERTHOUMIEU
Nombre de pages **Nombre de pages « Annexe »**	**:** **:**	68 0

Remerciements

Nous tenons à remercier tout particulièrement nos responsables de projet, Messieurs Patrice KADIONIK et Yannick BERTHOUMIEU, pour l'aide précieuse qu'ils nous ont apportée tout au long de ces trois mois. Leur encadrement et leur expérience fut vraiment une expérience enrichissante.

Nous tenons également à adresser un grand merci :

- à M. MARCHEGAY, responsable de la filière T.I.C.

- à M. DULAU pour son aide sur la partie vidéo.

Sommaire

Liste des figures

1. PRÉSENTATION DU SUJET

1.1 Cahier des charges

« feuille présentant le projet ».

1.2 Analyse de l'existant

1.2.1 Introduction

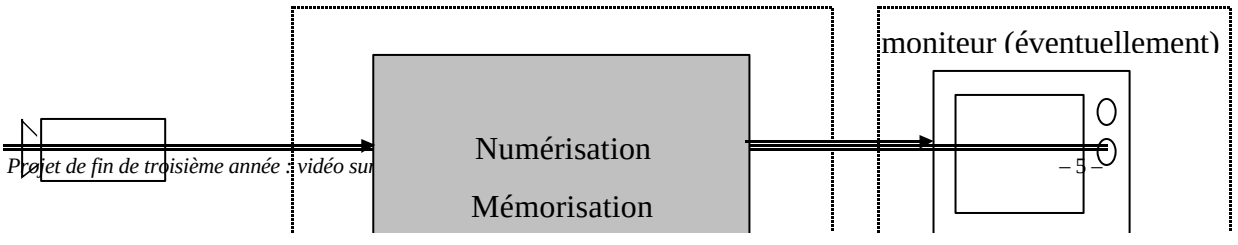
Ce projet fût présenté initialement au cours de l'année 1996 et en est donc à sa troisième année de développement. Cette année, nous nous sommes, en tant que binôme de l'option T.I.C., attachés à valider le travail effectué par nos prédécesseurs mais en approfondissant la gestion du signal vidéo, les algorithmes de traitement, les mécanismes de restitution en norme CCIR.

1.2.2 Travail effectué

Le binôme précédent avait réussi à réaliser la carte d'acquisition et restitution en échantillonnant l'ensemble du signal vidéo.

La camera fournit un signal vidéo monochrome analogique qui est numérisé et mémorisé. Ainsi, les données numérisées contiennent non seulement l'information de luminance mais aussi les tops de synchronisation du fait de la méthode d'échantillonnage.

Cette méthode permet de reconstruire sans de grandes difficultés le signal vidéo pour le visualiser sur un moniteur.



Caméra Vidéo

Résultat du traitement :

Déclenchement de l'alarme,

Sauvegarde et/ou transfert de l'image utile.

Figure 1 : Schéma bloc général.

Il ne restait donc plus qu'à piloter les phases d'acquisition et de restitution, ainsi qu'à programmer l'algorithme de détection. Cet à dire décrire le fonctionnement général de la carte.

1.2.3 Modifications apportées

Afin de simplifier les algorithmes de traitement, les données numérisées vont se limiter à l'information de luminance des images, d'où la réalisation des suivantes modifications :

- le CAN d'attaque a été changé de manière à échantillonner seulement le signal utile et à s'affranchir des tops de synchronisation ;
- reconfiguration des deux altéras de manière à tenir compte de cette modification. En effet, aussi bien pour l'acquisition comme pour la restitution, il faut piloter respectivement le CAN et le CNA pour pouvoir se limiter à l'échantillonnage de l'information de luminance ;
- modification de l'étage de sortie afin d'obtenir un signal conforme à la norme CCIR, c'est à dire comportant l'information de luminance ainsi que les tops de synchronisation.

2. PRINCIPE DE FONCTIONNEMENT

2.1 Synoptique

Le fonctionnement de cette carte va être orchestré par le micrcontrôleur 68HC11K1. La série K1 permet d'obtenir un plus grand nombre d'adresse et donc d'augmenter le plage mémoire adressable.

Les étapes décrite par la carte sont les suivantes :

1. l'acquisition des informations de luminance de l'image uniquement ;
2. le transfert des données entre deux mémoires et la restitution d'un signal vidéo aux normes CCIR ;
3. le calcul de valeurs nécessaire à la détection de mouvement.

C'est la dernière étape qui va faire déclencher l'alarme.

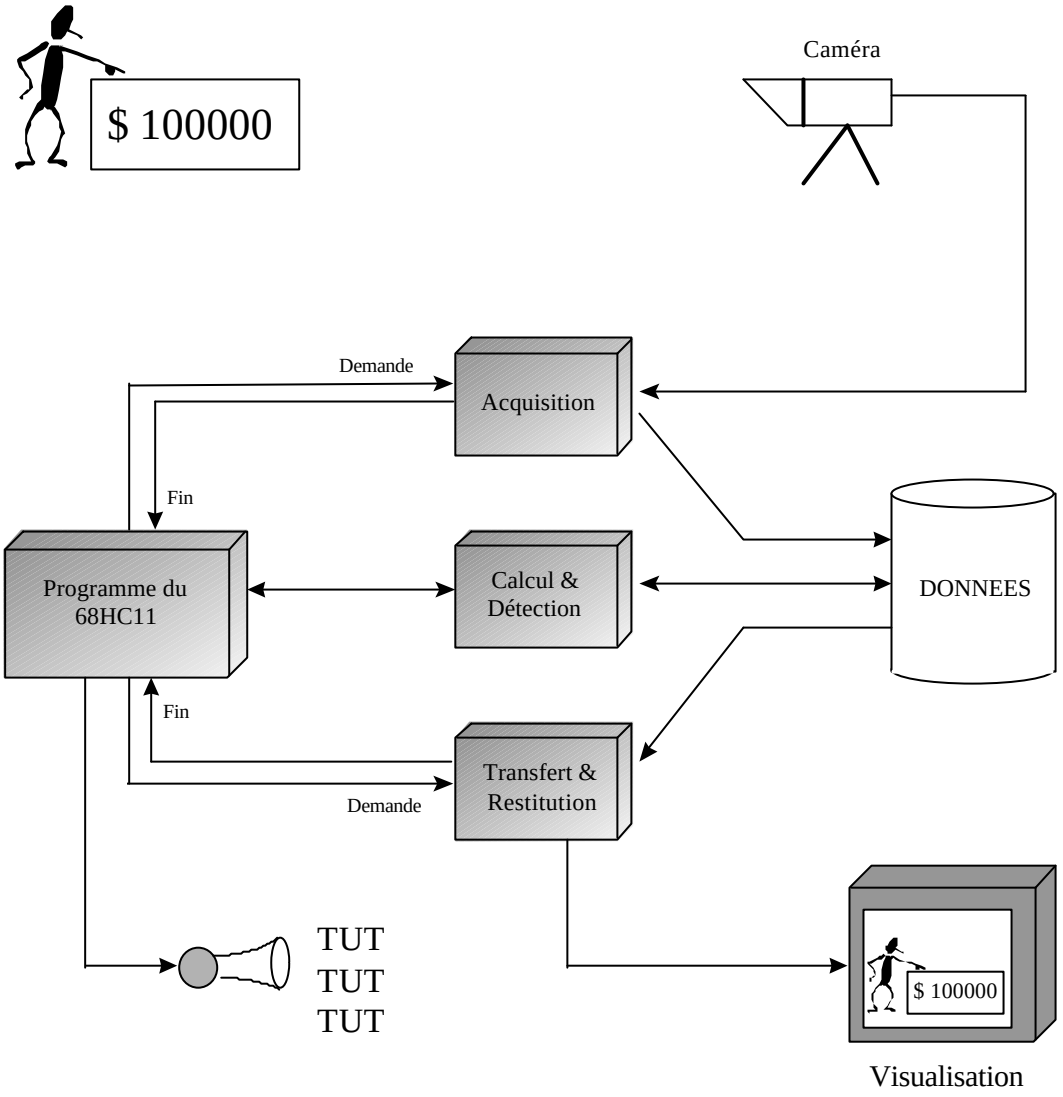


Figure 2 : Synoptique.

Remarques :

- *afin d'alléger la programmation du 68HC11, un séquenceur a été rajouté réalisant les procédés d'acquisition, de transfert des données et la restitution du signal vidéo ;*
- *une partie supplémentaire sera à prévoir concernant la compression d'une image et sa transmission par liaison série.*

2.2 Remarques sur le signal vidéo

Le signal vidéo délivré par la camera étudié ne comporte pas deux trames (trame paire et trame impaire) comme en télévision conventionnelle. En effet, après visualisation de deux trames consécutives, nous avons remarqué que leurs tops de synchronisation étaient positionné de façon identique.

Une trame correspond donc à une image donnée qui comprends donc un nombre de lignes moitié par rapport à une image entrelacée (625 en France). Cette trame se présente de la manière suivante :

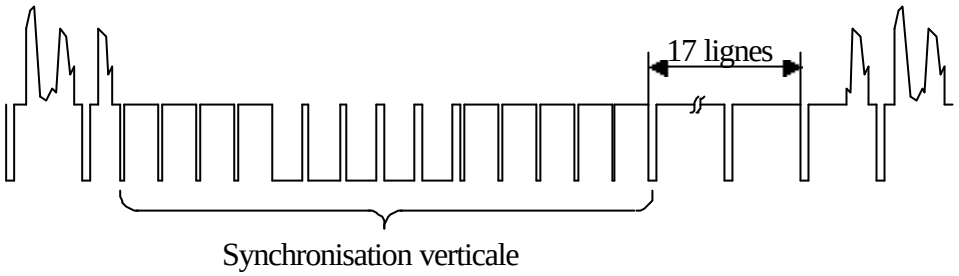


Figure 3 : Caractéristique du signal vidéo.

Après étude du signal, il est possible de déterminer certaines caractéristiques utiles par la suite :

- il se compose de 288 lignes de luminance dont deux demi ;
- la première demi ligne est calée sur la durée d'un ligne ;
- 17 lignes de suppression sont présentes après le signal vidéo.

3. ETUDE ET RÉALISATION

3.1 Critères de détection

L'algorithme de détection constitue le cœur de ce dispositif de télésurveillance. Il est basé sur l'analyse de la variation de luminance de l'image en provenance de la caméra.

Sa fiabilité devra être optimale, cela signifie que toute intrusion dans le champ de la caméra d'un intrus déclenchera l'alarme. Toutefois, il faudra, pour éviter un déclenchement intempestif de cette dernière, distinguer si la variation de luminance est causée par une présence humaine ou bien si elle résulte de phénomènes naturels (par ex : soleil intermittent dans un ciel nuageux).

3.1.1 Données vidéo disponibles

La caméra dont nous disposons présente :

- un angle de vision vertical de 50° ;
- un angle de vision horizontal d'à peu près 68° ;

Si nous avons un échantillonnage de l'image en résolution 100(H) x 288(V) pixels, alors pour la caméra, la surface perçue par le pixel dépendra de la distance entre la caméra et l'objet observé.

Distance	Pixel Horizontal	Pixel Vertical
1 m	1,35 cm	0,32 cm
2 m	2,70 cm	0,65 cm
5 m	6,75 cm	1,62 cm
10 m	13,5 cm	3,20 cm
20 m	27,0 cm	6,50 cm

Figure 4 : Représentation de la surface perçue par le pixel en fonction de la distance.

Ceci signifie que plus l'objet est lointain, plus celui-ci présente des caractéristiques de luminosité variables. En effet, cette dernière dépend, entre autres, des réflexions lumineuses sur les autres objets de la scène, essentiellement ceux présents entre l'objectif de la caméra et l'objet visé. L'information luminance sera donc à manier avec précaution à grande distance (supérieure à 5 m).

3.1.2 Principe de l'algorithme implanté

Prenons un exemple de système souhaité :

On veut détecter toute présence humaine de 2 mètres à 10 mètres dans un local clos (par ex : hangar).

Un petit homme, s'il s'y prend bien, peut arriver à diminuer son « écho visuel » - c'est à dire la surface apparente de son corps - à quelques milliers de cm^2.

Prenons une hypothèse très favorable pour lui de 2700 cm^2 = 180 cm x 15 cm : imaginez-le rampant sur le sol. Pour simplifier, on le modélise sous forme d'un rectangle posé horizontalement sur le sol.

Livrons nous à quelques petits calculs :

- A 2 m, il couvrira :
  180 / 2,70 ~ 66 pixels horizontaux ;
  15 / 0,65~ 23 pixels verticaux.

Son écho visuel représentera (66 x 23) / (100 x 288) = 5,27 % de l'image.

- A 10 m, il couvrira :
  180 / 13,5 ~ 13 pixels horizontaux ;
  15 / 3,20 ~ 4 pixels verticaux.

Son écho visuel représentera (13 x 4) / (100 x 288) = 0,18 % de l'image complète.

Pour que le système fonctionne, il faut que l'algorithme d'analyse de variation de luminance parvienne à percevoir la limite basse de l'écho, soit 0,18 %.

L'idée initiale est de diviser l'image en sections et d'analyser les variations sur chacun de ses blocs, car avec un écho faible, la variation globale de luminance ne sera pas significative (cf. tableaux page suivante).

L'idéal serait que l'individu à détecter couvre, quel que soit la distance, le plus possible de blocs pour optimiser la détection. Malheureusement, aucun système ne nous permet de prédire à quelle distance se trouve l'intrus. Il faudra donc faire un choix au niveau du nombre de blocs.

La taille des blocs en pixels dépendra de plusieurs paramètres :

- la rapidité de calcul des circuits électroniques ;
- la capacité de stockage des RAMs de mémorisation ;
- le degré de fiabilité de l'information luminance, i.e. qu'il faudra garder un nombre de pixels significatif par bloc, pour pouvoir opérer un traitement statistique, au moins plusieurs dizaines.

L'algorithme procédera en étapes pour analyser une image par rapport à la précédente :

1. segmentation de l'image en blocs et calcul de la valeur moyenne de luminance pour chacun ;
2. calcul de la valeur moyenne de la luminance de l'image entraînant un ajustement linéaire du seuil de détection S ;
3. calcul de la variation des moyennes des blocs entre 2 images et comparaison de la différence (D) avec le seuil S :

   D < S => R.A.S.

   D > S => ALARME

3.1.3 Choix de la taille des blocs

Notre choix s'est porté sur des blocs de 10 (H) x 9 (V) pixels.

On a ainsi (288 x 100) / 90 = 320 blocs.

Chaque bloc correspond à 0,31 % de l'image.

La variation de luminance due à l'intrus, si on suppose qu'elle est assez significative, affecterait donc un certain nombre de blocs de manière non négligeable (par ex : variation de luminance moyenne d'un bloc de 5 %) entraînant le déclenchement de l'alarme.

Le paramètre vitesse de l'intrus est à prendre en compte, il faut donc traiter le plus rapidement possible les images (200 ms semble un bon compromis)

La faille du système est la détection à faible distance. Ainsi, à 1 m, un bloc correspond à 13,5 (H) cm x 3,2 cm (V). A cette distance, un gros frelon pourrait faire varier significativement l'information luminance !

Cet algorithme est donc plus spécialement adapté à des systèmes de télésurveillance implantés dans des lieux clos.

3.1.4 Analyse taux de variation de luminance pour différentes séquences et conclusions

Remarques préliminaire :

- *ici, chaque bloc correspond à 1/28 de l'image, soit 7 blocs horizontalement et 4 blocs verticalement. Les calculs de variations moyennes et maximales ont été réalisés par l'intermédiaire du progiciel Matlab à partir d'acquisitions vidéo via la carte de Matrox Inspector.*
- *l'échantillonnage se fait à une image (+ traitement associé) toutes les 10 ms pour les séquences 1, 2 et 3.*
- *il se fait toutes les 30 s pour la séquence fixe 4.*
- *le source du programme Matlab qui nous a permis l'obtention des résultats qui suivent ainsi que la représentation des acquisitions sont fournis en annexe.*

Séquence 1 : Voleur peu discret

	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Image 7
Variation moyenne	----------	- 3,10 %	- 7,97 %	- 2,40 %	- 5,20 %	- 13,1 %	+ 1,12 %
Variation	----------	- 32,9 %	- 56,1 %	+ 78,6 %	- 88,3 %	- 117 %	+ 86,2 %

Figure 5 :Taux de variation par bloc pour la séquence 1.

Ici, pas de problème, les variations maximales de luminance sont très importantes, on n'a même pas à se préoccuper de niveau moyen général de luminance.

Séquence 2 : Déplacement Noir sur Fond Sombre

	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Image 7	Image 8
Variation	----------	- 0,24 %	- 2,44 %	- 0 ,16	- 0,20 %	+ 1,47	+ 0,54	+ 1,90
Variation	----------	+ 0,37	- 17,1 %	+ 17,5	+ 12,4	+ 10,9	+ 5,64	- 3,47 %

Figure 6 : Taux de variation par bloc pour Séquence 2.

Le mince câble noir que l'on déplace transversalement devant l'objectif de la caméra simule un voleur à grande distance. Les variations de luminance sont encore fort significatives et la détection sera aisée.

Séquence 3 : Déplacement Blanc sur Fond Clair

	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Image 7	Image 8
Variation	----------	- 0,02 %	+ 0,01	+ 0,02	+ 0,01	+ 0,05	+ 0,06	+ 0,07
Variation	----------	- 0,14 %	+ 0,12	+ 0,15	+ 0,16	+ 0,15	- 1,34 %	- 1,12 %

Figure 7 : Taux de variation par bloc pour séquence 3.

Le bout de tube plastique blanc simulant toujours un voleur à grande distance n'engendre pas, lui, une variation maximale importante, d'où l'idée de l'ajustement des seuils avec le niveau de luminance moyen global :

- Plus on sera dans les blancs, plus le seuil sera bas ;
- Plus on sera dans les noirs, plus il sera haut.

Séquence 4 : Images Fixes

	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Image 7
Variation moyenne	----------	- 0,14 %	- 0,09 %	- 0,14 %	+ 0,07 %	+ 0,10 %	+ 0,13 %
Variation maximale	----------	+ 0,00 %	+ 0,00 %	- 0,02 %	+ 0,01 %	+ 0,01 %	- 0,01 %

Figure 8 : Taux de variation par bloc pour séquence 4.

Le soleil jouant avec les nuages, ne fait pas varier, ou si peu, les niveaux de luminance même en 3 minutes. La variation de la luminosité du jour ne sera donc pas à prendre en compte dans notre algorithme.

Attention toutefois au soleil direct, qui, lui, sature l'objectif (voile blanc). La caméra sera donc disposée pour éviter cette éventualité.

3.1.5 Bilan

L'analyse de ces résultats corroborent notre idée que l'exploitation de la variation maximale de luminosité parmi N blocs est adéquate pour affirmer ou non la présence d'un intrus.

Cependant, si pour un voleur bien distinct, et occupant au moins largement un bloc, la variation maximale est très forte (qq. dizaines de % pour la séquence 1). Cette variation maximale est beaucoup plus faible (qq. %) pour un intrus à grande distance et sensiblement de même couleur que le fond (C'est ce que nous avons tenté de simuler avec les séquences 2 et 3). On doit , en outre, tenir compte du niveau de luminance moyen global.

Le compromis choisi est de diviser l'image en 320 blocs (cf. C) et d'ajuster linéairement le seuil en fonction de la luminance moyenne de l'image selon le graphe qui suit :

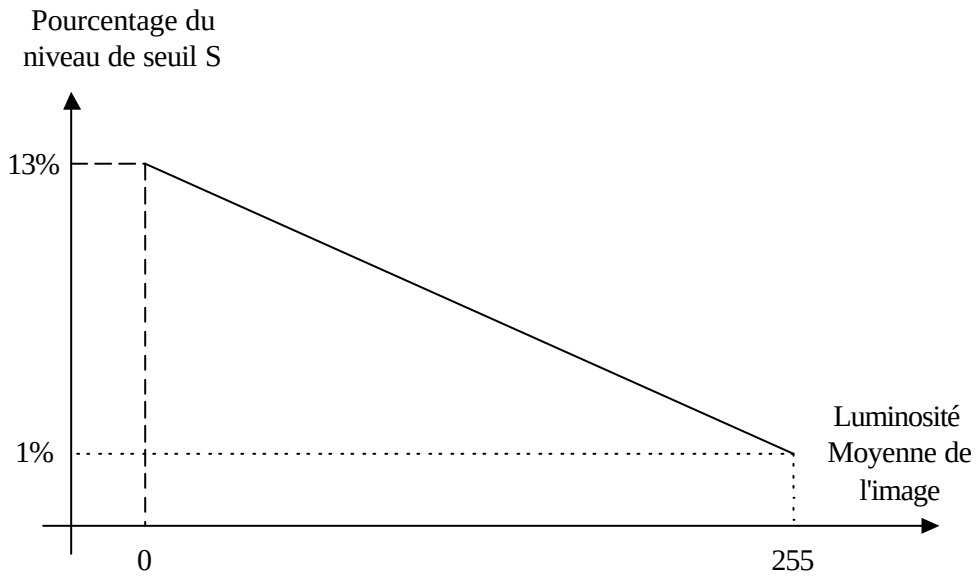


Figure 9 : Niveau du seuil.

3.2 Programmation du micro-controleur

3.2.1 Programme principal

Le but de ce logiciel est de piloter les diverses étapes de la carte et d'effectuer les opérations nécessaires à la détection d'un intrus.

On commence le programme par l'acquisition d'une image et on calcule en suivant les moyennes des différents blocs qui composent l'image. Après transfert des données dans la RAM 2, on procède à la restitution des informations de cette RAM.

Ensuite, on relance une nouvelle acquisition et, après calcul des valeurs nécessaires pour cette dernière image et restitution de l'image enregistrée, le test de détection de mouvement est réalisé.

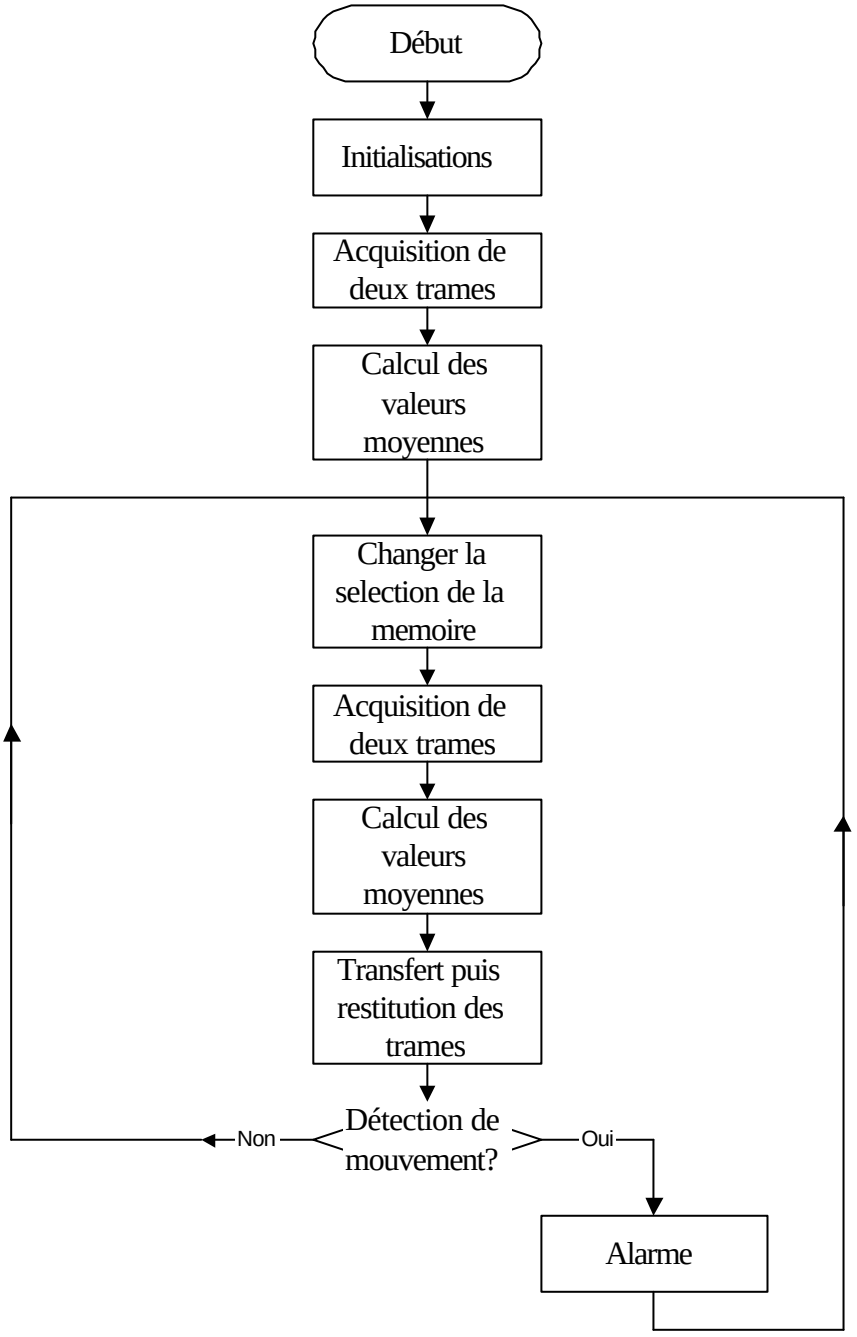


Figure 10 : Organigramme du programme principal.

Remarque : les procédures d'acquisition, de transfert et de restitution se limitent au lancement du séquenceur qui effectue les démarches nécessaires.

3.2.2 Procédure de calcul des moyennes

Cette procédure a pour objet le calcul des valeurs moyennes de chacun des blocs qui composent l'image (cf. Critères de détection page 10) servant à la détection de l'intrus.

Pour cela, elle va récupérer une à une toutes les informations de luminance pour chacun des blocs, calculer leur moyenne, puis sauvegarder cette valeur dans la RAM respective.

En même temps, elle effectue les mêmes opérations pour la valeur moyenne de l'image complète. Elle servira à déterminer la valeur du seuil de détection.

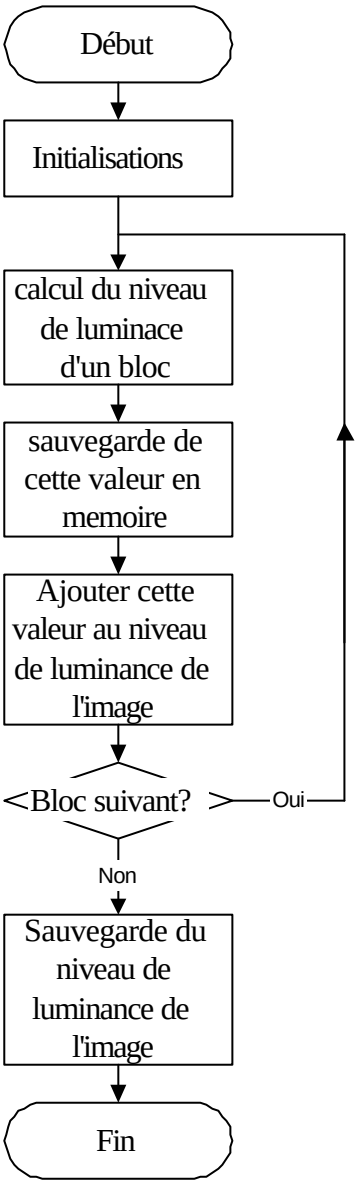


Figure 11 : Organigramme de la procédure de calcul des moyennes.

3.2.3 Procédure de détection du mouvement

Elle va utiliser les moyennes préalablement calculées et sauvegardées pour estimer s'il y a eu mouvement.

D'abords, Elle procède à la détermination de la valeur du seuil en utilisant pour cela la valeur moyenne de la dernière image stockée. Ensuite, elle effectue la différence entre chacune des moyennes bloc des deux images et le compare au seuil. Si elle est supérieure, l'alarme est enclenchée.

3.3 Programmation des alteras

3.3.1 Le compteur

Un des alteras est utilisé comme compteur servant à incrémenter les adresses lors de l'acquisition et la restitution de l'image ainsi que lors du transfert des données d'une RAM à une autre. Il comporte aussi un mode transparent dans lequel il recopie les adresses provenant du 68HC11.

Cette partie n'a pas été modifié, si ce n'est le nombre total d'échantillons, car le compteur est piloté de l'extérieur et dans notre cas par le séquenceur.

Remarque : l'utilisation d'un altera n'est pas obligatoire, il est possible de fabriquer le compteur avec des composants discrets. Le seul problème est que, lorsqu'il y a une modification, il est nécessaire de refaire la carte.

3.3.2 Le séquenceur

3.3.2.1 Introduction

Le séquenceur est utilisé pour décrire les différentes étapes afin de réaliser l'acquisition et la restitution d'une image, et aussi le transfert des données d'une RAM à l'autre.

Il comporte un état, dit d'initialisation, qui sert aussi comme mode transparent, et réalise alors la correspondance des fils d'adresse du port G du 68HC11 (pour adresser les fenêtres) et ceux des RAMs.

Il a été modifié car précédemment les données échantillonnées comportaient non seulement l'information de luminance mais aussi celle sur les synchronisations verticales et horizontales.

Les procédés d'acquisition et de restitution ont donc été complètement modifiés permettant ainsi de limiter la sauvegarde à l'information de luminance.

3.3.2.2 Acquisition

Afin de comprendre la méthode développée pour échantillonner le signal vidéo en provenance de la caméra, les suivantes explications vont être accompagnées par des illustrations.

Le fonctionnement d'un séquenceur se limite à la description des étapes préalablement introduite. Pour optimiser sa programmation et donc le nombre de circuits utilisés (portes logiques, bascules), donc le signal vidéo doit être décomposé de façon à obtenir une répétition dans les séquences.

Une trame est composée de deux parties :

- la synchronisation verticale ;
- les lignes qui comportent chacune d'elles la même structure.

D'où l'idée de se synchroniser en début de chaque ligne pour ensuite échantillonner le signal utile :

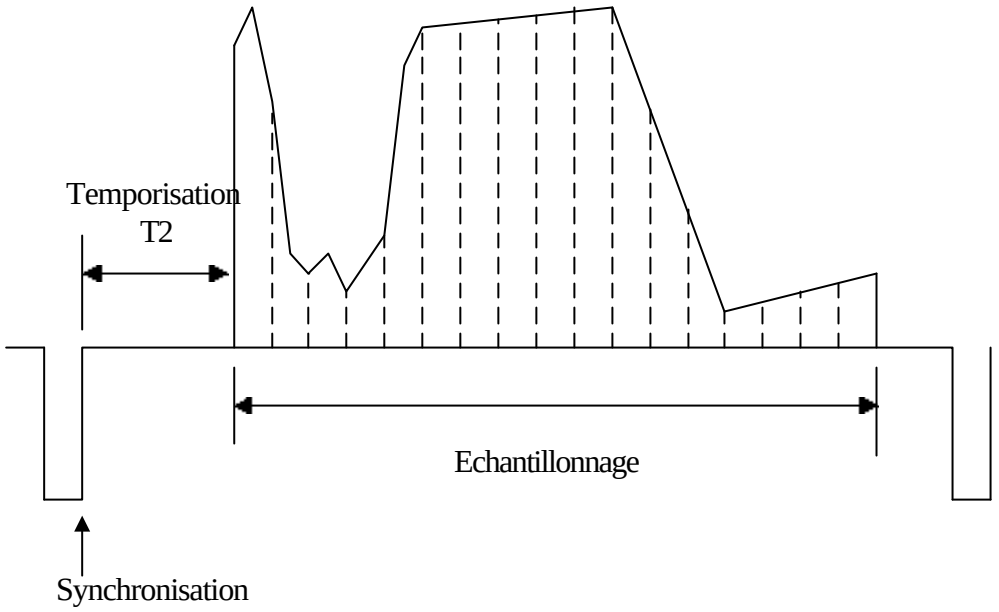


Figure 12 : Décomposition d'une ligne.

D'après l'information de la ligne, l'ajout d'une temporisation est évidente pour stocker seulement l'information de luminance.

En visualisant le signal vidéo, l'utilisation d'une seconde temporisation, visant à éliminer les lignes de suppression, est aussi nécessaire.

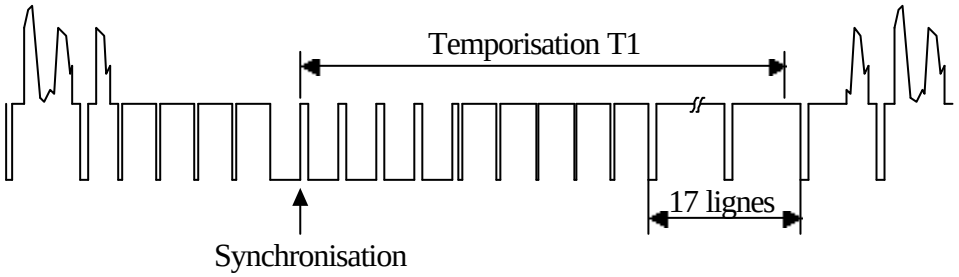


Figure 13 : Description d'une trame.

Ainsi, la séquence d'acquisition se décompose en 10 étapes dont une est celle d'initialisation :

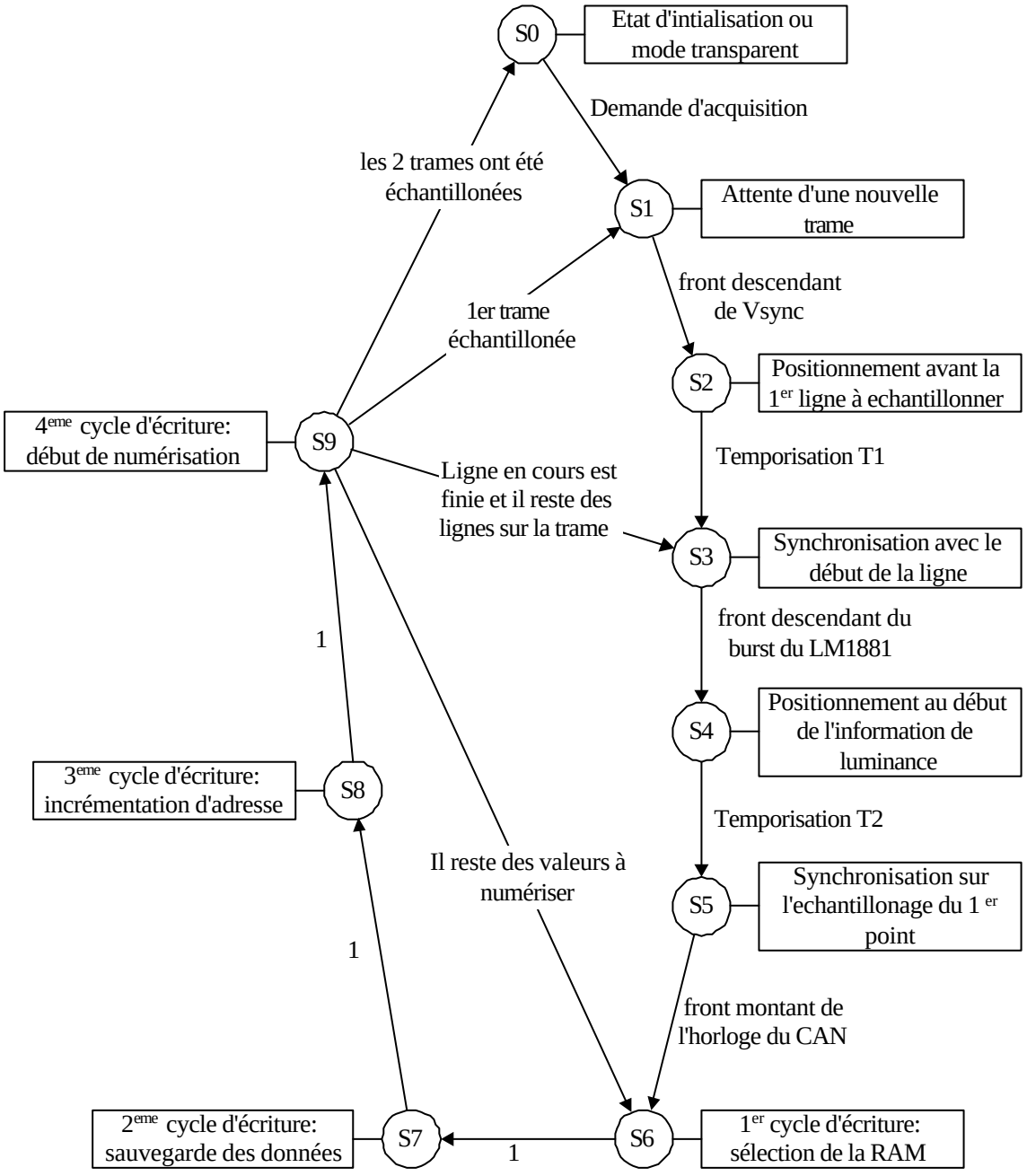


Figure 14 : Graphe des états de l'Acquisition.

3.3.2.3 Transfert

Cette séquence est la plus simple des trois. Les modifications apportés n'ont pas été très importante car elle lit l'information contenue dans une RAM et l'écrit sur une autre.

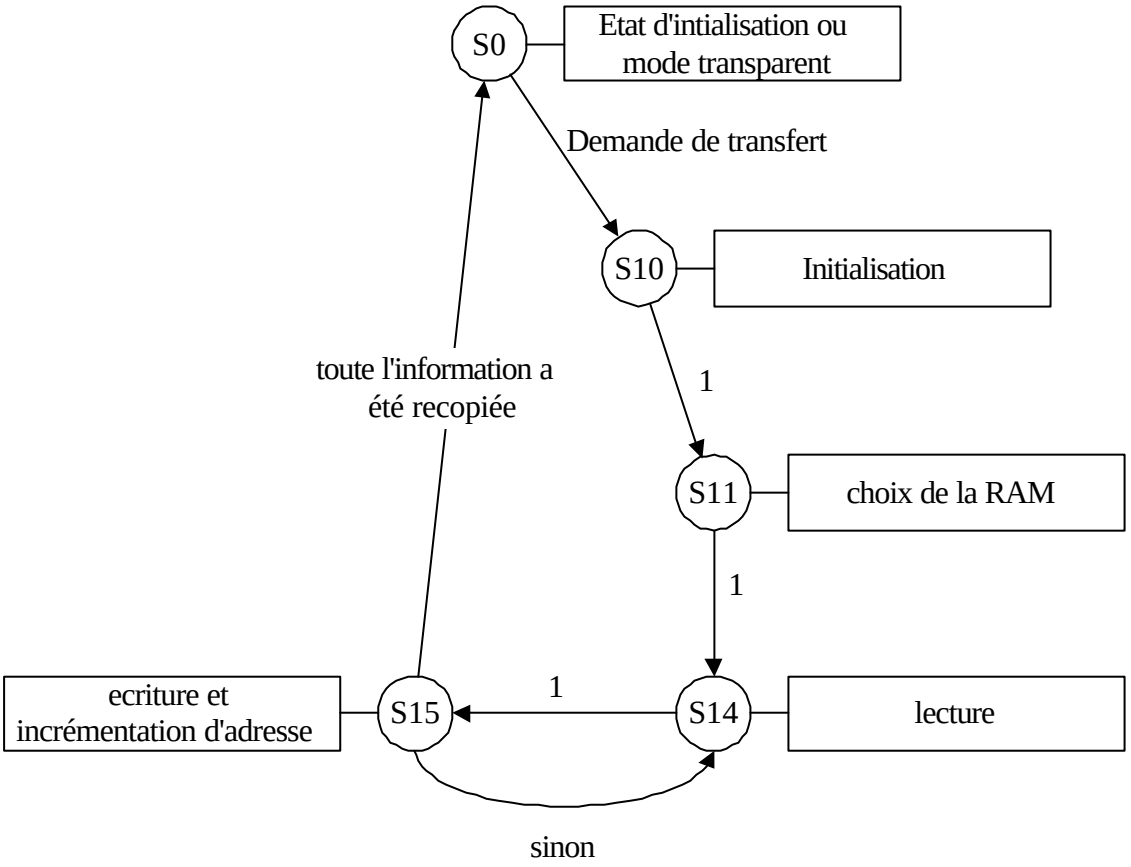


Figure 15 : Graphe des états du Transfert.

3.3.2.4 Restitution

Pour cette partie, la logique de séquencement est peu différente de celle utilisée pour l'acquisition (cf. page 24).

La modification de la sauvegarde des données, et plus particulièrement le fait que les tops de synchronisations ne soient pas stockés, oblige à changer, en plus du séquenceur, l'étage de sortie.

En effet, le signal de sortie doit être aux normes CCIR et donc comporter les tops de synchronisation. Pour cela, la solution adoptée se base sur la sélection soit de l'information luminance (en sortie du CNA) soit du signal de synchronisation délivrée par le LM1881 :

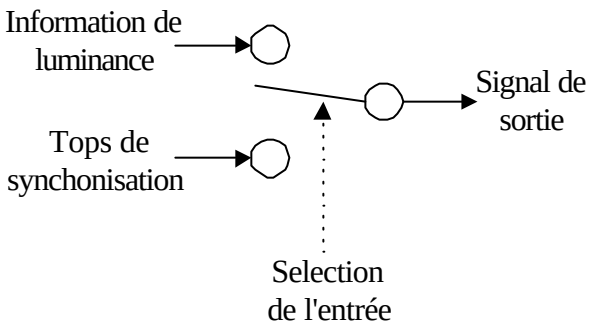


Figure 16 : Schéma de sortie.

Par contre, il est nécessaire de créer le signal effectuant la sélection de l'un ou l'autre signal. Ce dernier va être élaboré par le séquenceur étant donné que les deux phases de sélection sont déterminées lors de la lecture des données.

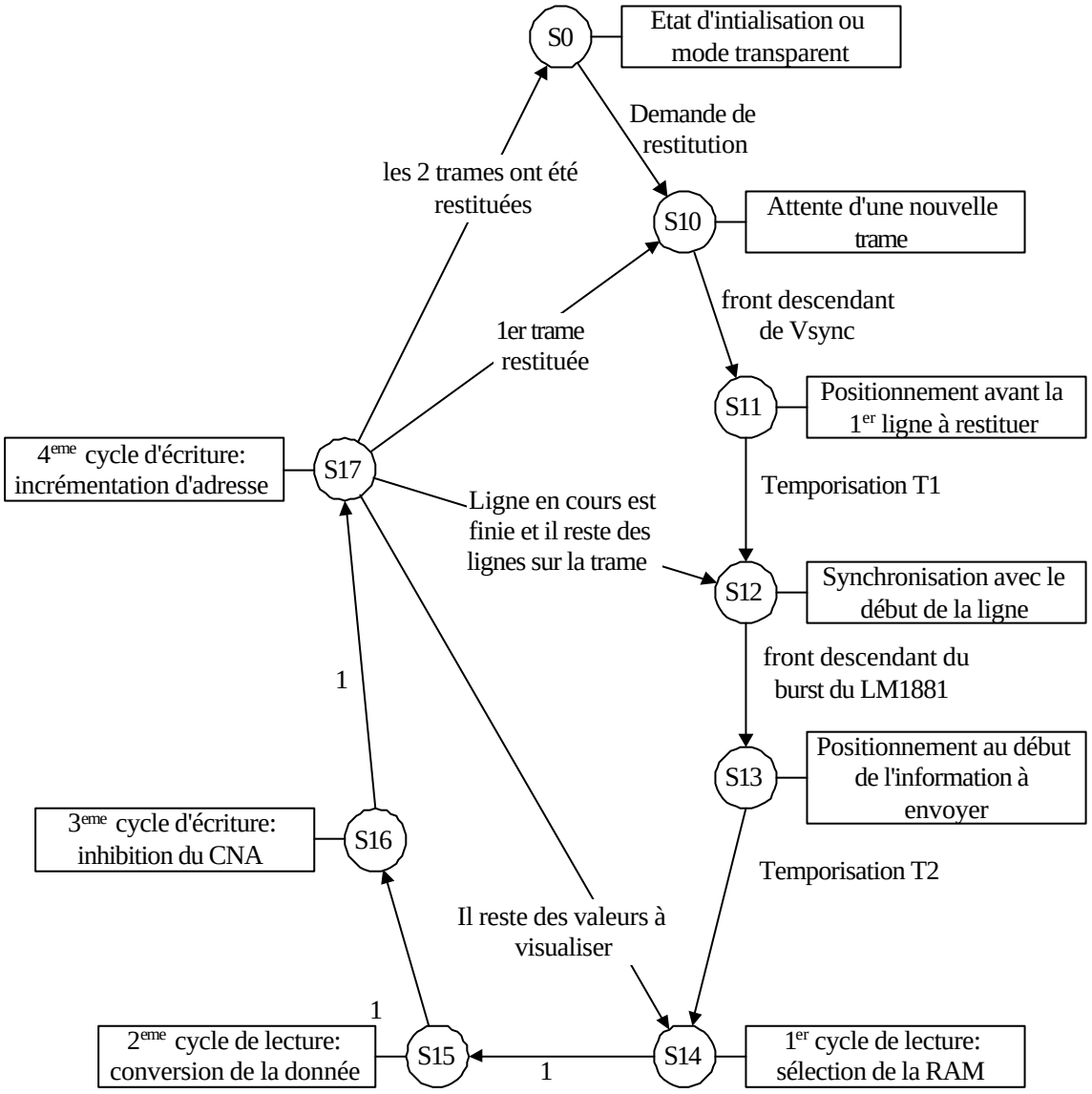


Figure 17 : Graphe des états de Restitution.

4. CONCLUSION

L'étude de ce système de vidéo-surveillance nous a permis d'appréhender toutes les difficultés liées à la gestion d'images. Qu'il s'agisse de problèmes d'acquisition (CAN), de stockage, de traitement ou de restitution (CNA), tous ont nécessité la mise en place de solutions électroniques ou informatiques adéquates.

Cette année, nous nous sommes efforcés de donner plus de "liant" au projet. Les procédés d'acquisition et de restitution ont été revus d'où la modification de la carte, ainsi que la programmation générale du séquenceur. Par ailleurs, un algorithme de détection, basée sur l'analyse des variations de luminance entre images successives, a été implémenté sur le microcontrôleur.

Le système, s'il ne peut prétendre être d'une fiabilité absolue, s'insérera parfaitement en appui d'un système de détection plus classique (ondes).

En outre, Le fait de disposer d'une image vidéo de l'intrus, même partielle, facilitera une éventuelle procédure d'enquête.

Ce sujet nous a permis de comprendre que lorsqu'on veut réaliser un projet composé d'au moins deux parties (électronique et informatique), il est préférable d'étudier chacune des parties (du point de vue général). En effet pour obtenir de meilleures performances, il est judicieux de simplifier le plus possible les échanges entre les différentes parties.

Dans notre cas, les données stockées en mémoires comportaient des informations de synchronisation non utilisable par la détection elle-même. Donc, il aurait fallu déterminer si l'information était utile ou pas d'où une perte de rapidité et une mauvaise utilisation de la capacité mémoire de la carte.

5. ANNEXES

5.1 Fichier source du programme en C

```
/*****************************************************************/
/*                  PROGRAMME DE VIDEO SURVEILLANCE              */
/*                    version 4.1 du 16 juin 1998                */
/*                    CREATEURS: LASSERRE Bertrand               */
/*                               DE LA FUENTE LEGASA Daniel      */
/*****************************************************************/

#include "mc68hc11.h"
#include "video.h"

main()
{
     enum boolean fin=false;    //indique la fin de la détection
     unsigned char   memoire=0; //choix de la RAM 0
                                //pour le premier stockage
     unsigned char   alarme=0;  //indique s'il y a eu détection
                                //d'intru


     acquisition(memoire,alarme);   //lecture la premiere image
     calcul_donnees(memoire);       //calcul des valeurs de la
                                    //premiere image
     while (!fin)
     {
       memoire=(memoire)?0:1;  //changer la s,lection de la RAM
       acquisition(memoire, alarme);//lecture d'une nouvelle image
       calcul_donnees(memoire);//calcul des valeurs de cette images
```

```
        transfert(memoire,alarme); //chargement de la derniere image
        comp_trans(); //envoie par liaison serie de l'image
                      //declenchement de l'alarme
        visualisation(memoire,alarme);
        if (detection(memoire))
        {
            alarme=1;
            *DDRA=0x01;     //PA0 en sortie les autres en entrees
            *PORTA=0x01;
        }
     }
}

/*********************************************************************/
/* PA0=tor   PA1=start_acq   PA2=visu   PA3=transf_ext   PA4=fin  */
/*********************************************************************/

/*********************************************************************/
/*                    Procedure acquisition                          */
/*        donne l'ordre a l'altera de lire une nouvelle image        */
/*********************************************************************/
void acquisition(unsigned char memoire,unsigned char alarme)
{
     unsigned char i;

     //selection de la RAM (0 ou 1) correspondante
     *DDRG=0xFF;  //Port G des adresses XA et du R/W en sortie
     *PORTG=(memoire)?0x10:0x00;

     *DDRA=0x03;  //PA1 et PA0 en sortie les autres en entrees
     if (alarme)  //mise a 1 de START_ACQ
       *PORTA=0x03;
     else
       *PORTA=0x02;

     for (i=1;i<=6;i++); //tempo pour générer le créneau
```

```
     *DDRA=0xEF;  //PA4(fin_num) en entree les autres en sorties
     if (alarme)
       *PORTA=0x01;
     else
       *PORTA=0x00;

     while ((*PORTA)&(0x10))
       ;  //Attendre la fin de l'acquisition
}

/*****************************************************************/
/*                      Procedure transfert                      */
/*donne l'ordre a l'altera de transferer la derniere image obtenue*/
/*****************************************************************/
void transfert(unsigned char memoire,unsigned char alarme)
{
     unsigned char i;

     //selection de la RAM (0 ou 1) correspondante
     *DDRG=0xFF;  //Port G des adresses XA et du R/W en sortie
     *PORTG=(memoire)?0x10:0x00;

     *DDRA=0x09;  //PA3 et PA0 en sortie les autres en entrees
     if (alarme)  //mise a 1 de TRANSF_EXT
       *PORTA=0x09;
     else
       *PORTA=0x08;

     for (i=1;i<=6;i++); //tempo pour generer le creneau

       *DDRA=0xEF;   //PA4 en entree les autres en sorties
     if (alarme)
       *PORTA=0x01;
     else
       *PORTA=0x00;

     while ((*PORTA)&(0x10))
```

```
       ;   //attendre la fin du transfert
}

/*****************************************************************/
/*                    Procedure visualisation                    */
/*      donne l'ordre a l'altera de visualiser l'image stockee    */
/*****************************************************************/
void visualisation(unsigned char memoire,unsigned char alarme)
{
     unsigned char i;

     //selection de la RAM (0 ou 1) correspondante
     *DDRG=0xFF;  //Port G des adresses XA et du R/W en sortie
     *PORTG=(memoire)?0x10:0x00;

     *DDRA=0x05;  //PA2(visu) et PA0 en sortie les autres en entrees
     if (alarme)  //mise a 1 de VISU
       *PORTA=0x05;
     else
       *PORTA=0x04;

     for (i=1;i<=6;i++); //tempo pour generer le creneau

     *DDRA=0xEF;     //PA4 en entree les autres en sorties
     if (alarme)
       *PORTA=0x01;
     else
       *PORTA=0x00;

     while ((*PORTA)&(0x10))
       ;   //attendre la fin de la restitution
}

/*****************************************************************/
/*                    Procedure calcul_donnees                   */
/*    calcule les valeurs pour la detection et les sauvegardes   */
/*****************************************************************/
```

```
void calcul_donnees(unsigned char memoire)
{
     unsigned short   colonne,ligne,m;
     unsigned short   Moy;
     unsigned char       dec_c;
     unsigned short   mat_c_max;
     unsigned char       dec_l;
     unsigned short   mat_l_max;
     unsigned short   pt_ram;
     unsigned char       test;
     unsigned long       Moy_image=0;

     for (m=0;m<(unsigned)DIVTOT;m++)
     {
       //initialisation de la m,moire en cours
       Moy=0;

       //calcul des diff,rents d,calages
       dec_l=m/DIVCOL; // compris entre 0 et DIVLIG-1
       dec_c=m-dec_l*DIVCOL;   // compris entre 0 et DIVCOL-1

       //calcul des limites de la matrice
       mat_c_max=(dec_c==DIVCOL-1)?NBEL-dec_c*COLSEC:COLSEC;
       mat_l_max=(dec_l==DIVLIG-1)?NBL-dec_l*LIGSEC:LIGSEC;

       //la matrice de données a en réalité 2 sous ensemble
       //"identiques"
       for (ligne=0; ligne<mat_l_max;ligne++)
       {
           for (colonne=0; colonne<mat_c_max;colonne++)
           {
               //ajout du premier sous-ensemble
               Moy+=valeur_pt(colonne+ligne*COLSEC,memoire);
               //ajout du premier sous-ensemble
               Moy+=valeur_pt(colonne+ligne*COLSEC+NBL,memoire);
           }
       }
```

```
        //calcul de la valeur moyenne de l'image
        Moy_image+=Moy;

        //sauvegarde de la valeur moyenne calculée
        sauvlec_moy(&Moy, memoire, m, !LECTURE);
     }
     //sauvegarde de la valeur moyenne de l'image
     //valeur comprise entre 0 et 255 * NBPT
     Moy=(unsigned short) Moy_image;
     sauvlec_moy(&Moy, memoire, DIVTOT, !LECTURE);
     Moy=(unsigned short) (Moy_image>>(8*sizeof(short)));
     sauvlec_moy(&Moy, memoire, DIVTOT+1, !LECTURE);
}

/*****************************************************************/
/*                      Procedure comp_trans                     */
/*   compresion et transmission de l'image par la liaison serie   */
/*****************************************************************/
void comp_trans(void)
{
}

/*****************************************************************/
/*                     Procedure sauvlec_moy                     */
/*   sauvegarde ou lecture des moyennes préalablement calculées   */
/*****************************************************************/
void sauvlec_moy(unsigned short *pt_Moy, unsigned char memoire,
unsigned short num, unsigned char choix)
{
     unsigned char       valeur;
     unsigned char       masque;
     unsigned short   masq;

     if (choix==LECTURE)
     { //initialisation des m,moires
        *DDRC=0x00;   //Port C - bus de données - configuré en entrée
```

```
    masque=0x8F;  //lecture
  }
  else
  {
    *DDRC=0xFF;   //Port C - bus de données - configuré en sortie
    masque=0x0F;  //écriture
  }

  *DDRB=0xFF;  //Port B config en sortie
  *DDRF=0xFF;  //Port F config en sortie
  *DDRH=0xFF;  //Port H de chip select config en sortie
  *DDRG=0xFF;  //Port G des adresses XA et du R/W en sortie

  //sélection de la RAM (0 ou 1) correspondante
  *PORTH=0x20; //CSGP1=1 et CSGP2=0
  *PORTG=(memoire)?masque+0x10:masque;

  //sélection de l'adresse
  masq=0xFF-2*num; //2* car chaque mémoire est sauvegardée sur 2 o
  *PORTB=(unsigned char)(masq>>8);
  masque=(unsigned char) masq;

  //poids fort
  *PORTF=masque-1;
  if (choix==LECTURE)
  {
    valeur=*PORTC;
    (*pt_Moy)=valeur;
    (*pt_Moy)<<=8;
  }
  else
  {
    valeur=(unsigned char)((*pt_Moy)>>8);
    *PORTC=valeur;
  }

  //poids faible
```

```
     *PORTF=masque;
     if (choix==LECTURE)
     {
       valeur=*PORTC;
       (*pt_Moy)+=valeur;
     }
     else
     {
       valeur=(unsigned char)(*pt_Moy);
       *PORTC=valeur;
     }

     *PORTG=0;
     *PORTH=0;
}

/*****************************************************************/
/*                      Procedure detection                      */
/*  effectue les comparaisons des valeurs calculees des 2 images  */
/*****************************************************************/
char detection(unsigned char memoire)
{
     unsigned char       result=0;
     unsigned short   Moy;
     long             Dif;
     unsigned short   num;
     unsigned long       seuil;
     unsigned long       Moy_im;

     //lecture de la valeur moyenne de l'image
     sauvlec_moy(&Moy,memoire,DIVTOT+1,LECTURE);
     Moy_im=(unsigned)Moy;
     Moy_im<<=(8*sizeof(short));
     sauvlec_moy(&Moy,memoire,DIVTOT,LECTURE);
     Moy_im=+(unsigned)Moy;

     //determination du seuil en fonction de la luminosite de la
```

```
     //piece
     seuil=(PTB-PTN)*Moy_im;
     seuil/=LIMITE;
     seuil+=PTN;  //valeur entiere comprise entre PTN et PTB
     seuil*=Moy_im;
     seuil/=100;

     //mise sous la meme echelle que les moyennes des sections
     seuil/=(2*DIVCOL*DIVLIG);

     //calcul de la diff,rence des moyennes
     for (num=0;num<(unsigned)DIVTOT;num++)
     {
       //lecture des valeurs moyennes stock,es dans la dernière
       //mémoire
       sauvlec_moy(&Moy,memoire,num,LECTURE);
       Dif=Moy;

       sauvlec_moy(&Moy,!memoire,num,LECTURE);
       Dif-=Moy;

       Dif=(Dif<0)?-Dif:Dif;//valeur absolue

       if ((unsigned)Dif>(unsigned)seuil)
       {//seuil dépassé
           result=1;
           break;
       }
     }
     return (result);
}

/*****************************************************************/
/*                    Procedure valeur_pt                        */
/*   retourne la valeur du point indiqu, de la m,moire choisie    */
/*****************************************************************/
unsigned char valeur_pt(unsigned short pt, unsigned char memoire)
```

```
{
     unsigned char valeur;
     unsigned char masque;

     *DDRC=0x00;  //Port C - bus de données - configuré en entrées.

     *DDRB=0xFF;  //Port B config en sorties
     *DDRF=0xFF;  //Port F config en sorties
     *DDRH=0xFF;  //Port H de chip select config en sorties
     *DDRG=0xFF;  //Port G des adresses XA et du R/W en sortie

     //selection de la RAM (0 ou 1) correspondante
     *PORTH=0x20; //CSGP1=1 et CSGP2=0
     masque=(0x80)|   //en lecture
     ((0x07)&((unsigned char)(pt>>9)));//init de XA13 a XA15
     if (memoire) //choix de la RAM
       masque+=0x10;
     *PORTG=masque;  //selection de la RAM en lecture


     //selection de l'adresse
     *PORTB=(unsigned char)(pt>>8);
     *PORTF=(unsigned char)pt;

     valeur=*PORTC;

     //modification du au probleme amené par le CAN
     valeur=(valeur & 0x80)?(valeur & 0x7f):(valeur | 0x80);
     //sauvegarde de la donn,e lu pour la visu
     masque=masque &0x7F;//masque a modifier=>ecriture
     *PORTG=masque;
     //attente!?
     *DDRC=0xFF;  //Port C - bus de donn,es - configur, en sortie
     *PORTC=valeur;

     *PORTG=0;
     *PORTH=0;
```

```
     return (valeur);
}
```

5.2 Fichier source du séquenceur

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          PROJET : SYSTEME DE TELESURVEILLANCE VIDEO               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Nom du fichier  : valmc_sq.tdf                                   %
% Version et date : 5.0 16/06/98                                   %
% Auteurs         : LASSERRE Bertrand                              %
%                  DE LA FUENTE LEGASA Daniel                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description: sequenceur d'acquisition.                           %
%              Etat So mode ou l'ALTERA est transparent            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
DESIGN IS 'valmc_sq';

%clk              : INPUT;   horloge %
%start_acq        : INPUT;   signal de demarrage d'acquisition %
%choix_ram        : INPUT;   choix de la memoire pour ecriture %
%burst            : INPUT;   signal de synchronisation%
%sync_comp        : INPUT;   signal de synchronisation composite%
%visu             : INPUT;   signal de lecture de l'image numérisée %
%vsync            : INPUT;   synchro verticale du signal video %
%val_compt[17..0] : INPUT;       bus d'adresse en entree %
%fin_compt        : INPUT;   signal de fin de comptage%
% R/W             : INPUT;   signal R/W du microcontroleur %
% C1              : INPUT;   signal CSGP1 du micro %
% C2              : INPUT;   signal CSGP2 du micro %
%transf_ext       : INPUT;   signal pour les transferts des données %
%clk_can          : OUTPUT;  horloge pour le CAN %
%/we_ram          : OUTPUT;  write enable memoire %
%/we_ram2         : OUTPUT;  write enable memoire 2 %
%cs0              : OUTPUT;  chip select vers CE2 de RAM0 %
%cs1              : OUTPUT;  chip select vers CE2 de RAM1 %
```

```
%cs2              : OUTPUT;  chip select vers CE2 de RAM2 %
%/oe_can          : OUTPUT;  output enable pour CAN %
%adresse_out [16..0]     : OUTPUT;  bus d'adresse en sortie %
%/oe_ram          : OUTPUT;  output enable pour mémoires 0 et 1 %
%/oe2             : OUTPUT;  output enable pour mémoire 2 %
%fin              : OUTPUT;  signal de fin du procédé en cours%
%selection        : OUTPUT;  signal de sélection du signal de
                             luminance ou celui de synchro pour le
                             mux%
%clk_cna          : OUTPUT;  horloge pour le CNA %
%reset            : OUTPUT;  RAZ du compteur d'adresses %
%increm           : OUTPUT;  incrémentation du compteur d'adresses %
%copy             : OUTPUT;  signal pour avoir count_adr = adr_in %
%/burst           : OUTPUT;  signal de synchronisation%
%/sync_comp       : OUTPUT;  signal de synchronisation composite%

CONSTANT   TEMPO_PREM_LIG = 2698; % tps:2700% % data -2%
CONSTANT   TEMPO_DEB_INFO = 12; %tps:12% % data -2%
CONSTANT   NBR_ECH_LIG = 100;  %100% % data %
CONSTANT   NBR_LIG_TRA = 288;  %288% % data -1%

SUBDESIGN 'valmc_sq'
(
     clk,
     start_acq,
     choix_ram,
     burst,
     sync_comp,
     visu,
     vsync,
     transf_ext,
     fin_compt,
     R/W,
     C1,
     C2,
     /reset_all,
     val_compt[17..0]: INPUT;
```

```
    clk_can,
    /we_ram,
    /we_ram2,
    /burst,
    /sync_comp,
    cs0,
    cs1,
    cs2,
    /oe_can,
    /oe_ram,
    /oe2,
    fin,
    selection,
    clk_cna,
    reset,
    increm,
    copy,
    adresse_out[16..0]  : OUTPUT;
)

VARIABLE
    ss : MACHINE WITH STATES (s0, s1, s2, s3, s4, s5,s6, s7, s8,
s9, s10, s11, s12,s13,s14,s15,s16,s17, s18, s19, s20, s21);

    tempo[11..0]:DFF;   % tempo d'attente pour echantilloner
                        uniquement les infos de luminance ainsi que
                        pour se positionner en début de la première
                        image%
    count_adr[16..0]: DFF;
    dcs0, dcs1, dcs2: DFF;
    arret : DFF;
    num_lig[8..0]: DFF; % numero de la ligne echantillonée ou
                        affichée %
    front : DFF;    % permet d'identifier le front montant de la
                synchro horizontale %
```

```
    bascule1, bascule2 : DFF;  % bascules utilisées pour réaliser
                               la division de freq / 4 %

BEGIN
    ss.clk = clk;
    count_adr[].clk = clk;
    tempo[].clk = clk;
    arret.clk = clk;
    num_lig[].clk = clk;
    front.clk = clk;

    ss.reset=!/reset_all;
    adresse_out[] =count_adr[];

    dcs0.clk = clk;
    dcs1.clk = clk;
    dcs2.clk = clk;
    cs0 = dcs0.q;
    cs1 = dcs1.q;
    cs2 = dcs2.q;
    % complémentation des sorties %
    /burst = !burst;
    /sync_comp = !sync_comp;



    % réalisation de clk_can = clk / 4 %
    bascule1.clk = clk;
    bascule1.d=!bascule1.q;
    bascule2.clk = bascule1.q;
    bascule2.d=!bascule2.q;
    clk_can = bascule2.q;

    CASE ss IS

      WHEN s0 => % Debut et initialisations %
                 % Mode transparent %
          IF (C1) THEN  % window 1 du 68HC11K1 sélectionné: RAM 0
```

```
                ou RAM 1%
    IF (choix_ram) THEN
        dcs0 = GND;
        dcs1 = VCC;
    ELSE
        dcs0 = VCC;
        dcs1 = GND;
    END IF;
    /oe_ram = GND;
ELSE          % /C1 %
    dcs0 = GND;
    dcs1 = GND;
    /oe_ram = VCC;
END IF;

IF (C2) THEN  %En mode transparent on valide ici la
              win. 2 %
    dcs2=VCC;
    count_adr[12..0]=val_compt[12..0];
    count_adr[16..13]=val_compt[17..14];
    /oe2 = GND;
ELSE          % /C2 %
    dcs2=GND;
    count_adr[13..0]=val_compt[13..0];
    count_adr[16..14]=val_compt[17..15];
    /oe2 = VCC;
END IF;

tempo[]=0;
num_lig[]=0;
selection = GND; % affichage en sortie de la
                 synchronisation uniquement %
/oe_can = VCC;
/we_ram = R/W;
/we_ram2 = R/W;
fin = VCC;
clk_cna = VCC; % latched mode <=> sortie bloquée sur
```

```
                derniere donnée %
    reset = VCC;
    increm = GND;
    copy = VCC;   % copy = VCC uniquement à l'état s0 pour
                  recopier adr_in %
    arret = GND;

    IF (!start_acq) & (!visu)& (!transf_ext) THEN
       ss = s0;
    ELSIF (start_acq) & (!visu) & (!transf_ext)THEN
       front = vsync;
       ss = s1;
    ELSIF (!start_acq) & (visu) & (!transf_ext)THEN
       front = vsync;
       ss = s10;
    ELSIF (!start_acq)& (!visu) & (transf_ext) THEN
       ss = s18;
    END IF;


%                                          %
%            etats pour acquisition        %
%                                          %


WHEN s1 => %  Attente de la nouvelle trame %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    count_adr[] = 0;
    tempo[] = TEMPO_PREM_LIG ;  % temporisation pour se
                                positionner juste avant la
                                première ligne-1 a cause de
                                la décrementation et detect%
    num_lig[] = 0;
```

```
        selection = GND;
        /oe_ram = VCC;
        /oe2 = VCC;
        /oe_can = GND;
        /we_ram = VCC;
        /we_ram2 = VCC;
        fin = GND;
        clk_cna = VCC;
        reset = GND;
        increm = GND;
        arret = arret;
        copy = GND;

        IF ((!vsync)&(front)) THEN % Détection du front
                                     descendant de vsync =>
                                     nouvelle trame %
            ss = s2;
        ELSE
            front = vsync;
        END IF;


    WHEN s2 => % positionnement juste avant la première ligne %

        dcs0 = GND;
        dcs1 = GND;
        dcs2 = GND;
        count_adr[] = count_adr[];
        tempo[] =  tempo[] -1;
        num_lig[] = NBR_LIG_TRA;
        selection = GND;
        /oe_ram = VCC;
        /oe2 = VCC;
        /oe_can = GND;
        /we_ram = VCC;
        /we_ram2 = VCC;
        fin = GND;
```

```
        clk_cna = VCC;
        reset = GND;
        increm = GND;
        arret = arret;
        copy = GND;

        IF (tempo[] == 0) THEN
            front = burst;
            ss = s3;
        END IF;

    WHEN s3 => % Attente du front montant de la
               synchronisation horizontale%

        dcs0 = GND;
        dcs1 = GND;
        dcs2 = GND;
        count_adr[] = count_adr[];
        num_lig[] = num_lig[];
        selection = GND;
        /oe_ram = VCC;
        /oe2 = VCC;
        /oe_can = GND;
        /we_ram = VCC;
        /we_ram2 = VCC;
        fin = GND;
        clk_cna = VCC;
        reset = GND;
        increm = GND;
        arret = arret;
        copy = GND;

        IF ((!burst)&(front)) THEN  % front descendant de la
                                    synchro verticale%
            ss = s4;
            tempo[] = TEMPO_DEB_INFO;
        ELSE
```

```
        front = burst;
    END IF;

WHEN s4 => % positionnement en début de l'information de
           luminance %
    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    count_adr[] = count_adr[];
    num_lig[] = num_lig[];
    selection = GND;
    tempo[] =  tempo[] -1;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = GND;
    /we_ram = VCC;
    /we_ram2 = VCC;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;

    IF (tempo[] == 0) THEN
        ss = s5;
        front = clk_can;
    END IF;

WHEN s5 => % Attente du front montant de
           l'horloge du CAN %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    count_adr[] = count_adr[];
    num_lig[] = num_lig[];
```

```
        selection = GND;
        /oe_ram = VCC;
        /oe2 = VCC;
        /oe_can = GND;
        /we_ram = VCC;
        /we_ram2 = VCC;
        fin = GND;
        clk_cna = VCC;
        reset = GND;
        increm = GND;
        arret = arret;
        copy = GND;

        IF ((clk_can)&(!front)) THEN % front montant signalant le
                                       1er echantillon de la ligne%
            ss = s6;
            tempo[] = NBR_ECH_LIG;   % nombre d'échantillons par
                                       ligne %
            %reflechir un peu%
        ELSE
            front = clk_can;
        END IF;

    WHEN s6 => %1ere étape d'ecriture: selection de la RAM %

        IF (choix_ram) THEN
            dcs0 = GND;
            dcs1 = VCC;
        ELSE
            dcs0 = VCC;
            dcs1 = GND;
        END IF;

        dcs2 = GND;
        count_adr[] = count_adr[];
        num_lig[] = num_lig[];
        tempo[] = tempo[];
```

```
    selection = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = GND;
    /we_ram = VCC;
    /we_ram2 = VCC;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;
    ss = s7;

WHEN s7 => % 2eme etape d'ecriture: sauvegarde des données %

    dcs0 = dcs0;
    dcs1 = dcs1;
    dcs2 = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = GND;
    /we_ram2 = VCC;
    /we_ram = GND;% écriture %
    count_adr[] = count_adr[];
    num_lig[] = num_lig[];
    tempo[] = tempo[];
    selection = GND;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;
    ss = s8;

WHEN s8 => % 2eme etape d'ecriture: Incrementation adresse %
```

```
    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = GND;
    /we_ram = VCC;
    /we_ram2 = VCC;
    num_lig[] = num_lig[];
    selection = GND;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
    arret = arret;
    copy = GND;
    increm = VCC;
    count_adr[] = val_compt[16..0];
    tempo[] = tempo[] - 1;
    ss = s9;

WHEN s9 => % Début numérisation: front montant de la CLK du
           CAN %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = GND;
    /we_ram = VCC;
    /we_ram2 = VCC;
    count_adr[] = count_adr[];
    selection = GND;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
```

```
            increm = GND;
            copy = GND;

            IF (tempo[]==0) THEN
                tempo[] = NBR_ECH_LIG;

                IF (num_lig[]==0) THEN
                    num_lig[] = NBR_LIG_TRA;

                    IF (arret) THEN   % deuxième trame echantillonée %
                        ss = s0;   % nécessité d'une étape d'arrêt????
%
                    ELSE
                        arret = VCC;
                        front = vsync;
                        ss = s1;
                    END IF;
                ELSE
                    arret = arret;
                    num_lig[] = num_lig[] -1;
                    front = burst;
                    ss = s3;
                END IF;
            ELSE
                arret = arret;
                tempo[] = tempo[];
                num_lig[] = num_lig[];
                ss = s6;
            END IF;

        %                                       %
        %             etats pour visu           %
        %                                       %

        WHEN s10 => %  Attente de la nouvelle trame %

            dcs0 = GND;
```

```
        dcs1 = GND;
        dcs2 = GND;
        count_adr[] = 0;
        tempo[] = TEMPO_PREM_LIG ;  % temporisation pour se
                                    positionner juste avant la
                                    première ligne%
        num_lig[] = 0;
        selection = GND;
        /oe_ram = VCC;
        /oe2 = VCC;
        /oe_can = VCC;
        /we_ram = VCC;
        /we_ram2 = VCC;
        fin = GND;
        clk_cna = VCC;
        reset = GND;
        increm = GND;
        arret = arret;
        copy = GND;

        IF ((!vsync)&(front)) THEN  % Détection du front
                                    descendant de vsync
                                    => nouvelle trame %
           ss = s11;
        ELSE
           front = vsync;
        END IF;

    WHEN s11 =>   % positionnement avant la première ligne %

        dcs0 = GND;
        dcs1 = GND;
        dcs2 = GND;
        count_adr[] = count_adr[];
        tempo[] =  tempo[] -1;
        num_lig[] = NBR_LIG_TRA;
        selection = GND;
```

```
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = VCC;
    /we_ram = VCC;
    /we_ram2 = VCC;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;

    IF (tempo[]==0) THEN
        front = burst;
        ss = s12;
    END IF;

WHEN s12 =>   % Attente du front montant de la
              synchronisation horizontale%

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    count_adr[] = count_adr[];
    num_lig[] = num_lig[];
    selection = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = VCC;
    /we_ram = VCC;
    /we_ram2 = VCC;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;
```

```
    IF ((!burst)&(front)) THEN  % front montant de la synchro
                                verticale%
        tempo[] = TEMPO_DEB_INFO;
        ss = s13;
    ELSE
        front = burst;
    END IF;

WHEN s13 =>   % positionnement en début de l'information de
              luminance %
    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    count_adr[] = count_adr[];
    num_lig[] = num_lig[];
    selection = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = VCC;
    /we_ram = VCC;
    /we_ram2 = VCC;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;

    IF (tempo[]==0) THEN
        ss = s14;
        tempo[] = NBR_ECH_LIG;   % nombre d'échantillons de la
                                 ligne à afficher%
    ELSE
        tempo[] =  tempo[] -1;
    END IF;
```

```
WHEN s14 =>   % 1ere étape de lecture: selection de la RAM %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = VCC;   % sélection de la RAM 2 pour la lecture %
    count_adr[] = count_adr[];
    num_lig[] = num_lig[];
    tempo[] = tempo[];
    selection = selection;
    /oe_ram = VCC;
    /oe2 = GND;
    /oe_can = VCC;
    /we_ram = VCC;
    /we_ram2 = VCC;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;
    ss = s15;

WHEN s15 =>   % 2eme etape de lecture: lecture par le CNA %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = VCC;
    /oe_ram = VCC;
    /oe2 = GND;
    /oe_can = VCC;
        /we_ram2 = VCC;
    /we_ram = GND;% écriture %
    count_adr[] = count_adr[];
    num_lig[] = num_lig[];
    tempo[] = tempo[];
    selection = selection;
    fin = GND;
```

```
    clk_cna = GND;
    reset = GND;
    increm = GND;
    arret = arret;
    copy = GND;
    ss = s16;

WHEN s16 =>   % 3eme etape de lecture: inhibition du CNA %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
    /oe_can = VCC;
    /we_ram = VCC;
    /we_ram2 = VCC;
    count_adr[] = count_adr[];
    num_lig[] = num_lig[];
    tempo[] = tempo[] - 1;   %nombre restant d'echantillons %
    selection = VCC;
    fin = GND;
    clk_cna = VCC;
    reset = GND;
    arret = arret;
    increm = GND;
    copy = GND;
    ss = s17;

WHEN s17 =>   % 4 eme etape de lecture: incrémentation
              d'adresse %

    dcs0 = GND;
    dcs1 = GND;
    dcs2 = GND;
    /oe_ram = VCC;
    /oe2 = VCC;
```

```
/oe_can = GND;
/we_ram = VCC;
/we_ram2 = VCC;
selection = selection;
fin = GND;
clk_cna = VCC;
reset = GND;
copy = GND;

increm = VCC;
count_adr[] = val_compt[16..0];

IF (tempo[] == 0) THEN   %ligne terminée d'être affichée%
    tempo[] = NBR_ECH_LIG;
    selection = GND;

    IF (num_lig[]==0) THEN
        num_lig[] = NBR_LIG_TRA;

        IF (arret) THEN   % deuxième trame echantillonée %
            ss = s0;   % nécessité d'une étape d'arrêt? %
        ELSE
            arret = VCC;
            front = vsync;
            ss = s10;
        END IF;
    ELSE
        arret = arret;
        num_lig[] = num_lig[] - 1;
        front = burst;
        ss = s12;
    END IF;
ELSE
    arret = arret;
    tempo[] = tempo[];
    num_lig[] = num_lig[];
    ss = s14;
```

```
            END IF;



%-------------------------------------------------------------------%
%        TRANSFERT D'IMAGE DE RAM 0 OU 1 VERS RAM2                  %
%-------------------------------------------------------------------%

        WHEN s18 =>   % Initialisation du transfert %

            clk_cna=VCC;
            /we_ram=VCC;
            /we_ram2=VCC;
            dcs0.d=GND;
            dcs1.d=GND;
            dcs2.d=GND;
            /oe_can=VCC;
            /oe_ram=VCC;
            /oe2=VCC;
            reset=VCC; % Remise à zero du compteur d'adresse %
            increm=GND;
            copy=GND;
            count_adr[]=0;
            selection = GND;
            num_lig[] = 0;
            tempo[] = 0;
            fin=GND;
            ss=s19;


        WHEN s19 =>   % Début du cycle de transfert %

            % On prépare la lecture pour le coup d'horloge suivant. %

            clk_cna=VCC;
            /we_ram=VCC;
            /we_ram2=VCC;
```

```
    dcs2=GND;
    /oe_can=VCC;
    /oe_ram=VCC;
    /oe2=VCC;
    reset=GND;
    increm=GND;
    copy=GND;
    count_adr[]=count_adr[];
    selection = GND;
    num_lig[] = 0;
    tempo[] = 0;
    fin=GND;

    IF (choix_ram) THEN
        dcs0=GND;
        dcs1=VCC;  % donnée RAM 1 en sortie %
    ELSE
        dcs0=VCC;  % donnée RAM 0 en sortie %
        dcs1=GND;
    END IF;

    ss=s20;


WHEN s20 =>   % lecture dans une des RAM 0 ou 1 %

    dcs0=dcs0;
    dcs1=dcs1;
    /oe2=VCC;
    /we_ram2=VCC;
    dcs2=VCC;  % On prépare l'écriture dans la RAM2 %
    clk_cna=VCC;
    /we_ram=VCC;
    /oe_can=VCC;
    /oe_ram=GND;  % On permet de lire sur RAM 0 ou 1 %
    reset=GND;
    increm=GND;
```

```
    copy=GND;
    count_adr[]=count_adr[];
    fin=GND;
    selection = GND;
    num_lig[] = 0;
    tempo[] = 0;
    ss=s21;


WHEN s21 =>   % écriture RAM2  %

    clk_cna=VCC;
    /oe_can=VCC;
    fin=GND;
    /we_ram=VCC;
    /we_ram2=GND;
    /oe_ram=GND;
    /oe2=VCC;
    dcs0=GND;  % Pour le coup d' H suivant %
    dcs1=GND;
    dcs2=GND;
    copy=GND;
    reset=GND;
    selection = GND;
    num_lig[] = 0;
    tempo[] = 0;

    IF (!fin_compt) THEN
        increm=VCC;
        count_adr[]=val_compt[16..0];
        ss=s19;
    ELSE
        increm=GND;
        count_adr[]=val_compt[16..0];
        ss=s0;
    END IF;
```

```
END CASE;
END;
```

5.3 Fichier matlab

%%

%% FICHIER MATLAB D'ANALYSE DE VARIATION DE LUMINANCE %%

%% ENTRE DEUX IMAGES SUCCESSIVES %%

%%

% Lecture fichiers image binaire en 256 niveaux de gris

% Récupération des informations de deux images BMP dans les matrice I1 et I2

% Nous prenons dans cet exemple les deux premières images de la séquence voleur

[I1,MAP] = BMPREAD ('C:\PROJET~1\9798\VIDEO\MATLAB\voleur\image1.bmp');

[I2,MAP] = BMPREAD ('C:\PROJET~1\9798\VIDEO\MATLAB\voleur\image2.bmp');

% Mise a zéro des matrices de moyenne M1 et M2, de somme S1 et S2 et de variation V

clear M1

clear S1

clear M2

clear S2

clear V

% Calcul des moyennes et sommes des blocs dans M1,M2,S1,S2 (ici, taille image W=191 H=145)

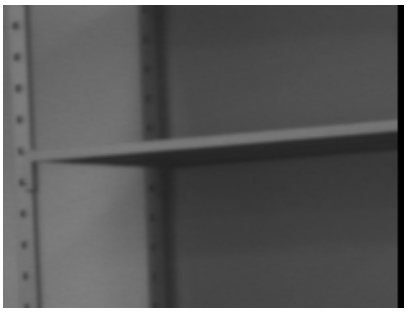
```
% Par blocs 36*25

% Calcul des matrices M1 et S1

for (C=1:7)
 for (L=1:4)
      M1(L,C)=mean2(I1((L-1)*36+[1:26],(C-1)*25+[1:25]));
      S1(L,C)=M1(L,C)*25*36;
 end
end

% Calcul des matrices M2 et S2 et de la matrice variation V

for (C=1:7)
 for (L=1:4)
      M2(L,C)=mean2(I2((L-1)*36+[1:26],(C-1)*25+[1:25]));
      S2(L,C)=M2(L,C)*25*36;
      V(L,C)=S2(L,C)-S1(L,C);
 end
end

% Niveau moyen de la somme

Niv=mean2(S1);




% Recherche de la variation maximale et minimale

min=V(1,1);
```

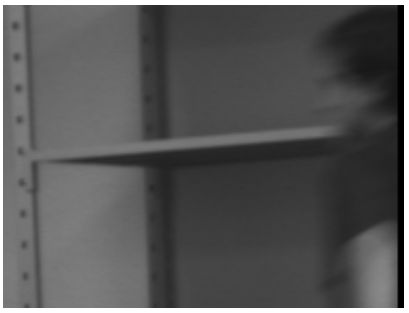
```
max=V(1,1);

for (C=1:7)
 for (L=1:4)
       if abs(V(L,C))>abs(max)
          max=V(L,C);
       end
       if V(L,C)<min
          min=abs(V(L,C));
       end
 end
end

% Affichage des données recueillies
% Les variations sont à interpréter en %

max
min
Niv

varmax=max/Niv
varmoy=mean2(V)/Niv
```

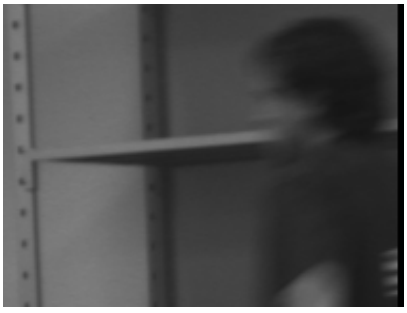
5.4 Représentation des séquences vidéo

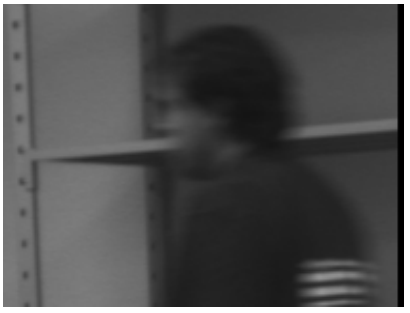
La numérisation des images pour chacune des séquences s'est faite au format bmp via Matrox Inspector.

5.4.1 Séquence 1

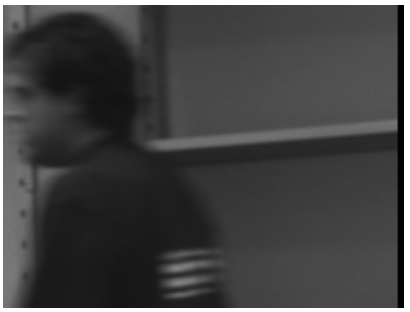
1. 

2. 

3. 

4. 

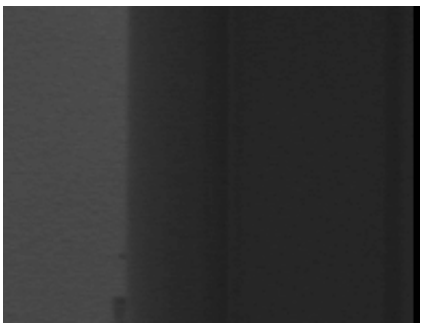
5. 

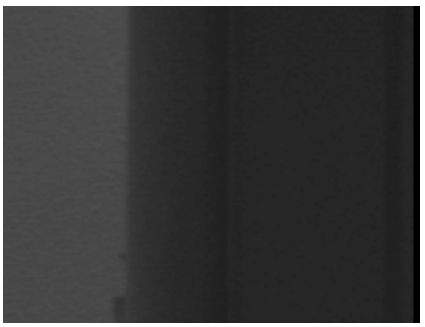
6. 

7. 

5.4.2 Séquence 2

La représentation papier de cette séquence vidéo nous a obligé à un léger éclaircissement afin de bien distinguer la scène. On parvient ainsi nettement à distinguer le câble noir qui se déplace horizontalement de gauche à droite.

1. 

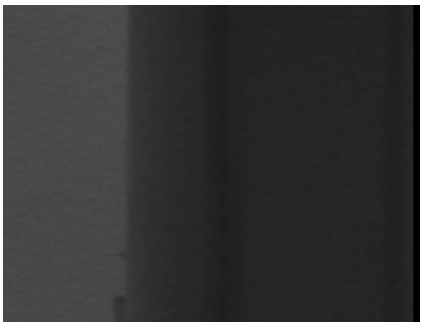
2. 

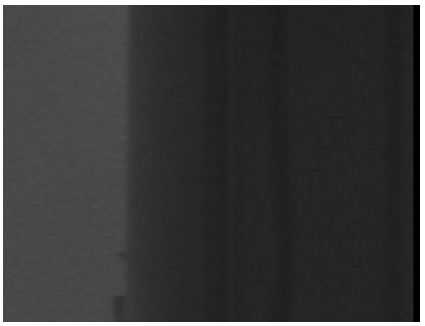
3. 

4. 

5. 

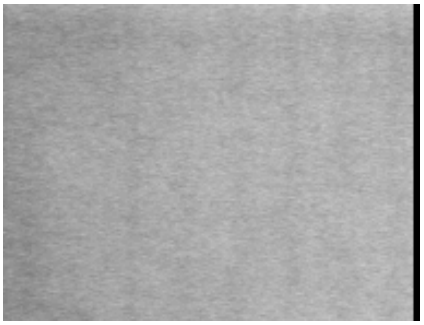
6. 

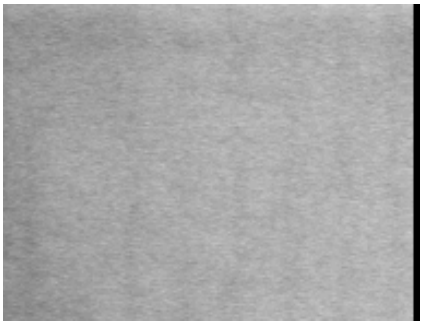
7. 

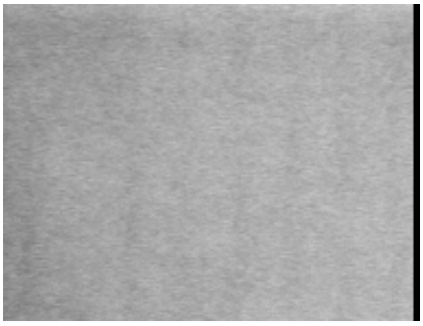
8. 

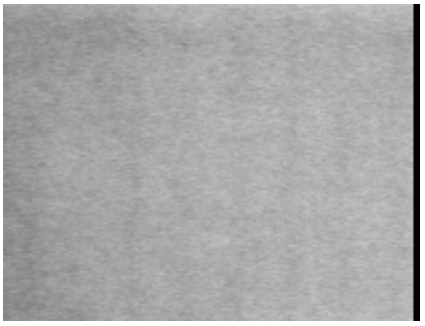
5.4.3 Séquence 3

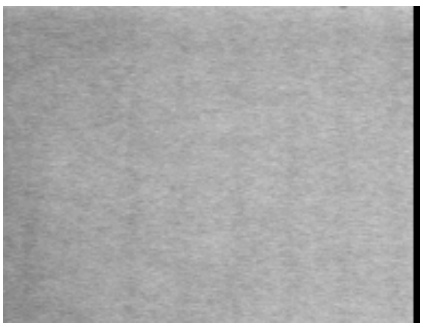
La représentation vidéo papier de cette séquence vidéo nous a obligé à un léger renforcement du contraste afin de bien distinguer la scène. On peut alors distinguer l'apparition de l'extrémité du tube de plastique blanc à l'image 6.

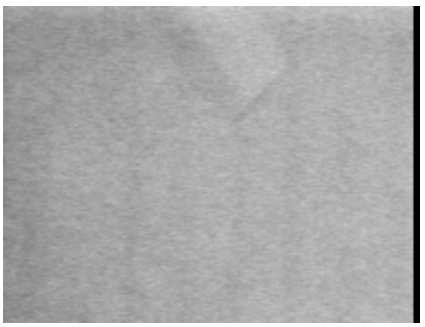
1. 

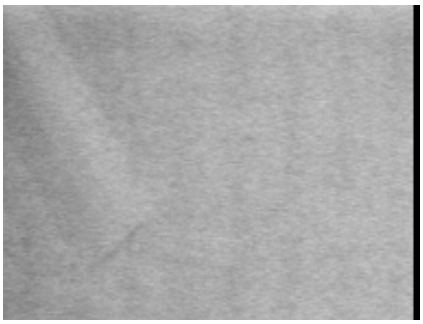
2. 

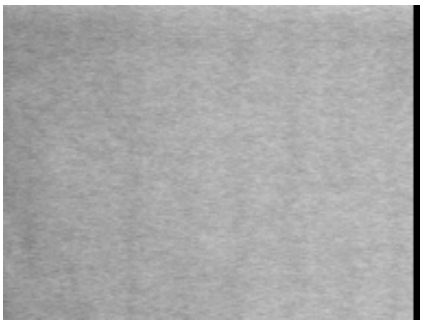
3. 

4. 

5. 

6. 

7. 

8. 

5.4.4 Séquence 4

Ici, les images ont été prises tous les 30 secondes. La scène filmée se situe près d'une fenêtre donnant sur un ciel changeant. Les différences de luminance engendrées ne sont pas discernables.

1. 

2. 

3. 

4. 

5. 

6. 

7.