Germain Barret
Estelle Brémont
Emeric Golfier
Yan Zhang

Mourad Benoussaad
Gergana Gocheva
Guillaume Lambert

PROJET DE TELEDETECTION

Master II Pro SIS Option Imagerie Numérique, Cours de Télédétection

1°) APPROCHE DU PROBLEME :

1.1°) Objectif du projet:

1.2°) Présentation de l'approche du problème

2°) TRAITEMENT DU PROJET :

2.1°) Extraction d'images / Lecture de vidéo

2.2°) Traitement des images en niveaux de gris & Extraction des objets mouvants

2.3°) Utilisation des Morphomaths : Traitement des images bruitées

2.4°) Etiquetage

2.5°) Reconnaissance et suivi d'objet

2.6°) Analyse de l'anormalité

3°) OPTIMISATION DU CODE

4°) VERS UNE AMELIORATION DES ALGORITHMES / AUTRES APPROCHES

1°) APPROCHE DU PROBLEME :

1.1°) Objectif du projet :

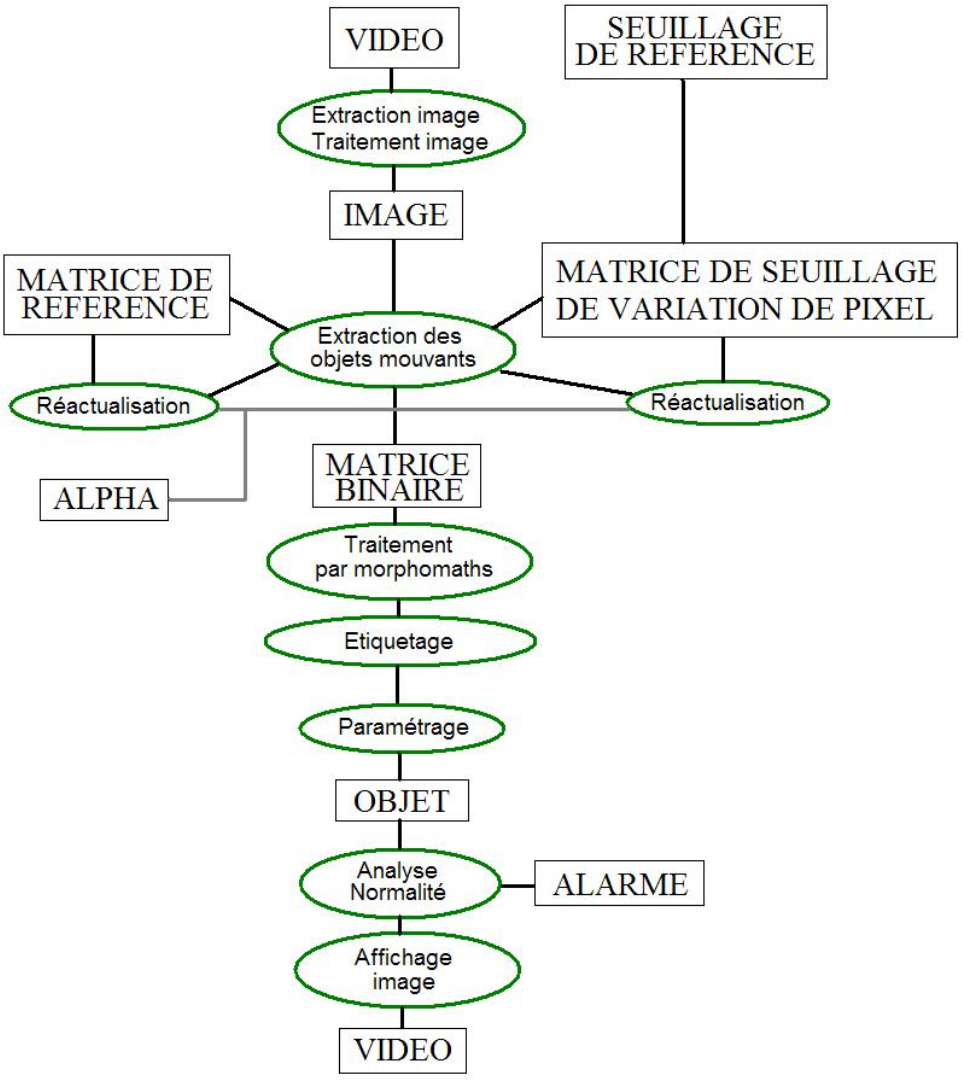
L'objectif du projet consiste à extraire un ensemble d'images numériques d'une capture vidéo d'un environnement fixe. Dans ses images, on cherche à localiser des objets en mouvement pour en déterminer une évolution que l'on qualifie de normal ou d'anormale.

1.2°) Présentation de l'approche du problème

Une première réflexion a permis de définir l'objectif du projet comme une suite d'étapes :

- Extraction des images à traiter à partir d'une vidéo au format *.avi*, puis mise en forme de ces images afin d'accroître la pertinence des données qui en seront extraites aux étapes suivantes.
- Détection de l'objet en mouvement (personne qui se déplace), pour chaque image.
- Traitement du résultat par les algorithmes de morphologie mathématique afin de ne garder que les informations appropriées.
- Tirer des paramètres pertinents d'une évolution d'un même objet entre chaque image.
- Analyser les paramètres pour en déduire la normalité ou non de l'activité du personnage.

Ces activités sont ordonnées selon l'organigramme suivant :



2°) TRAITEMENT DU PROJET :

2.1°) Extraction d'images / Lecture de vidéo

Estelle Brémont
5 semaines

2.1.1°) Travaux, résultats, et intérêts

Dans un premier temps, tous les travaux préliminaires ont été réalisés sur un ensemble d'images de format *.pnm* (plus précisément *.pgm* [format d'images en niveau de gris] et *.ppm* [format d'images en couleurs]). Images extraites de vidéos par des logiciels, plus ou moins puissants.
Un grand nombre de réglages des paramètres de sortie ont été envisagés et appliqués comme montré dans les paragraphes suivants :

Travail sur des images en niveau de gris

Ce travail simplifie les calculs de traitement d'image, en particulier sur les soustractions d'images.
Il est important de préciser que le passage couleur -> niveau de gris ne se fait pas par une moyenne des composantes sur chaque couleur, mais par une pondération précise pour chaque couleur (fondée sur la luminance naturelle), soit :

*Gris(Pixel_i) = 0.30 * Rouge (Pixel_i) + 0.59 * Vert(Pixel_i) + 0.11 *Bleu (Pixel_i)*

Travail sur des images 256 couleurs avec gestion de palette

Ce travail permet tout d'abord d'avoir peu de couleurs, donnant ainsi le même bénéfice qu'en niveau de gris. Mais il permet également de garder les informations de couleurs qui peuvent être déterminantes pour la distinction de deux personnes dans une scène.
Pour ce faire, la vidéo a été convertie en fichier *gif* animé, puis les images ont été extraites.

Extraction en images ppm « normales » et travaux en niveaux de gris.

Cette solution a été retenue pour la plus grande partie des tests d'algorithme. En effet, stocker de l'information de couleur la plus précise possible s'avère être une nécessité, en particulier lorsque les habits des personnes en mouvement se fondent dans le décor. Toutefois, effectuer tous les travaux (que ce soit pour les morphologies mathématiques, pour l'étiquetage, etc...) sur des images en niveaux de gris permet un allègement non négligeable des calculs (et une diminution de la largeur des matrices par trois !).

2.1.2°) Problèmes rencontrés

Même si les résultats de traitement donnaient des résultats assez concluants, de manière très concrète, la mise en oeuvre et les nombreux tests se heurtaient à deux gros problèmes (et non des moindres) :

Un problème de lenteur

Cela est provoqué par l'ouverture, la lecture, l'écriture d'un ensemble d'images, en particulier si ces images sont sur un périphérique amovible.

Un problème d'encombrement du disque dur

En effet, le stockage d'un ou de plusieurs « ppms – vidéos », selon la longueur de la vidéo et la fréquence de capture, prenait entre 40Mo et 230Mo. Ce qui est considérable.

2.1.3°) Résolution des problèmes

Un système de lecteur de vidéo *.avi* a été mis en place. Il résout pratiquement tout l'ensemble des problèmes posés par les images et n'ajoute pas d'inconvénient, si ce n'est une difficulté de programmation.
Le lecteur vidéo implémenté a été programmé en c++ avec un support openGl, aidé par une librairie de gestion des fichiers avi/codecs pour avi (avifile, version 0.7). Il consiste en lecture, puis affichage par texture (nécessaire pour optimiser la vitesse de traitement) de chaque frame récupérée de la vidéo.

Principe :

Récupération des images (des N premières) qui permettront l'initialisation de la référence réf

répéter N fois
lire frame
pour chaque pixel P (triplet RVB) de Frame :
stockage dans ref de (ex-valeur de ref en P + Passage en Niveau de gris(P))
fin pour

réactualise le pointeur de frame vers la prochaine frame
fin répéter

pour chaque élément e contenu dans la matrice ref,
e <- e/N
fin pour

Traitement général pour la gestion de la détection

Tant que la vidéo n'est pas finie
Récupération d'une frame F
Stockage d'une copie de F en niveau de gris dans une matrice img
Récupération des propriétés de la frame
Traitement détection et modification sur la matrice img
Modification de la texture d'affichage en fonction des données de img
Affichage de la texture
Ré-actualisation du pointeur de frame vers la prochaine frame
fin tant que

Il est important de préciser que, pour des raisons de temporisation et de respect (tant que possible) de la vitesse intrinsèque de la vidéo (donnée par le nombre de frame par secondes), une forme de chronométrage du traitement est faite. On en tient alors compte pour lancer la vidéo suivante.
Ainsi, nous avons accès à des images de qualité, nous avons minimisé l'espace disque nécessaire, et nous arrivons à minimiser les accès disques. Nous avons donc un bon support pour lancer la détection.

2.2°) Traitement des images en niveaux de gris & Extraction des objets mouvants

Emeric Golfier / Estelle Brémont

Tout le long du processus de développement

Cette étape consiste à prendre en entrée une image en niveau de gris, et à retourner en sortie une matrice binaire, avec une valeur 1 pour un pixel repéré comme appartenant potentiellement à une personne en activité, et une valeur 0 sinon.

2.2.1°) Méthode utilisée

La méthode retenue est issue des articles donnés en référence, et notamment la formule :

|Matrice Image [en x,y] – Matrice de Référence [en x,y]| > Matrice Seuil [en x,y]

Nous effectuons donc une comparaison pixel par pixel de l'image en cours d'étude avec une valeur de référence stockée par une matrice. Si la comparaison dépasse une valeur seuil stockée dans une matrice, alors on considère que le pixel est potentiellement intéressant.

La partie la plus délicate du problème réside dans la gestion des seuils, car elle est le premier maillon dans le processus de détection, et dépend entièrement de la qualité vidéo et du lieu filmé. Dans notre cas, les vidéos de référence fournies ont toutes lieu dans la même salle fermée, possédant plusieurs zones critiques par leur contraste, leur propension aux reflets et aux ombres, ou leur couleur et leur structure.

2.2.2°) Initialisation des matrices

Les Matrice de Référence et Matrice Seuil sont donc des matrices de même taille qu'une image. Cette matrice image est elle même représentée par une matrice dont chaque élément contient une valeur chromatique correspondant à son niveau de gris.
Chaque valeur de chaque pixel de la Matrice de Référence est issue de la moyenne des valeurs des pixels correspondants sur les 7 premières images de la vidéo (où il n'y a pas de variation importante dans l'image).

Les articles de référence disent utiliser une valeur initiale de toutes les valeurs de la Matrice Seuil égale à 50. Or, nos expérimentations nous indiquent qu'une valeur de 20 offre de meilleurs résultats dans notre cas. De plus, une étude des résultats de la Matrice Seuil en sortie de traitement d'une série d'images montrait que chaque valeur avait évolué jusqu'à arriver à hauteur de 50% de la valeur initialement mise. Et ce quelque soit la valeur seuil mise en initialisation (dans l'étude on a testé pour 50 et 20).

Ainsi, on peut supposer qu'il n'existe pas de valeur-seuil optimale, et qu'il nous revient de choisir la valeur en fonction d'une étude des résultats des images traitées et du traitement ultérieur de la matrice renvoyée en résultat. Une étude nous a donc permis dans un premier temps de définir une valeur de 20 dans un cas 'générique' du traitement de l'image.

2.2.3°) Un problème de seuillage

Une étude empirique de seuillage a permis de détecter le fait qu'il faille probablement adapter la valeur initiale de chaque valeur de la Matrice de Seuillage en fonction des zones qui se détachent.

Par exemple, quand le personnage passe dans une zone bien éclairée, on peut se permettre un seuil élevé, sans quoi nous 'capturons' en résultat l'ombre du personnage sur le sol (ce qui ne nous intéresse pas). Quand il passe dans une zone peu éclairée (par exemple le mur en partie centrale de l'image vers le haut), on constate qu'il vaut mieux utiliser un seuil

faible sous peine de ne pas repérer le passage du personnage si son tee-shirt est d'une couleur proche.

Une analyse plus détaillée du résultat nous a également permis de mettre évidence que pour une même couleur dans l'image des seuils différents peuvent être nécessaires afin de faciliter la détection. Par exemple, un seuil qui permettra de bien détecter une personne venant de rentrer par la porte puis passant derrière la table (problème de contraste avec le mur), s'avèrera générer beaucoup de bruit et de composantes superflues dans une trajectoire dans les escaliers (qui possède de mêmes composantes chromatiques).

Ainsi, les deux approches ont été développées:

Le seuillage géographique: par initialisation de la Matrice Seuillage avec des valeurs dépendant de zones géographiques prédéfinies.

Le seuillage chromatique: initialiser la Matrice Seuillage en fonction des niveaux de gris de chaque pixel de la Matrice de Référence obtenue des premières images.

De plus, le challenge est double : non seulement il s'agit d'arriver à détecter une ou plusieurs personnes en mouvement, mais il va falloir également trouver un traitement supprimant au maximum tout le bruit généré par la vidéo et les variations entre images, afin d'épurer le résultat et faciliter les étapes suivantes.

2.2.4°) Un seuillage adapté

Une première étude a permis de détecter, que sur les escaliers, par exemple, un seuillage de zone donnait de meilleurs résultats qu'un seuillage global sur valeurs (nécessite un étalonnage particulier des valeurs de seuillage), même si le résultat n'est pas non plus exceptionnel. Des problèmes restent présents : les 'patates' de 2 personnes marchant côte à côte peuvent se rejoindre. Le problème vient des valeurs chromatiques trop irrégulières et trop peu homogènes de l'escalier, ce qui pose problème au principe même de seuillage. La recherche d'un seuil optimal se transformant donc en la recherche d'un seuil suffisant. En effet, le problème de fusion de personnes côte à côte sera laissé au traitement de la gestion des recoupements de 'patates' : deux 'patates' qui se recoupent sont considérées comme soudées, et deux patates soudées sont considérées comme une seule patate, soit un seul objet que l'on étudiera en tant qu'objet unique par la suite. Le problème est donc contourné.

Ainsi, 3 seuillages ont été définis, chacun ayant points forts et points faibles:

Un seuillage géographique spécial pour l'escalier (seuil 1), permettant de gérer relativement convenablement les valeurs chromatiques particulière de cette zone:

Les valeurs de la zone de l'escalier >=60 ont un seuil fixé à 40.

Les valeurs entre 60 et 20 (exclues) ont un seuil fixé à 10.

Les valeurs <=20 ont un seuil fixé à 0.

Un seuillage chromatique multiple défini sur des intervalles de valeurs chromatiques (seuil 2). Ces intervalles étant obtenu en récupérant les valeurs des zones considérées comme critiques :

- le bas de la porte : les valeurs au delà de 120 sont fixée à 80.
- les zones de reflets: les valeurs entre 100 et 120 sont fixées à 60.
- la porte: les valeurs entre 90 et 60 ont un seuil fixé à 5.
- le mur avec le store: les valeurs <=60 sont fixées à 7
- l'escalier : traité à part avec le seuillage géographique précédent.
- le reste ayant une valeur de seuillage définie à 20.

Un seuillage chromatique issu de la matrice de référence (seuil 3). Plusieurs tentatives de traitements automatique de définition linéaire, polynomiale ou exponentielle des valeurs de seuillage ont permis de mettre à jour un rapport donnant une valeur de seuil chromatique global intéressant : pour chaque pixel, la valeur de seuil est égale à la valeur de référence/4.

Une comparaison de ces 3 seuillages appliqués à toute l'image a permis de définir un classement de meilleur rendu sur deux exigences : en global (quantité et type de bruit généré) et sur les 'patates' (allure et taille exploitable, connexité récupérable), et ce sur les zones critiques de la scène. Soit le tableau suivant (remarque : il y a eu des ex aequo):

	SEUIL 1		SEUIL 2		SEUIL 3	
Zones :	Globalement	Sur patates	Globalement	Sur patates	Globalement	Sur patates
Escalier	2	2	3	3	1	1
Mur	3	2	2	3	1	1
Porte	2	3	3	1*	1	1
Reflet	2	1	3	1*	1	1
Table	1	1	3	3	2	2
Cas Générique	1**	2	3	3	1**	1

* : Détoure très bien, le personnage au niveau des jambes.
** : Equivalentes si bruit bien géré.

Ainsi, le seuil 3 se démarque nettement des autres, mais il serait intéressant de récupérer la capacité du seuil 1 à bien gérer les zones sombres et celle du seuil 2 à bien gérer les zones de reflet.

D'où la mise en place du seuillage utilisé :

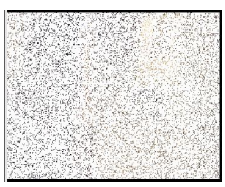
- Pour les valeurs chromatiques >120 : on fixe le seuil à 80 (récupéré du seuil 2)
- Pour les valeurs >60 et <=120 : on fixe le seuil à ref/4
- Pour les valeurs <=60 : on fixe le seuil à 10 (récupéré du seuil 1)

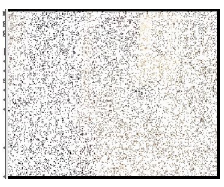
On constate donc que le seuillage à ref/4 donne des valeurs entre 15 et 24, soient proches de 20, valeur optimale trouvée empiriquement 'à l'œil' au début, ce qui permet de valider le résultat. Les comparaisons des méthodes sont dans l'Annexe 1.

2.2.5°) Réactualisation des matrices

Les instruments de comparaison que sont la Matrice de Référence et la Matrice de Seuillage sont réactualisés en utilisant les formules issues des articles de référence, et selon un coefficient ALPHA, représentant une inertie de la matrice de référence vis à vis de la réactualisation obtenue avec l'image courante.

Or nous constatons que, dans le cadre qui nous est imposé pour ce projet, nous obtenons de meilleurs résultats en prenant ALPHA=1. Ce qui signifie que nous ne tenons pas compte des variations temporelles du décors dans lequel se déroulent les évènements. Ainsi, pourrions donc retirer ce paramètre de l'analyse (laissé dans le programme en cas de test de présentation). Nous ne traitons donc qu'une simple soustraction d'image.


ALPHA=0.5


ALPHA=0.9


ALPHA=1


…et même ALPHA=1.1 (absurde)

2.2.6°) Méthodes soulevées non traitées

Afin d'essayer d'affiner des étapes du traitement, une méthode a été proposée pendant une réunion, consistant en la recherche du contour des personnages, puis de le remplir pour obtenir les 'patates'. Ainsi, il y aurait moins de problèmes de fusion entre personnages voisins. Cependant le traitement lourd des contours, la comparaison ardue entre des contours d'images successives, le fait que pour l'analyse de la trajectoire des 'patates' grossières conviennent, et le temps restreint imparti pour le projet ne nous ont pas incité à poursuivre dans cette voie. Le gain semblant trop peu intéressant.

L'utilisation d'une compression à base d'ondelettes des images a été également évoquée. L'avantage étant alors de traiter une image/matrice compressée et gérer donc une homogénéisation des textures des murs, porte ou autres escaliers (éliminant le problème de variation de texture) tout en restant extrêmement sensible aux variations des images. Cependant, l'homogénéisation pouvait aussi 'embarquer' les parties du personnage de couleur trop proche du fond lors de passages critiques ou laisser des variations non voulues ailleurs dans l'image (problème de seuillage global : l'indice de compression est le même dans toute l'image). C'est donc un donné pour un rendu, qui, tenant compte du produit matriciel important que la méthode implique, n'apporte finalement pas d'amélioration suffisante. Un seuillage géographique peut cependant être combiné au seuillage global lors de cette méthode. Cette modification peut probablement rendre l'utilisation de cette méthode extrêmement performante, mais il faudrait alors utiliser les mêmes outils (zonage) qui nous suffisent à eux seuls. La méthode a donc été abandonnée.

2.3°) Utilisation des Morphomaths : Traitement des images bruitées

Mourad Benousaad / Gergana Gocheva

4 semaines + 2 semaines tests

Le traitement des images en niveaux de gris permet l'obtention d'une matrice binaire qui est alors envoyée dans un traitement de morphologie mathématiques.

2.3.1°) Problématique

Les images binaires issues de la vidéo ne devraient contenir que les objets en mouvement. Or, elles présentent les caractéristiques suivantes :

- Elles sont assez bruitées : le bruit se présente sous forme de points dispersés dans l'image.
- Les objets en mouvement à détecter sont: soit incomplets et troués, soit coupés en plusieurs parties distinctes et très proches. Il faut donc les compléter ou les regrouper en un même et seul objet.

2.3.2°) Intérêt des morphomaths

Les traitements par morphologies mathématiques peuvent répondre à ces deux types de problèmes :

- Pour éliminer le bruit on utilisera l'opération d'érosion, en tachant de ne pas perdre l'information utile
- Pour compléter ou regrouper les parties d'un objet en mouvement, on utilisera la dilatation en s'assurant de ne pas augmenter la taille et l'importance du bruit.

2.3.3°) Application des morphomaths dans le cadre du projet

La dilatation : cette opération permet de dilater les objets dans l'image, dans le but de les compléter, de regrouper les objets assez proches. Elle consiste à modifier l'état des pixels concernés par l'application d'un élément structurant sur un objet.

L'érosion : cette opération permet d'éliminer les objets dans l'image dont la taille est inférieure à celle d'un élément structurant choisi. Ainsi, on peut l'utiliser pour éliminer le bruit présent dans l'image mais en réduisant la taille de tous les objets.

2.3.4°) Utilisation des morphomaths et tests réalisés

Objectifs :

Notre choix d'une meilleure opération d'érosion et/ou de dilatation s'est fait d'après les critères suivants :

- Elimination du bruit dans l'image
- Conservation de l'information utile dans l'image (dans notre cas les objets en mouvement)
- Rapidité de traitement pour respecter l'aspect temps réel, sachant que d'autres opérations importantes sont à faire qui nécessitent un temps de calcul considérable).

Nous avons réalisé des tests en considérant les points suivants:

- Nous considérons séparément les opérations de dilatation et d'érosion, pour trouver ensuite un ordre adéquat (dilatation+érosion ou érosion+dilatation), ou une combinaison plus complexe.
- Nous considérons plusieurs types d'élément structurant : carré, croix, rectangle.
- Tester des tailles d'élément structurant différentes
- Tester des éléments structurants symétriques et non-symétriques.

Pour les deux opérations (dilatation et érosion) nous utilisons les étapes suivantes:

1- Balayage de l'image ligne par ligne, et pixel par pixel

2- Pour chaque pixel en cours:

a) Vérifier son état et celui de son voisinage suivant l'élément structurant considéré, ce qui revient à faire une opération logique entre cet ensemble de pixel:
OU logique: pour l'opération de dilatation,
ET logique : pour l'opération d'érosion,

b) Le résultat de cette opération logique définit le nouveau état du pixel en cours.

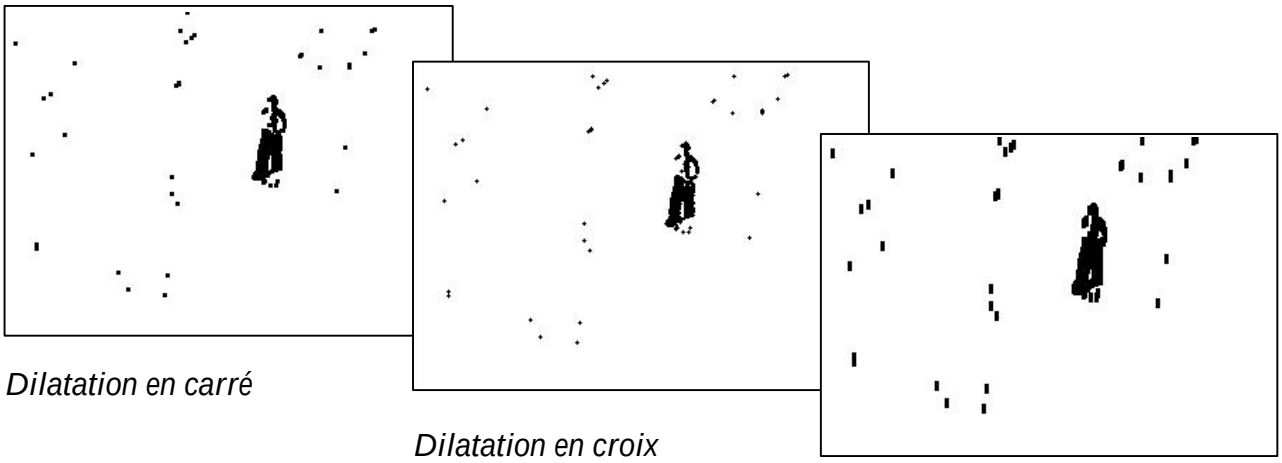
3- Passer aux pixels suivants.

On remarque que pour l'érosion il n'est pas nécessaire de balayer les pixels dont l'état est à 0 (si on suppose que 0 représente le fond) puisque son état reste inchangé après une opération ET logique.
De plus, traiter uniquement les pixels en 1 réduit considérablement la complexité de calcul.

2.3.5°) Résultats Obtenus

Image utilisée comme source :

Résultats de la dilatation:

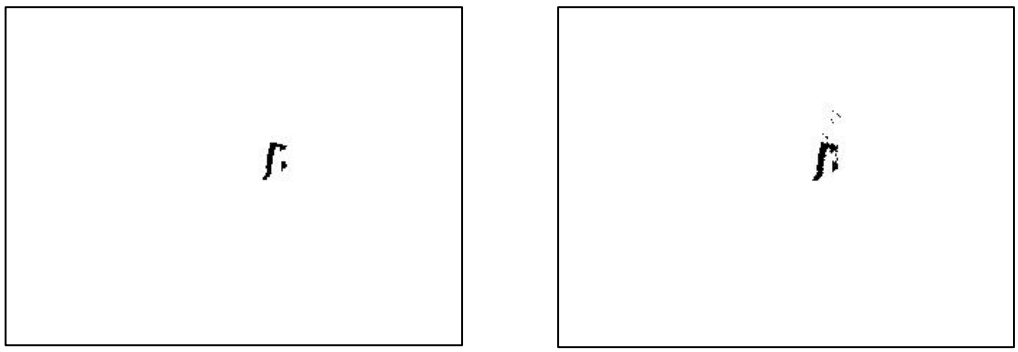


Dilatation en carré

Dilatation en croix

Dilatation en rectangle

Résultats de l'érosion :



Erosion en carré / rectangle

Erosion en croix

2.3.6°) Interprétation des résultats et corrections

Nous avons constaté, après quelques tests, que :

- La dilatation complète les objets manquants et regroupe les parties séparées
- Des éléments structurants de plus grandes tailles donnent de meilleurs résultats, en s'assurant de ne pas trop augmenter la taille de l'objet.
- Les éléments structurants de forme rectangulaire et verticale s'adaptent le mieux à la majorité des images où les objets sont séparés horizontalement.
- L'application d'une érosion en premier élimine le bruit, mais également une grande partie utile de l'objet.

Pour remédier à ce dernier problème, deux solutions peuvent être envisagées:

- L'utilisation de la dilatation en premier pour fermer les objets utiles, l'érosion s'appliquant ensuite. Cette solution nécessite des éléments structurants incompatibles pour la dilatation et l'érosion ; c'est à dire, une dilatation avec un rectangle vertical suivie d'une érosion avec un rectangle horizontal. Cette solution exige d'autres opérations de dilatation, ce qui augmente le temps de calcul.

- L'utilisation d'autres outils d'élimination du bruit sans modifications des objets utiles: en exploitant les caractéristiques du bruit par rapport aux objet utiles. Ces caractéristiques: la taille et le fait que le bruit est isolé (un ou quelques point isolés), nous a permis de mettre en oeuvre un débruiteur qui ne s'occupe que de l'élimination du bruit. Cette solution s'est avérée plus efficace et plus rapide qu'une érosion classique.

En effet, après analyse de ce bruit, il s'est avéré qu'il ne dépassait quasiment jamais les 4 pixels contigus. Par conséquent, un petit algorithme de nettoyage de cette neige parasite, fondé sur la notion de comptage du nombre de voisin a été implémenté de sorte que le bruit disparaisse sans que la forme ne subissent de trop grandes destructions :

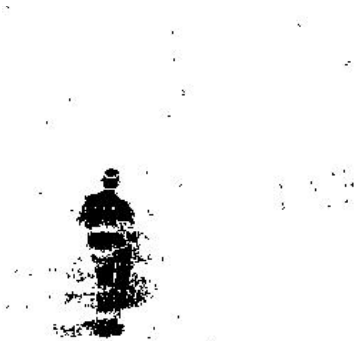
Pour tous les éléments i détectés :
si nombre de voisins 8-connexes de i <=3
alors, suppression de i dans la détection

Cette étape d'élimination du bruit a donc lieu avant la dilatation car elle n'affecte pas les objets utiles et remplace l'érosion sans (trop) attaquer les éléments qui nous intéressent pour la suite.

2.3.7°) Résultats du débruiteur

Image originale

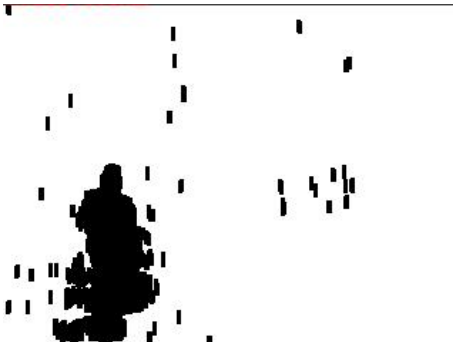
Résultats du débruitage (Anti-parasite) :



Entourage 3*3

Entourage 9*9

Suivis d'une dilatation (élément structurant : largeur: 3 pixels, hauteur: 9 pixels) :



Débruitage 3*3 + dilatation

Débruitage 9*9 + dilatation

2.3.8°) Conclusion

Nous avons utilisé une opération dédiée de débruitage, évitant ainsi l'érosion, plus l'opération de dilatation.

Nous avons appliqué le débruitage avant la dilatation afin d'éviter l'augmentation de la taille des bruits et leurs regroupement en un objet important et difficile à éliminer.

La dilatation ne s'appliquera qu'aux objets utiles restants, et quelques bruits importants qui n'ont pu être éliminés.

Les objets bruits qui n'ont pu être éliminé peuvent être traité et éliminé dans l'étape suivante d'étiquetage.

Les résultats sont satisfaisants, même si certaines images posent ponctuellement quelques problèmes. Cela reste acceptable au vu du nombre important d'images issues d'une vidéo.

2.3.9°) Méthode soulevée non traitée

Guillaume Lambert
3 semaines

On a vu qu'une simple fermeture n'était pas efficace dans l'image bruitée que l'on avait à cette étape de l'algorithme. On remarque également que le personnage est souvent 'coupé' en plusieurs parties. L'objectif est donc, après utilisation d'un pré-débruitage, de programmer une fermeture plus puissante capable de lier des morceaux éparses d'un même élément.
Un autre algorithme utilisant les morphomaths a donc été développé en parallèle afin de:

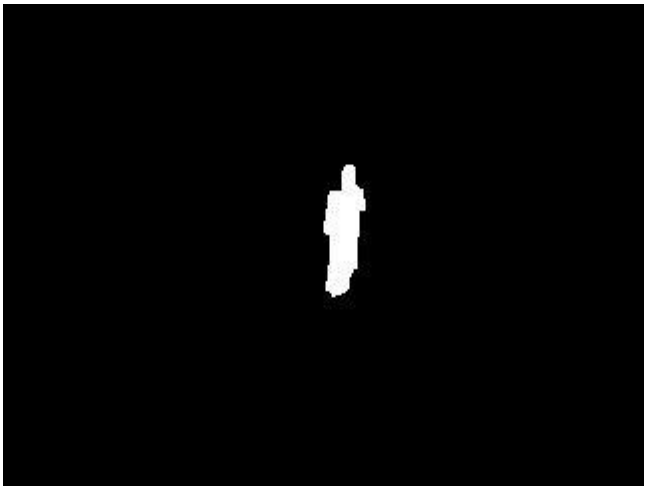
- Compter les composantes connexes sur l'image binaire, puis suppressions des composantes de tailles inférieures a un seuil.
- Appliquer une fermeture particulière: succession de 3 dilatations, suivies de 3 érosions d'un même élément structurant carré de taille 3x3.

Nous constatons, en effet, que les résultats sont bien meilleurs :

Image après traitement initial :

Fermeture basique:

Débruitage puis 3 dilatations suivies de 3 érosions (élmt. struct. de taille 3) :



Exemple sur une autre image :

Sans traitement

Après fermeture

L'idée, bien qu'intéressante au vu des premiers résultats, ne donne pas de résultats suffisants dans les conditions critiques qui nous préoccupent. La méthode ne gère pas la trop grande dégradation que produit l'érosion dans ces parties.

De plus, suite à une amélioration conséquente de la qualité des images extraites de la vidéo notamment au cours de l'évolution du seuillage, cette fermeture n'a pas été implantée.

Cependant, l'idée est restée et inspire l'application des morphomaths faite dans le programme.

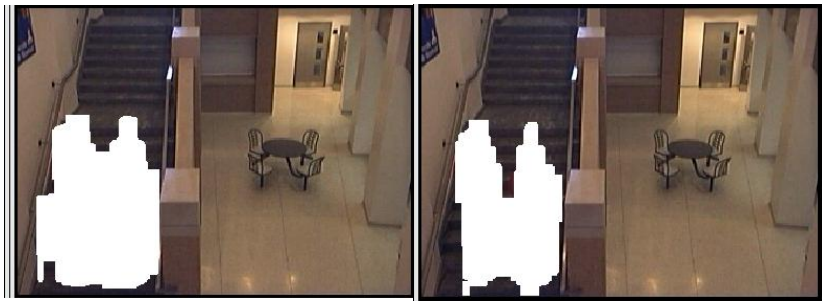
2.3.10°) Conclusion sur l'utilisation des morphomaths dans le programme

Dans la pratique, aucun traitement n'arrive à être extrêmement stable et efficace dans le temps. Cependant, une combinaison des éléments trouvés dans différentes méthodes permet d'établir le protocole utilisé dans le programme.

Le choix s'est finalement porté sur l'application de deux dilatations d'un élément structurant rectangulaire de taille (hauteur, largeur) de (9,3), suivies de deux érosions d'élément structurant carré (5,5). Cette configuration permettant un compromis entre l'efficacité d'une fermeture adaptée pour relier des composantes quasi-connexes situés l'une au-dessus de l'autre (comme c'est généralement le cas dans les conditions du projet), et la perte pas trop importante de 'patate' sur certaines images dans des conditions critiques. Perte du moins suffisamment gérable pour être détectée et surmontée par l'analyse de la trajectoire.

Ce choix évite également de fusionner à tout va, comme le fait une séquence d'une seule dilatation (18,6) suivie d'une seule érosion (10,10), ce qui fait trop perdre la notion de non-connexité de deux personnes marchant côte à côte.

Image traitée (de haut en bas) avec une 1, 2 et 3 dilatations suivies d'autant d'érosions



Différence de conservation de 'non-connexité' entre deux personnages côte à côte, entre un traitement d'une dilatation (9,3) seule (à gauche) et une dilatation (9,3) suivie d'une érosion (5,5) (à droite).

2.4°) Etiquetage

Germain Barret / Estelle Brémont

4 semaines

2.4.1°) Objectif

L'étiquetage est une étape de la chaîne de traitement permettant de distinguer des objets séparés spatialement.

On associe à chaque objet, un nombre que l'on nomme étiquette (un entier strictement supérieur à 0). Cela permet, après l'étiquetage, un accès aux pixels de l'image via cette étiquette (numéro de l'objet considéré). L'image n'est plus une simple matrice, mais une association d'étiquette à chaque objet, laissant découler l'idée de structure (au sens informatique) qui viendra ensuite.

A l'aide d'une structure, on associe à chaque composante (c'est à dire à chaque objet détecté), des informations comme sa taille, sa position, sa superficie, ou toute information utile par la suite.

L'étiquetage permet de retirer de l'ensemble des objets, ceux de taille trop petite, ceux ne correspondant pas à la forme attendue, ou ne se trouvant pas à une position réaliste. Il est également possible, à la suite de l'étiquetage (en utilisant son résultat), de supprimer les objets inutiles (bruit, porte...).

2.4.2°) Problématique

Vue de façon plus mathématique, l'étiquetage a pour but d'attribuer à chaque composante connexe du graphe (constitué par la matrice) une étiquette distincte. Une composante connexe est constituée d'un ensemble de points voisins (notion de voisinage définie ci dessous).

En théorie des graphes, on parle ainsi de composante connexe par rapport à un sommet: à partir d'un premier point objet trouvé, il est possible de chercher la composante connexe qui lui est associée, qui correspond à l'objet à trouver.

Le but de l'étiquetage est aussi et surtout d'attribuer 2 étiquettes différentes à 2 éléments de l'image espacés l'un de l'autre, donc de différencier les objets. L'implémentation doit pouvoir donner la même valeur à tous les points d'un même objet, quelle que soit leur complexité.

La complexité des algorithmes est une partie importante à prendre en compte. En effet, travaillant en temps réel, l'intervalle de temps dont on dispose pour traiter une image est inférieur à 1/25 seconde (40ms), d'autant que ce temps est déjà en partie utilisé pour d'autres traitements, et que le traitement d'étiquetage n'est pas la dernière étape.
Il serait bon d'avoir un temps de traitement fixé ou ne serait-ce qu'un encadrement fini du temps nécessaire à cette étape, pour que les traitements ne débordent pas ou peu le temps disponible sur certaines images.

En traitement d'image, la notion de voisinage n'est pas uniquement le voisinage direct (pourraient être considérés comme voisins, 2 pixels distants de moins de la distance x), néanmoins, dans notre cas, nous prendrons comme voisins les points qui se touchent.

Le critère de voisinage considéré ici est un voisinage direct, c'est-à-dire entre points qui se touchent. Même dans ce cas, on distingue 2 sortes de voisinage : la 4-connexité et la 8-connexité, distinguant les voisins qui se touchent par un bord et ceux qui se touchent aussi par un coin.
En 8-connexité les voisinages sont considérés dans toutes les directions, en 4-connexité, seuls les verticales et horizontales sont prises en compte.

Si on prend des matrices binaires (noir & blanc), chaque point a 4 (ou 8) voisins, sauf les bordures. Ce qui amène à l'idée que des effets de bord seront à prendre en compte, ou alors trouver une implémentation fixant les effets de bord.

Concernant le codage du fond et de la forme, nous avons pris pour convention de coder en binaire le fond à 0 (considéré comme noir) et la forme à 1 (considéré comme blanc).

2.4.3°) Méthodes et choix

Il faut noter que selon les algorithmes, la lecture des voisins d'un pixel peut être localisée temporellement ou répartie sur plusieurs étapes au cours de l'algorithme. C'est à dire que pour chaque pixel (x, y), ses voisins sont tous lus (souvent seule la moitié des voisins, grâce au sens de balayage) et utilisés de suite ou lus par d'autres méthodes de lecture, par exemple à plusieurs instants.

2.4.3.1°) Le choix des algorithmes

Parmi les algorithmes que nous avons testé pour ne garder que le plus adapté au problème, il y a des algorithmes gloutons, récursifs, limités à la 4-connexité ou encore sans besoin de penser aux effets de bord.

Un algorithme glouton, « l'étiquetage séquentiel itératif » (I) parcourt la matrice de point en alternant balayage avant et balayage arrière, plusieurs fois, et répand le numéro du premier point trouvé sur ses voisins. Cette méthode assure une même numérotation pour chaque composante connexe. Mais cette méthode a une complexité dépendant de la forme des objets, et surtout parcourt un très grand nombre de fois la matrice image.

Il existe un algorithme d' « étiquetage en parallèle » (II) qui répand en parallèle des étiquettes sur l'image (un peu comme un remplissage de forme), mais nécessite pour fonctionner un grand nombre de place mémoire, chaque modification des étiquettes étant stocké pour comparaison aux versions suivantes. Nous n'avons pas cherché à implémenter cette méthode aux vues de ses exigences en place mémoire.

Parmi les autres algorithmes : « l'étiquetage séquentiel avec correspondance entre points » (III). Il s'agit de parcourir la matrice de points une fois tout en mémorisant les valeurs des étiquettes du voisinage qui sont en concurrence. Ainsi, dans une deuxième phase, les étiquettes peuvent être correctement ré-attribuées.
Cela réclame une table de correspondance (c'est-à-dire un graphe, contenant autant de cliques que d'objets). Il faut ensuite le « trier » : effectuer la fermeture transitive du graphe.
Nous verrons par la suite cette notion qu'il nous a fallu étudier pour bien comprendre cette table de correspondance.
Les étiquettes sont ensuite réaffectées dans la matrice image.

Enfin, « l'algorithme en double ratissage » (IV), qui ressemble au précédent. Il s'agit de lire l'image une première fois et d'attribuer des indices d'étiquette, uniquement à l'échelle des colonnes. On ne regarde que le point précédent (tout le voisinage n'est pas pris en compte ici). Cette étape est donc extrêmement rapide.
Puis, dans une seconde phase (et comme dans l'algorithme précédent), on lit l'image dans une direction orthogonale à la précédente pour y repérer les problèmes d'étiquetage et construire un graphe des correspondances.
Une étape de fermeture transitive est nécessaire ici.
Les étiquettes sont ensuite réaffectées dans la matrice.

La comparaison des algorithmes s'est faite grandeur nature, en les implémentant et en les comparant surtout au niveau du temps d'exécution. Nous avons implémenté l'étiquetage séquentiel itératif (I), l'étiquetage séquentiel avec correspondance entre points (III) et l'étiquetage en double ratissage (IV).

Le dernier semblait donner des résultats plus satisfaisant. Il est facile de le comprendre après coup. Cet algorithme est en 4-connexité (tout son principe est basé dessus), ce qui est déjà un avantage en vitesse et ne dénature pas l'image. Il ne lit (ou écrit) que 3 fois la matrice image. Son seul inconvénient, mais que son concurrent possède aussi est de devoir « simplifier » une table de correspondance entre points. Et plus cette table est compliquée (beaucoup d'étiquettes lors de la première passe), plus c'est long. D'autant que les algorithmes de fermeture transitive sont en $O(n^3)$.
Pour comparaison, on trouve environ 20 étiquettes pour (III) et moins de 100 pour (IV).
Mais l'inconvénient en vitesse perdu ici est apparemment gagné sur la lecture de la matrice.

On stocke le résultat directement dans la matrice image (sans matrice intermédiaire).

2.4.3.2°) Simplification de la table de correspondance

Nous avons dit que la simplification est un problème de fermeture transitive de graphe. Ce n'est pas tout à fait exact. En effet, la fermeture transitive s'applique à un graphe orienté.
Les algorithmes de fermeture transitive sont en $O(n^3)$ [voir Roy-Warshall] et au mieux peuvent être simplifiés en $O(n^2 \log(n))$.
Le notre ne l'est pas. Mais le problème à résoudre est pratiquement le même.

Il s'agit de construire un graphe étoilé autour d'une des valeurs de la composante connexe. Notre structure n'est pas orientée, au sens de graphe orienté, mais les couples (x, y) ont été stockés dans l'ordre décroissant (x>y). Ajoutant une orientation interne aux données. Cela nous aide à minimiser la valeur des étiquettes. Nous sommes sûr qu'un couple (x, y) est correcte si x est le min de toutes la composante connexe à laquelle elle appartient.
Le problème à résoudre est finalement à nouveau de trouver une composante connexe au sein de cette table.

2.4.3.3°) Structure

Pour stocker notre graphe, nous avons testé plusieurs structures (tableau triangulaire, liste de couples). Et avons opté pour la liste de couple, car elle prend moins d'espace mémoire, est facilement compréhensible à l'œil et ne perd pas en efficacité, toujours au vu des temps d'exécution.
Pour le couple (A, B) il faut lire B est à remplacer par A.

Pseudo code de simplification de la table de correspondance

Pour chaque couple (A, B) de la table de correspondance, on le compare aux suivants :
Pour les couples suivant (I, J)

Si on trouve (I, A)
Alors on le remplace par (I, B)

Si on trouve (A, J)
Alors on le remplace par (J, B)

La première étape vise à répandre la bonne étiquette dans tous les couples qu'on sait être voisins de A, les voisins de voisins étant traités lors des passages suivants (récursivité du Pour).

La deuxième étape est un cas de détection de cycle, et rétablie la bonne étiquette de l'autre côté de la jonction.

2.4.4°) Optimisation

Un compromis doit être trouvé entre un passage rapide dans la matrice image et un traitement rapide de la table de correspondance. Nous avons apporté une optimisation en limitant l'étiquetage à une zone plus petite que toute l'image. Cette zone est trouvée lors de l'étape de débruitage, elle est de superficie 5 à 10 fois plus petite que la matrice image. Le traitement est beaucoup plus rapide. Néanmoins, la table de correspondance garde la même taille (puisqu'elle est identique à celle que l'on aurai obtenu en parcourant toute l'image).

Il est important de pouvoir lire directement la table de correspondance sans avoir à faire de post-traitement pour connaître les valeurs de remplacement des étiquettes. Donc la table de correspondance est un point crucial de l'algorithme puisqu'elle contient toute l'information d'étiquetage et de voisinage. De nombreuses erreurs peuvent venir de là.

On se rend compte que le coût de notre simplification de table de correspondance est en O(n log n), ce qui est inférieur aux algorithmes de fermeture transitive, car on utilise l'organisation de la table pour faire moins de calculs, les difficultés sont réduites.

Exemple d'étiquetage

2.5°) Reconnaissance et suivi d'objet

Guillaume Lambert / Estelle Brémont

1 semaine (dense)

2.5.1°) Création d'une liste des objets détectés

En sortie de l'étiquetage, la reconnaissance récupère les paramètres de chacune des composantes connexes et les stocke dans une liste. Cette liste, définie par une suite de structures contenant donc: une étiquette, un nombre de pixels appartenant à l'objet, la somme des valeurs rouges des couleurs de chaque pixel appartenant à l'objet, la sommes des valeurs vertes, la somme des valeurs bleues ainsi que les coordonnées des deux points extrêmes du rectangle englobant l'objet.

2.5.2°) Reconnaissance d'objet

La reconnaissance d'objet consiste à identifier d'une image sur l'autre et à assimiler les objets contenus dans la liste précédente, en fonction d'une mémorisation des objets détectés dans l'image précédente. Ainsi, soit l'objet est reconnu comme déjà existant, et alors mis à jour (couleur, trajectoire), soit, il est détecté comme étant un nouvel objet et alors placé en mémorisation.

Le paramètre utilisé, afin de reconnaître les objets entre eux, est la moyenne des couleurs des pixels considérés comme appartenant à l'objet détecté. Ce paramètre a l'avantage d'être indépendant de la trajectoire, et donc de pouvoir suivre l'objet dans n'importe quelle direction. De plus, ce paramètre peut raisonnablement varier, mais pas autant qu'un contour qui se déforme, ou subir toute une lourde batterie de prédiction comme une position.

En sortie, la reconnaissance et le suivi d'objet transmet la trajectoire de chaque objet réactualisée vers une analyse de normalité. Plus précisément, le centre du rectangle englobant de l'objet est utilisé comme point de localisation de l'objet dans l'image.

2.5.3°) Des problèmes persistants

La méthode de reconnaissance et de suivi d'objet est globalement satisfaisante, tout en limitant le nombre de singularités, mais il reste cependant deux problèmes difficiles à traiter avec la seule moyenne des couleurs des pixels concernés:

- 2 objets dont un se coupe en deux, donnant ainsi trois objets. Ce cas peut parfois être traité en comparant les quantités de points concernés par chacun des objets, et en remarquant qu'un objet de grande taille à disparu et que deux de petite taille sont apparus. Néanmoins, ce cas arrive surtout lorsqu'il y a deux personnages proches, généralement côte à côte, et l'une des deux parties fractionnées fusionne avec le deuxième objet resté intact. Ce cas est à contre-cœur finalement considéré comme étant impossible à traiter.

Une solution pour le résoudre serait un étiquetage qui garde en mémoire les étiquettes attribuées précédemment, et, en considérant qu'un objet ne se déplace que raisonnablement lentement, l'étiquetage remettrait l'ancienne étiquette aux points qu'il doit traiter se trouvant à la place de points déjà étiquetés précédemment. Cependant, il est toujours possible de trouver un cas où cela ne fonctionnerait encore pas (deux personnes qui courent côte à côte), et il faut savoir considérer des cas intraitables dans des délais restreints.

- 2 personnes de même taille, même tenue, même couleur de cheveux, même vitesse, etc, qui se croisent (sans fusionner). Malheureusement le hasard doit alors intervenir. Un contre-exemple pourrait facilement être trouvé dans des solutions intuitives (prédiction de trajectoire). En cas de fusion, on traiterait un nouvel objet constitué des deux, mais en absence de fusion et de mois supplémentaires d'étude, il faut bien reconnaître l'impossibilité du traitement d'un cas aussi particulier.

2.6°) Analyse de l'anormalité

Emeric Golfier
5 semaines

2.6.1°) Etude des vidéos

Une étude des vidéos données en sources permet de recenser les anormalités et de les regrouper en quelques Types d'anormalités :

I - Un personnage se retrouve 'dans un mur' ou 'sur un mur'.
II - Un personnage tombe. Il se met à marcher plus lentement, s'effondre relativement

lentement puis s'immobilise.
III - Mouvement répétitif. De divers types (avant arrière, circulaire, etc)
IV - Course.

Plus précisément:

Anormale1_1:	Mouvement Aller-retour devant la porte.	Type III
Anormale1_2:	Course (1 seul aller-retour)	Type IV
Anormale1_3:	Course	Type IV
Anormale1_4:	Affalement sur la table	Type II
Anormale1_5:	Chute dans l'escalier	Type II
Anormale1_6:	S'assoit 'dans le mur'	Type I
Anormale1_7:	Tombe au sol	Type II
Anormale1_8:	Mouvement circulaire répétitif	Type III
Anormale1_9:	Rase un mur	Type I
Anormale1_10:	Monte sur un mur	Type I
Anormale1_11:	S'assoit 'dans le mur'	Type I
Anormale1_12:	Mouvement répétitif avant-arrière	Type III
Anormale1_13:	La porte est fermée... il courre un peu, avant et après.	Type III
Anormale1_14:	Mouvement répétitif avant-arrière. La personne n'est pas toujours en entier.	Type III
Anormale1_15:	S'effondre sur la table, lentement.	Type II
Anormale1_16:	Tombe dans l'escalier	Type II
Anormale1_17:	Tombe par terre	Type II
Anormale2_1:	Les deux personnes font un mouvement circulaire répétitif	Type III
Anormale2_2:	Les deux personnes font un mouvement gauche-droite répétitif (en se tenant les bras et sans se tenir les bras)	Type III
Anormale2_3:	'Bagarre' entre deux personnes. Mouvements chaotiques qui peuvent être interprétés comme un mouvement répétitif.	Type III
Anormale2_4:	'Bagarre' entre deux personnes. Mouvements chaotiques.	Type III

2.6.2°) Une analyse sous contraintes

La détection d'anormalité doit résister à diverses conditions, qui pourraient être repérées comme de l'anormalité :

- On perd une personne momentanément, mais on la retrouve 'plus loin'.
- Une personne reste sur place (lecture, discussion avec une autre personne).
- Les personnes se chevauchent plus ou moins partiellement, se tiennent par le bras (se serrent la main etc): fusions de durées non déterminable.
- Marche soutenue et non course.
- Personne ralentit et s'arrête (pour discuter avec une autre).
- Un personnage peut s'asseoir à une chaise, mais pas sur le rebord d'un mur.
- etc

De plus, l'algorithme doit permettre, à partir d'une trajectoire (liste de points), de détecter les quatre types d'anormalités décrits précédemment sous des contraintes matérielles supplémentaires découvertes par une étude des éléments d'entrée:

- Une trajectoire trop chaotique : au lieu d'une trajectoire rectiligne, on a une trajectoire 'en accordéon', due à la variation de la position du point servant de référence. Ce point est en effet sensible à la moindre déformation des 'patates' soit lors des traitements précédents, soit lors des 'petits mouvements internes' de la patate concernée (des bras qui bougent le long du corps, des jambes qui bougent, une tête qui s'incline, etc).

- Un fonctionnement pour chaque patate qui fonctionne sans être perturbé (et même sans en tenir compte du tout) par le nombre de personnages repérés dans la vidéo.

2.6.3°) Approche du traitement des types de normalités et gestion des contraintes

2.6.3.1°) Approche des contraintes physiques :

Chaque objet a sa propre trajectoire consistant en un tableau des 10 dernières positions de l'objet. Ainsi, l'analyse est bien liée uniquement à une trajectoire et en aucune façon à un objet. De plus, chaque objet étant défini par sa couleur, c'est à dire indépendamment de sa trajectoire, il ne peut y avoir de problème de croisement entre deux personnages. En cas de fusion, les deux objets sont remplacés par un plus grand, donc une nouvelle trajectoire est analysée sans se soucier du type d'objet. De plus, seules dix valeurs sont utilisées : il s'agira d'une analyse de l'allure de la trajectoire en cours. Outre quelques variables de mémorisation d'inertie, cette analyse ne dépend pas du passé de la trajectoire, plus loin que ces 10 points conservés.

La trajectoire chaotique s'explique par la faible précision de localisation du centre du rectangle englobant. Pour compenser cette frénésie, l'algorithme ne va traiter qu'une paire de coordonnées toutes les cinq images. Ainsi, l'allure de la trajectoire devient beaucoup plus rectiligne, et les traitements sur la trajectoire alors définie plus précis. Par contre, cela entraîne qu'une détection d'anormalité de type 'course', met plus de temps à être détecté (50 images en tout, soit environ deux secondes en temps réel, au lieu de 10). Cependant, le choix de 5 images nous y fait gagner beaucoup plus à travailler sur une trajectoire rectiligne par rapport à la perte du traitement de moins d'image (qui se traduit par un retard temporel raisonnable).

2.6.3.2°) Définition d'une alerte

Les signaux d'alerte lancés par le programme sont décidés à partir de l'analyse de 2 paramètres :

- Un signal de pré-alerte indiquant qu'il se passe quelque chose que l'on pourrait qualifier d'anormal. Autrement dit: la trajectoire est entrée dans l'un des 4 types de trajectoires anormales (dans un mur, tombe, mouvement répétitif et course).
- Le lieu d'où est lancé cette pré-alerte.

L'algorithme fonctionne donc en deux étapes: la recherche de condition(s) d'alerte, puis le traitement de l'alerte en fonction du lieu où elle se déclencherait.

2.6.3.3°) Définition d'une pré-alerte

Il existe 6 types de pré-alerte :

- 'MUR', qui signifie que le personnage se trouve dans un lieu où personne ne devrait se trouver. Par exemple: une personne s'assoit sur le rebord de l'escalier, longe un mur, s'assoit sur un rebord contre le mur devant le store, etc. Ce signal d'alerte est particulièrement facile à détecter: c'est un simple test d'inclusion du point de localisation du personnage à une zone. Pour simplifier la méthode de recherche d'inclusion, les quelques murs définis l'ont été par des quadrilatères. Celle-ci s'effectue par une étude du signe du produit vectoriel entre le point de localisation du personnage et deux points consécutifs de l'un des quadrilatères. En effet, ce signe permet de détecter si le 'virage' au point commun des deux vecteurs définis par les trois points et utilisés dans le produit vectoriel est 'vers la droite', 'vers la gauche' ou si les points sont alignés (le produit est égal à 0). Or, si les quadrilatères sont convexes et orientés, que l'on prend à chaque fois deux points consécutifs dans le sens de cette orientation, et comme ces quadrilatères ont toujours un, voire deux côtés inaccessible(s) au personnage

(parce que bord cadre), on peut définir que si les 4 virages sont vers le même sens, alors le point est inclus dans le quadrilatère, sans quoi il ne l'est pas.
Pouvant s'effectuer avec un seul point de localisation, et étant donné que la réponse est rapide et sans appel (dans une zone interdite = message d'alerte) ce traitement est effectué en premier, et en cas d'affirmative, un message est lancé et l'algorithme ne va pas plus loin.
MUR est un signal de pré-alerte particulier puisqu'il s'agit aussi d'un lieu.

- 'COURSE': signifie que le personnage a une somme de ses 9 dernières vitesses (calculées chacune avec deux points (distance euclidienne) et la somme totale avec les 10 points du tableau) qui dépasse un seuil déclenchant l'alarme.

- 'IMMOBILITE' qui détermine, à l'aide d'un 'carré englobant', si le personnage reste dans une zone. La précision de la détermination de l'immobilité dépend donc du pas de sélection d'images (fixé à 5) et de la taille du carré englobant. Si le point reste dans le carré à la position suivante, un compteur s'incrémente, sinon le carré suit le point qui en devient le centre. L'immobilité se déclenche donc par seuillage du-dit compteur. L'immobilité n'est pas une pré-alarme en tant que telle, mais juste un signalement d'une situation du personnage. Après, les combinaisons entre les signaux de pré-alerte et les lieux de déclenchement déterminent s'il y a lieu de lancer une alerte ou non. Le tableau suivant montre le fonctionnement des combinaisons entraînant le déclenchement d'alertes, de pré-alertes, de messages d'avertissement ou de non lancement d'alerte.

- 'STAGNATION' est issu de l'IMMOBILITE. Un personnage trop immobile, si ce n'est pas dans un lieu approprié (chaises pour la lecture), sera considéré comme stagnant (système de seuillage). Un personnage considéré comme trop stagnant (système de seuillage), se verra lancer une alerte (sauf lieu autorisant la stagnation).

- 'MVT-REPET': signifie que le personnage effectue un mouvement répétitif, de type vertical, horizontal, en boucle, en huit, en carré, en croix et dans n'importe quel sens. Un système de compteur avec une incrémentation permettant de conserver une inertie du déplacement détermine si le personnage tourne trop à droite, un autre détermine s'il tourne trop à gauche. En effet, tout mouvement répétitif est interprété comme une trajectoire qui tourne 'excessivement'.

La notion d'inertie conservée du mouvement signifie que si l'on détecte une aptitude à tourner dans une direction (par exemple vers la droite), cette attitude doit diminuer si la personne se met à tourner vers la gauche. Pour cela, il faut que des directions opposées fassent diminuer les compteurs opposés. Cependant, un simple +1/-1 (+1 en incrémentation et –1 en décrémentation) ferait que les compteurs s'auto-annuleraient en cas de mouvement en forme de 8.
Pour garder la notion d'inertie, on fait donc un +2/-1. Ainsi, un mouvement vers la gauche permet d'annuler la moitié de la 'trace' laissée par le dernier mouvement vers la droite. Une inertie du mouvement est conservée et permet ensuite, avec un seuil, de déterminer si cette inertie est trop importante et donc s'il y a mouvement récurrent, soit une Anormalité de Type III.
Une étude plus approfondie pourra permettre de définir un seuil plus efficace, ou un pas d'incrémentation avec une inertie plus ou moins forte et donc plus efficace (par exemple : +3/-2, +5/-4, ou avec une inertie plus forte : +3/-1 etc). Pour le cas qui nous intéresse, un seuillage de type +2/-1 est suffisant.

- 'REPOS' signale qu'il n'y a aucune anormalité détectée.

2.6.3.4°) Définition d'un lieu

Certaines zones ont montré des singularités propres quant aux valeur de seuillage. Ces même zones posent aussi problème avec les variations de la trajectoire. 6 zones ont été définies comme 'particulières':

- ‘MUR’, défini dans la partie précédente. La zone ‘MUR’ est définie par 4 quadrilatères dans le programmes (MUR1 à MUR4).
- ‘PORTE’, définit une zone concernant la porte.
- ‘TABLE’, définit le centre de la table.
- ‘CHAISE’, définit une zone autour de la table où sont situées les chaises.
- ‘ESCALIER’, définit la zone de l’escalier.
- ‘REPOS’, signifie que le personnage n’est pas dans un des lieux pré-cités.

Ces lieux ont des noms suffisamment explicites, mais correspondent à des zones qui ne ‘collent’ pas forcément à la position réelle des objets que leur nom mentionne. En fait, ces lieux définissent les zones où le centre de rectangle englobant se situe lorsque l’on peut considérer le personnage comme étant dans un mur, assis sur une chaise, ouvre une porte, etc.

2.6.3.5°) Relations ALERTES-LIEUX

L’envoi ou non de signal est déterminé en fonction des relations en les lieux de déclenchement et le type de pré-alerte détecté. Ces relations sont définies par le tableau :

	COURSE	MVT_REPET	IMMOBILITE	STAGNATION	REPOS
MUR	X* **	X*	X*	X*	X*
PORTE	X**	Le personnage FORCE PORTE	Si le personnage force la porte, envoyer le signal FORCE PORTE. S’il bloque la porte, envoyer BLOQUE PORTE. Sinon, juste signal ATTENTION PORTE	Si le personnage force la porte, envoyer FORCE PORTE, sinon envoyer BLOQUE PORTE	Pas d’alarme
TABLE	X**	Si le personnage lit, alors on n’envoie pas de signal. Sinon, si le personnage est déjà tombé sur la table, on renvoie le message TOMBE SUR TABLE. Sinon, on renvoie le signal PERSONNAGE SUR TABLE	Si le personnage est sur la table, on envoie PERSONNAGE SUR TABLE. S’il est tombé sur la table, on envoie PERSONNAGE TOMBE SUR TABLE. Sinon, on prévient ATTENTION TABLE	Si le personnage ne lit pas, envoyer PERSONNAGE TOMBE SUR TABLE. Sinon, il lit.	Pas d’alarme

CHAISE	X**	Sauf si le personnage est déjà en alerte (et alors on renvoie la même), le personnage lit, et donc pas d'alerte à envoyer.	Sauf si le personnage est déjà en alerte (tombé ou en bagarre ou mvt anormal) et alors on renvoie la même, le personnage lit, et donc pas d'alerte à envoyer.	Sauf si le personnage est déjà en alerte (et alors on renvoie la même), le personnage lit, et donc pas d'alerte à envoyer. (facultatif, cas jamais rencontré)	Pas d'alarme
ESCALIER	X**	Si le personnage est déjà tombé, alors on renvoie le signal PERSONNAGE TOMBE DANS ESCALIER. Sinon, on envoie le signal PERSONNAGE SE BAGARE OU MVT ANORMAL DANS L'ESCALIER.	Si le personnage se bagarre ou fait un mouvement anormal, on renvoie BAGARE OU MVT ANORMAL DANS ESCALIER, s'il est déjà tombé, on envoie PERSONNAGE TOMBE DANS ESCALIER	Le PERSONNAGE TOMBE DANS ESCALIER.	Pas d'alarme
REPOS	X**	Si le personnage ne lit pas, alors si le personnage est déjà tombé, on envoie le signal PERSONNAGE TOMBE. Sinon, à moins que le personnage ne soit en train de forcer la porte, on renvoie PERSONNAGE SE BAGARRE OU MVT ANORMAL. Si le personnage lit, on ne lance pas de signal. Si rien de tout	Si le personnage ne lit pas et qu'il se bagarre ou fait un mouvement anormal, on envoie PERSONNAGE BAGARRE OU MVT ANORMAL. S'il était en train de lire, il ne lit plus.	Si le personnage n'est pas dans les chaises et qu'il lisait (il discute) ne rien faire, s'il sortait d'un lieu ne rien faire, sinon le PERSONNAGE TOMBE Sinon (il est dans les chaises), le PERSONNAGE LIT	Pas d'alarme

		cela, le personnage se BAGARRE OU FAIT UN MVT ANORMAL			

* : quand on détecte le lieu, on lance l'alarme PERSONNAGE_DANS_MUR et on stoppe l'algorithme.
** : quand on détecte l'alarme PERSONNAGE_COURRE, on la lance et on stoppe l'algorithme.

Ainsi, le MESSAGE_ALERTE_DE_SORTIE peut être défini par plusieurs signaux :
- ALERTE_NON_IDENTIFIEE
- PERSONNAGE_DANS_MUR
- PERSONNAGE_COURRE
- PERSONNAGE_TOMBE
- PERSONNAGE_BAGARE_OU_MVT_ANORMAL
- PERSONNAGE_BLOQUE_PORTE
- PERSONNAGE_FORCE_PORTE
- PERSONNAGE_SUR_TABLE
- PERSONNAGE_TOMBE_SUR_TABLE
- PERSONNAGE_TOMBE_DANS_ESCALIER
- PERSONNAGE_BAGARE_OU_MVT_ANORMAL_DANS_ESCALIER

Ainsi que quelques messages, qui ne sont pas d'alerte, mais juste qui en annonce un et sont donc facultatifs:
- ATTENTION_TABLE
- ATTENTION_PORTE
- ATTENTION_ESCALIER

2.6.3.6°) Problèmes émergeants

Certaines contraintes n'étaient pas flagrantes dans une première analyse de la problématique. La pratique les mettent à jour :

- Plus les personnages vont loin dans la pièce plus ils 'marchent lentement'. C'est un simple effet optique dû au point de vue de la caméra et au déplacement des personnages vers les points de fuite de l'image. Cet effet est négligé dans la partie droite de l'image grâce au choix judicieux du seuil de course (marche trop rapide) et la taille du carré-englobant (marche trop lente). Par contre, le problème est plus manifeste de l'autre côté de l'image. En effet:

- Les personnages marchent plus vite dans l'escalier qu'ailleurs, pour une raison de point de vue de la caméra. L'escalier monte et la caméra est placée en hauteur. Ainsi, si les personnages s'éloignent dans la pièce et semblent alors marcher plus lentement, ils se 'rapprochent' en montant l'escalier. De plus, un palier dans l'escalier donne une allure de ralentissement à la trajectoire. Ceci n'est dû qu'à la position de la caméra et peut entraîner une mauvaise interprétation de la trajectoire.

- Le système de seuillage des mouvements anormaux est lié au carré englobant et, plus un personnage est loin, plus son déplacement nous paraît 'petit' et plus il reste dans le carré. Par contre, plus il est proche plus son déplacement aura tendance à sortir du carré. Ainsi, ce système de seuillage devrait dans l'absolu être adaptatif à la position des personnages dans la pièce (par zonage (escalier, pas escalier) et par localisation (en bas de l'image = plus près)).

Ces problèmes apparaissent donc avec une étude plus attentive du fonctionnement de l'algorithme, et ne peuvent être traités qu'a posteriori (ou à moins d'être très fort !). Ils auraient pu être traités avec plus de temps de programmation.

3°) OPTIMISATION DU CODE

Estelle Brémont

L'utilisation d'un programme basé sur une lecture vidéo (et donc, approximativement quasiment en temps réel) a impliqué la nécessité d'un certain nombre d'optimisations afin de garder de la fluidité malgré les traitements appliqués entre chaque frame.
Cela a permis d'avoir un code ayant un minimum de calculs inutiles, et aussi a obligé une plus grande rigueur.
De plus, il ne faut pas perdre de vue que le fait de rassembler et de « merger » des programmes faits plus ou moins indépendamment, même s'ils ont été encadrés, peut impliquer des problèmes, tant algorithmiques qu'au niveau de la complexité. Certaines améliorations ont parfois été une obligation pour le bon respect des objectifs fixés.

Les principales optimisations ont été :

- l'anti-parasite, non seulement efface les parasites, mais indique par la modification d'une variable globale si en sortie, l'image comporte des éléments en mouvement ou non. L'ensemble des traitements suivants ne sera lancé que si nécessaire.
- l'ensemble des traitements (morphologie mathématique, étiquetage, suppression des petites composantes connexes) ne sont pas lancés sur la totalité de la matrice, mais uniquement sur le rectangle englobant la partie détectée et reconnue comme « en mouvement ».
- Optimisation du traitement d'étiquetage (un des traitement les plus lourds de tout le programme !) en essayant de limiter par des tests la taille des structures manipulées (en particulier la table de correspondance). En effet, même si à l'origine, cet algorithme est linéaire, la gestion de la table de correspondance, elle, peut s'avérer une catastrophe au niveau complexité, et résolution temporelle.

Les principales améliorations ont touché les ajouts d'éléments à cette table :

- On contrôle et on empêche tout doublon lors de l'insertion d'élément (utile pour limiter le temps de parcours pour la mise à jour de la matrice par rapport à la table)
- Pour la suppression des petites composantes connexes, on s'assure que l'on ne travaille que pour les éléments de la matrice non nuls, c'est-à-dire des points appartenant à un objet
- On alloue en global, et on désalloue en fin de processus, ce qui évite des allocations conséquentes et successives trop fréquentes.

Ainsi, grâce à ces quelques optimisations au niveau du traitement de l'étiquetage, on peut considérer que la taille en mémoire (stockage) a été approximativement divisée par 4. Ce qui allège nettement l'exécution.

4°) VERS UNE AMELIORATION DES ALGORITHMES / AUTRES APPROCHES

Germain Barret / Yan Zhang

Plusieurs notions ont été évoquées, qui n'ont, faute de temps pu être développées.

La prédiction des variables significatives

Pour choisir les bonnes variables, on garde en vue l'objectif à atteindre : discriminer les cas normaux et anormaux.

Parmi les variables permettant d'atteindre cet objectif, la principale variable est la position du personnage, c'est à dire, à notre niveau, la position du cadre encadrant la forme (le blob), mais également la position de son centre, ou de son barycentre.

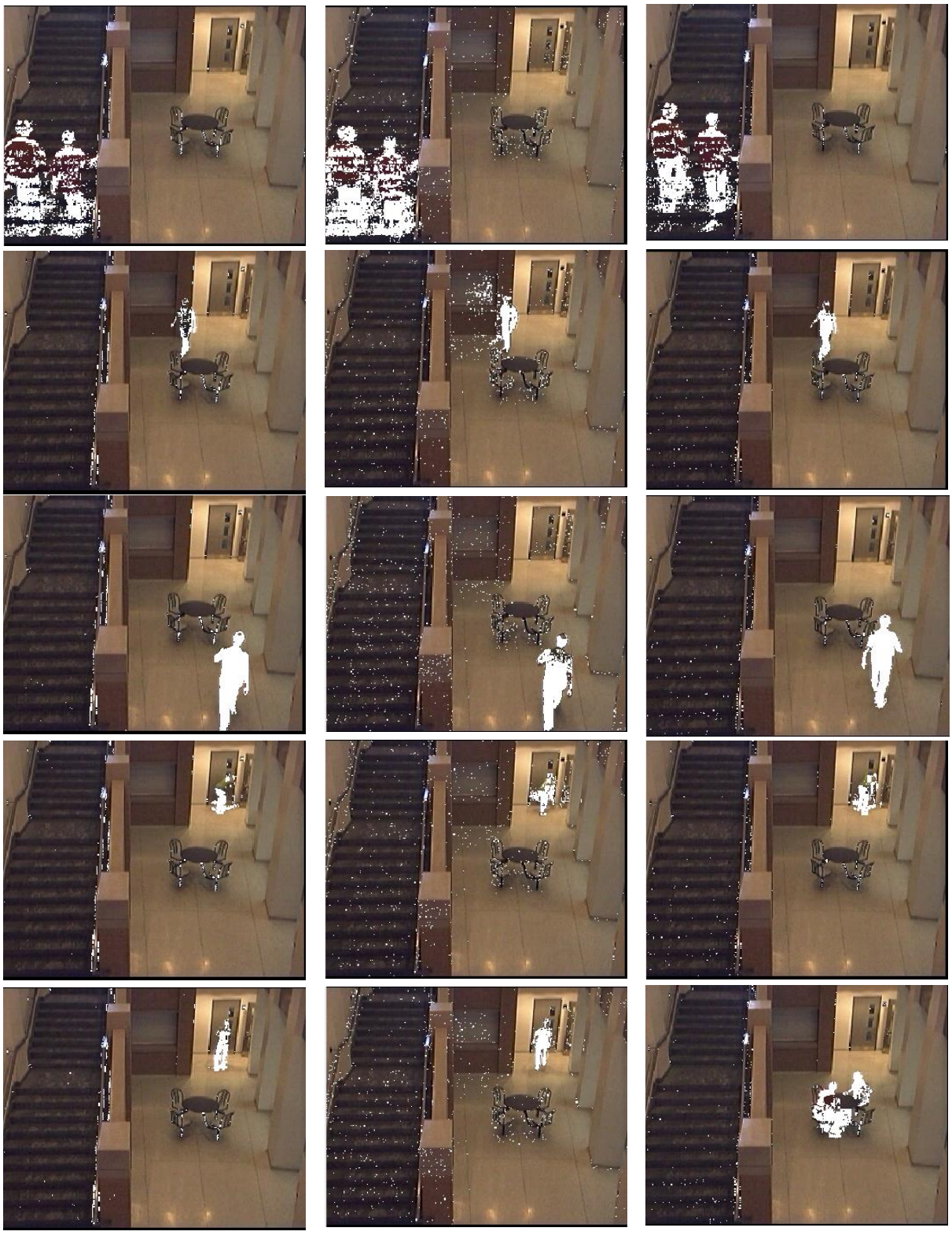
La prévision consiste à trouver les valeurs futures de ces variables à un instant quelconque, c'est à dire soit pour l'image en cours (pour mieux retrouver les personnes) ou l'image suivante, voire même bien plus tard (pour retrouver la personne qui passe derrière un obstacle).

En utilisant ces valeurs pour détecter la normalité, il faut rester tolérant face à l'erreur de prédiction, en se fixant des marges (seuils), telles que un mouvement normal reste dans les bornes et qu'un mouvement extra-ordinaire soit hors bornes.

Ou inversement, un mouvement anormal peut être une variable qui devient constante, qui s’annule ou prend certains types de valeurs facilement identifiable.

Annexe 1 : Comparaison des seuillages de base

SEUIL 1 SEUIL 2 SEUIL 3



Annexe 2: Etiquetage

Un algorithme de fermeture transitive est l'algorithme de Roy-Warshall

On construit la fermeture transitive A en effectuant sur l'ensemble des sommets s le traitement $F_s(A)$ qui consiste à ajouter les arcs (y, z) tels que (y,x) et (x, z) existent.

L'algorithme est donc le suivant : Fermeture-Transitive(G)

1. Pour tout s faire
2. Pour tout s'
   Si (s', s) existe alors
   Pour tout s''
   si (s, s'') existe alors ajouter l'arc (s', s'')

La complexité de cet algorithme est en $O(n^3)$