MASTER INFORMATIQUE
Université de Bordeaux I
2006-2007

Projet de Programmation
Rapport Final

Détection temps-réel de visages en mouvement dans une séquence vidéo : Application à la vidéosurveillance

Christophe GACHINIARD
Nicolas AUCOUTURIER
Antoine LAMBERT
Elric DELORD

Client : M. NICOLAS
Chargé de TD : M. DESBARATS

Résumé du Projet

Dans l'optique de sécuriser certaines salles du LaBRI ainsi que le matériel présent, nous réaliserons un système de vidéosurveillance par ordinateur. Le but principal étant de détecter, à partir de caméra(s) numérique(s) installée(s) dans les salles en question, les personnes pénétrant dans ces salles, et plus particulièrement leurs visages, et ensuite d'informer de l'intrusion. Pour ce faire, nous appuierons nos travaux sur la Thèse de doctorat de L. Carminati (Université de Bordeaux 1, mars 2006) [1], ainsi que sur le logiciel de détection de visage qu'il a mis en place.

A l'heure actuelle, le programme implémenté par L. Carminati fonctionne, chaque visage détecté est marqué par un rectangle. En outre, celui-ci ne permet pas une détection d'un visage dans toutes circonstances, c'est à dire le cas où la luminosité est faible et les cas concernant la position ainsi que les déplacements de ce visage.

Notre travail consistera à améliorer ce programme et faire en sorte que la détection de visages soit plus efficace, dans les limites fixées par le matériel. Ensuite, l'ajout de fonctionnalités tel que la notification par courriel qu'un visage a été détecté, ou d'une information quand à la présence d'une personne dans la salle.

Dans le cadre de ce projet, nous devrons tenir compte de la législation concernant la vidéosurveillance, et plus particulièrement des droits à l'image des personnes filmées.

Abstract

In order to secure some rooms in LaBRI, especially for preventing high-tech equipment's robbery, we will implement a video monitoring software with face detection functionality. The main goal is to detect people, and more precisely faces, in the video streams coming from the cameras installed in the rooms. Of course, when an intrusion is detected, the software will have to inform its users.
To do these tasks, we will base our work on L. Carminati's thesis (Bordeaux 1 University, March 2006) [1], as well as on the face detection software he has implemented.
At this time, the software of L. Carminati is working pretty well. A lot of faces are detected and marked by a rectangle. Moreover, he is not able to detect all the faces in a video stream, especially when the luminosity is low, when a face is in profile or when a face doesn't move.
Our work will consist of improving L. Carminati's software and to make the face detection more efficient. Then, we will add new functionnalities like the capacity of warning users that a face has been detected. Eventually, we will make the software deployment easier by adding some management's functionalities.
Within the framework of this project, we will have to take care of the legislation regarding video surveillance, and more precisely the picture rights of the people filmed.

Table des matières

Table des figures

Chapitre 1

Introduction au domaine d'application

1.1 Vidéosurveillance

La vidéosurveillance est un «procédé de surveillance à distance qui met en oeuvre un système de télévision en circuit fermé » [1]. Bien entendu, cette définition a évolué, et ne se réduit pas aujourd'hui à un système de télévision (cf. Technologie Actuelle 1.2) Elle consiste à placer des caméras de surveillance dans les lieux publics ou privés pour prévenir des vols ou des agressions, ou encore servir de système de dissuasion. En plein essor depuis les années 1990, les caméras font leur apparition dans de nombreuses villes européennes. Aujourd'hui, ces caméras sont utilisées dans les transports publics (très pratique dans les réseaux de transports sans chauffeur), les autoroutes, les parkings et certains établissements sécurisés (supermarchés, casinos, ...).

1.1.1 Arrivée de la vidéosurveillance

Dans les années 1970 et 1980, les premières expériences sont réalisées au Royaume-Uni. Peu de temps après, il s'est révélé que les vidéos enregistrées ont servies de preuves irréfutables dans plusieurs grands procès au début des années 1990. Ces succès conduisirent le gouvernement à lancer une série d'installations de caméras. Aujourd'hui, les caméras installées au Royaume-Uni en font le pays le plus surveillé au monde. Les chiffres approximatifs révèlent que plus de 400 000 caméras sont actuellement installées à Londres, sur un total de 4 000 000 dans tout le Royaume-Uni. Aujourd'hui, on estime qu'un londonien est filmé en moyenne 300 fois par jour.

La France aussi s'intéresse aux systèmes de vidéosurveillance. En 1998, le nombre de caméras en France était estimé à un million dont 150 000 dans le domaine public. Ces installations commencent aussi à fleurir dans les villes françaises. Cependant, il existe aussi des associations qui militent contre cette forme de surveillance, la plus connue étant l'association «Souriez, vous êtes filmés»[2], ce qui gèle le déploiement de ces systèmes. Certaines personnes s'inquiètent de l'ampleur que prend le phénomène de vidéosurveillance. Elles y voient à l'avenir une surveillance de chaque faits et gestes des gens, faisant ainsi référence à "Big Brother" d'où le nom donné à la vidéo surveillance dans les rues en Angleterre[3].

[1] Petit Larousse Illustré 2002 (page 1065)
[2] http ://souriez.info/
[3] roman de 1984 publié en 1949 écrit par George Orwell

1.1.2 Constats

Les caméras de surveillance ne suppriment pas les délits ou les crimes mais elles aident à mieux les détecter et à poursuivre les auteurs. Les britanniques affirment que la vidéosurveillance réduit cela. Cependant aucune étude n'a pu montrer l'impact de cette technique. Néanmoins, après les attentats dans le métro de Londres du 7 juillet 2005, les enregistrements des caméras de surveillance ont été utilisés pour identifier les poseurs de bombes, fait qui a considérablement boosté le phénomène de vidéosurveillance en Angleterre.

Par contre, tous les pays n'en sont pas au même stade de déploiement de ces outils de surveillance. Par exemple, l'Angleterre a déployé ces caméras sans se préoccuper en premier lieu des lois et des principes éthiques associés au sujet. Alors que d'autres pays comme la Nouvelle-Zélande se sont d'abord penchés sur les lois concernant l'installation d'un tel dispositif ainsi que les droits des personnes touchées par la mise en place de surveillance électronique.

1.2 Technologie actuelle

Les caméras ont maintenant la capacité de filmer de jour comme de nuit, en extérieur comme en intérieur. Elles peuvent pivoter sur elles-mêmes, être contrôlées par joystick ou automatiquement via un logiciel, ... De plus les caméras peuvent être regroupées en réseau, ce qui permet d'obtenir un champ de vision bien plus large, rendant ainsi possible le suivi de passants ou de véhicules. Conséquence de la généralisation d'Internet, de nombreuses entreprises ont opté pour des caméras IP. Ces dernières sont consultables et contrôlables depuis n'importe quel lieu dans le monde. Par définition, une caméra IP possède les fonctions optiques d'une caméra auquel il a été ajouté un serveur web interne. On peut donc raccorder facilement plusieurs caméras sur différents postes en utilisant une architecture réseau et en connectant ce réseau à Internet. Il est alors simple de s'y connecter depuis l'extérieur. Ces caméras se branchent par l'intermédiaire d'un port RJ45, elles sont aussi disponibles avec une interface Wifi. Ce nouveau système est donc plus facile d'utilisation et surtout moins onéreux.

1.3 Exemple d'application à la reconnaissance

1.3.1 Reconnaissance automatique des plaques d'immatriculation

Les autorités britanniques viennent de mettre en place un système de reconnaissance automatique des plaques d'immatriculation des véhicules intitulé ANPR[1]. Ce système combine vidéosurveillance "intelligente" et bases de données policières. Il permet donc de savoir en quelques secondes si un véhicule a été volé, s'il n'est pas assuré, si son propriétaire est recherché ou s'il ne s'est pas acquitté de certaines taxes. A l'heure actuelle, l'ANPR fait ses premières apparitions dans les brigades britanniques. Les véhicules dotés de ce système sont équipés de plusieurs caméras, d'un système GPS, d'un écran tactile coté passager et d'un ordinateur relié au réseau informatique général de la police. Lorsque ce dispositif est en action, il prend en photo tous les véhicules passant dans le champ des caméras, questionne la base de données puis informe le policier par une alerte sonore, différente en fonction du problème détecté, et affiche la plaque d'immatriculation sur l'écran. Néanmoins si un véhicule est en règle, la photo, l'heure, la date et sa position GPS sont tout de même enregistrées dans la base de données

[1] Automatic Number Plate Recognition

policière. Cette dernière devient nationale en Angleterre, ainsi les données concernant tous les véhicules contrôlés au hasard dans le pays sont stockées dans la National ANPR Data Centre.

L'ANPR peut analyser jusqu'à 3000 véhicules par heure. Il détecte aussi bien les véhicules arrêtés que le véhicules en mouvement dans les deux sens de circulation à l'aide de caméras mobiles dirigés par logiciel. Ce système va maintenant être installé à l'aide de caméras fixes camouflées aux endroits stratégiques (parking, carrefour, station service, sortie de bars, zone industriel, aéroport, ...).

Les applications de l'ANPR sont diverses et variées. Un exemple proposé par les créateurs du système est :«Si après un cambriolage, des témoins ont vu s'enfuir une Ford verte mais n'ont pas noté son numéro, nous demanderons au système de repérer toutes les Ford vertes en train de s'éloigner du lieu du crime, dans toutes les directions et de capter leurs numéros»

L'ANPR fonctionne également en utilisant des repères temporels, illustré par l'exemple qui suit donné par les créateurs : «Nous demanderons au système de retrouver toutes les Ford vertes qui ont roulé en direction du lieu du cambriolage avant qu'il ait eu lieu. Nous saurons d'oú elles venaient, et nous pourrons obtenir un listing de leurs déplacements antérieurs sur plusieurs mois.» ou encore «Si nous avons trois viols commis par le même homme à une semaine d'intervalle dans trois quartiers de la ville, le système nous dira quelles voitures se trouvaient près du lieu du crime, à chaque fois, à l'heure dite.».

Ce système pourra également servir à retrouver des témoins, ou encore fournir des preuves. En effet si une personne est soupçonné de meurtre alors que sa voiture à été scanné par l'ANPR de l'autre côté de la ville, alors cette personne pourra être innocenté.

Une extension de l'ANPR est en cours de développement nommé ISAS[1], capable de déceler automatiquement un comportement anormal d'un véhicule (véhicule qui zigzague, qui changer brutalement d'allure, arrêté à un carrefour dangereux, ...). Le projet phare pour les autorités britanniques est celui concernant la détection du chauffeur, dont on peut facilement en comprendre les raisons.

1.3.2 Détecteur de visages développé par Quividi

La détection de visages est un enjeu majeur en raison des nombreuses applications qui en découlent. Citons comme exemple d'application le système mis en place par l'entreprise Quividi[2]. Cette dernière a développé une solution d'analyse en temps réel des personnes passant à proximité d'un point d'intérêt (par exemple une vitrine ou un panneau publicitaire). Le système détecte grâce à une camera un visage, analyse ce dernier (homme/femme, direction des yeux, temps d'attention, ...) puis enregistre ces données dans une base de données, pour pouvoir en tirer des statistiques quant au rapport entre le type de population et leurs réactions vis-à-vis du point d'intérêt. Cette entreprise développe également des applications en temps réel : modification d'un spot de publicité en fonction du sexe des spectateurs, de leur âge, ...

Cependant, ce système n'est pas assimilable à de la vidéosurveillance, en effet, le flux vidéo n'est pas enregistré, et n'est pas non plus transmis vers un poste de contrôle.

[1] analyse intelligente des situations

[2] http ://www.quividi.com

1.3.3 La ville de Puteaux

Pour parler des systèmes de vidéosurveillance, nous pouvons aborder le cas de la ville de Puteaux[3], en effet, cette ville est une des villes française les plus « vidéosurveillées ». La Mairie a fait installé 190 caméras dans les lieux publics de la ville (parkings, rues, mairie, ...). La mise en place de ce système n'est pas acceptée par tous les citoyens de Puteaux[4], qui regrettent en général le manque de concertation, et le manque d'information sur les lieux où sont installées les caméras ; ainsi que le manque d'informations sur le traitement des données enregistrées. On peut aussi regretter le fait que le site de la ville de Puteaux ne fasse nulle part mention de ce système de surveillance.

[3] Hauts de Seine

[4] http ://www.puteauxvideo.com

1.4 La vidéosurveillance et la loi

Tout commence dans les années 80, à Hyères plus exactement. La municipalité a confié à la police municipale la gestion de ses activités vidéo et informatique.

Le 13 juillet 1988, la commission nationale informatique et des libertés (CNIL[1]) lui infligeait un avertissement, ordonnant la destruction du fichier de la population déjà deux fois détruit, ainsi que la suppression de la connexion informatique avec le central des cartes grises détenues par les préfectures du Var et des Alpes-Maritimes [2].

Le texte légalisant la vidéosurveillance est très controversé. En effet, C. Pasqua, ministre de l'intérieur sous la cohabitation entre 1993 et 1995 avait pour objectif d'écarter toute présence de la commission nationale informatique et des libertés afin de garder le dispositif d'organisation et de contrôle de la surveillance entre les mains du pouvoir gouvernemental incarné par le préfet mais il n'obtiendra pas une totale satisfaction [2].

Le président de la république, le 22 juin 1994, attire l'attention du gouvernement sur le risque de restriction de la liberté constitutionnelle d'aller et venir.

A. Heinis, sénateur, a bien montré le 5 juillet 1994 que la vidéosurveillance se situait au carrefour de la vie privée, du droit du travail, de l'informatique et des libertés individuelles ce qui rend complexe le cadre juridique touchant l'image. Pour ces raisons, l'analyse du sujet commencera par les conditions légales d'utilisations de la vidéosurveillance puis les limites de son utilisation, la licence d'utilisation du logiciel, un tableau récapitulatif de toutes les sanctions encourues et enfin les extraits de lois concernés. La rédaction de cette partie a été faite en s'appuyant sur le document "Droit à l'image et droit de l'image" [3].

1.5 Conditions d'utilisations

La surveillance vidéo est soumise à une détection de visages. Le logiciel procède donc directement à une reconnaissance d'identité. De plus, une manipulation de l'image est rendu possible notamment si l'on désire stocker ces images dans une base de données.

Par conséquent, une déclaration ordinaire doit être effectuée auprès de la CNIL. Enfin, conformément à la loi n° 95-73 du 21 janvier 1995 dont un extrait relatif à la vidéosurveillance est disponible page 92, l'installation des systèmes de vidéosurveillance est soumise à l'autorisation du préfet bien que, pour le moment, suite à cette même loi, la vidéo n'ait pas de pouvoir juridique reconnu en France mais dispose toutefois d'un fort pouvoir dissuasif.

La vidéosurveillance est en outre, règlementée par les dispositions de la loi n° 95-73 du 21 janvier 1995 et du décret n° 96-926 du 17 octobre 1996, dispositions qui ont été précisées par une circulaire du 22 octobre 1996 (JO du 7 décembre 1996).

1.5.1 Les lieux publics ou ouvert au public

Ce logiciel peut être utilisé dans différents cadres dont le secteur public. Il faut pour cela la nécessité d'un impératif de sécurité. Pour empêcher le grignotage des libertés par le progrès technologique, l'étendu de la surveillance a été délimitée. Il y a un point d'équilibre entre les nécessités publiques et les libertés individuelles.

[1] www.cnil.fr/

[2] Le Monde, 16 juillet 1988 : "Hyères ses caméras et ses fichiers"

Suite à plusieurs discutions au Parlement et suite notamment, à l'intervention de R. Pagès le 10 novembre 1994 auprès du ministre, il a été décidé que la vidéosurveillance peut-être utilisée pour protéger certains bâtiments et installations publics et leurs abords, ainsi que pour des installations utiles à la défense nationale. Son utilisation peut, en outre, être étendue à la régulation du trafic routier, à la constatation des infractions aux règles de la circulation et enfin, à la prévention des atteintes à la sécurité des personnes et des biens.

Le nécessaire respect de la vie privée

La vidéosurveillance est par contre fermement et résolument interdite n'importe où sur la voie publique, y compris pour visionner l'intérieur ou l'entrée des immeubles car elle est attentatoire à la liberté fondamentale d'aller et venir ainsi qu'au respect de la vie privée. (cf. page 17).

La durée de conservation des données

Cette durée est règlementée lors de la demande d'autorisation d'utilisation de la vidéosurveillance mais ne peut excéder un mois. Au delà de la date fixée, les données enregistrées doivent être détruites. Il peut cependant y avoir un délai supplémentaire lors d'une enquête de flagrant délit, d'une enquête préliminaire ou d'une information judiciaire.

L'information des personnes

Lors de son entrée dans un lieu sous surveillance vidéo, le public doit être informé de manière claire et permanente de l'existence du système.

1.5.2 Les lieux privés

Ce système de vidéosurveillance peut être implanté dans des lieux privés, c'est-à-dire fermé au public, s'il relève des dispositions de la loi n° 78-17 du 6 janvier 1978 relative à l'informatique[3], aux fichiers et aux libertés.

1.6 Limites d'utilisations

L'utilisation non autorisée d'images de choses ou de personnes fait courir à l'utilisateur le risque d'être condamné civilement et pénalement. De plus, la qualité de fonctionnaire peut être perdue à l'occasion d'une condamnation.

1.6.1 Risque civil

L'article 9 du Code civil stipule :

[3]loi disponible à l'adresse www.cnil.fr/index.php ?id=301

«Chacun a droit au respect de sa vie privée »(loi du 17 juillet 1970 tendant à renforcer la garantie des droits individuels des citoyens).

L'usage, sans son autorisation, de l'image d'une personne dans le cadre de sa vie privée peut donc entraîner la mise en cause de la responsabilité de l'utilisateur.

Si l'usage fait apparaître en plus une intention de nuire, l'affaire sera alors traitée au pénal.

1.6.2 Risque pénal

Atteinte à la vie privée

L'intention de nuire n'est pas obligatoirement nécessaire à la pénalisation d'une atteinte à l'image d'une personne. L'article 1382 du Code civil prévoit :

«Tout fait quelconque de l'homme qui cause à autrui un dommage oblige celui par la faute duquel il est arrivé, à le réparer».

La faute lourde est la faute commise avec intention de nuire. L'usage de l'image d'une personne avec intention de nuire est donc passible de plusieurs sanctions pénales en rapport aux articles 226-1, 226-2, 226-3.

Pour les personnes présumées innocentes dont une image serait diffusée alors qu'elles sont menottées ou pour les victimes d'attentats dont il aurait été porté atteinte à leur dignité, la peine encourue est définit dans l'article 35 ter I de la loi du 29 juillet 1881 dite loi sur la liberté de la presse.

De plus, la loi informatique et liberté n ° 78-17 du 6 janvier 1978 réprime fortement l'usage illégal de données nominatives tant sur fichier informatique que sur fichier mécanographique, ainsi que leurs divulgations lorsqu'elle porte atteinte aux personnes (article 226-17 et suivants du code pénal).

Les sanctions encourues sont décrites page 17.

1.6.3 Risque de perte de la qualité de fonctionnaire

Suivant l'infraction commise par l'agent public dans le cadre de ses fonctions à l'occasion de l'usage d'images de choses ou de personnes, en particulier si l'acte a porté atteinte à la mission de service public confiée par l'Etat, et a été diffamatoire, le juge peut appliquer les dispositions de l'article 226-31 du Code pénal (privation des droits civiques, interdiction d'exercer un emploi public).

1.7 Licence d'utilisation

L'employeur se voit titulaire des droits du logiciel crées par ses employés. Il possède donc le droit à la reconnaissance de paternité du logiciel, le droit à la première divulgation, le droit d'exploitation. Ce dernier se concrétise par l'octroi d'une licence à l'acheteur. Une licence classique comporte les éléments suivant : possibilité d'effectuer des copies de sauvegarde, possibilité d'effectuer une diffusion limitée du produit, nécessaire à son utilisation par l'acheteur ou son représentant. Les modifications apportés au produit requiert un accord du producteur et enfin le code source n'est ni cédé, ni accessible à l'acheteur.

1.8 Tableau récapitulatif des sanctions encourues

<table>
<tr><th>INFRACTION</th><th>TEXTE</th><th>PEINE</th><th>CONSEQUENCES</th></tr>
<tr><td>traitements automatisés d'informations nominatives sans respecter la loi informatique et liberté</td><td>Code pénal 226-16</td><td rowspan="2">45 000 euros
3 ans de prison</td><td rowspan="13">Risque de perte de la qualité de fonctionnaire par décision du juge pénal (Art 226-31 du Code pénal : privation des droits civiques, interdiction d'exercer un emploi public</td></tr>
<tr><td>conserver des informations sous une forme nominative au-delà de la durée prévue</td><td>Code pénal 226-20</td></tr>
<tr><td>traitements automatisés d'informations nominatives sans garantie de sécurité</td><td>Code pénal 226-17</td><td rowspan="4">300 000 euros
5 ans de prison</td></tr>
<tr><td>collecte de données nominatives par un moyen frauduleux, déloyal ou illicite</td><td>Code pénal 226-18</td></tr>
<tr><td>traitement d'informations nominatives malgré l'opposition de cette personne</td><td>Code pénal 226-19</td></tr>
<tr><td>données nominatives faisant apparaître les origines raciales ou les opinions politiques, philosophiques ou religieuses ou les appartenances syndicales ou les moeurs détourner les informations nominatives de leur finalité</td><td>Code pénal 226-21</td></tr>
<tr><td>Recueil d'information nominatives portant atteinte à la personne</td><td rowspan="2">Code pénal 226-22</td><td>15 000 euros
1 an de prison</td></tr>
<tr><td>Divulgation d'information nominative portant atteinte à la personne</td><td>7 500 euros
1 an de prison</td></tr>
<tr><td>Contrefaçon</td><td>CPI L 335-2</td><td>300 000 euros
3 ans de prison</td></tr>
<tr><td>Atteinte à la vie privée</td><td>Code pénal 226-1</td><td>45 000 euros
1 an de prison</td></tr>
<tr><td>Usage de l'image sans autorisation de la personne</td><td>Code pénal 226-8</td><td>15 000 euros
1 an de prison</td></tr>
<tr><td>Atteinte à la présomption d'innocence</td><td rowspan="2">art. 35 ter 1 de la loi du 29 juillet 1881 dite loi sur la liberté de la presse</td><td rowspan="2">15 000 euros</td></tr>
<tr><td>Atteinte à la dignité des victimes d'attentats</td></tr>
</table>

Chapitre 2

Introduction à la Détection de visages dans une image.

Dans cette section, nous introduisons le problème de la détection automatique de visages humains dans une image. On peut l'énoncer ainsi :*"Etant donné une image arbitraire, on doit déterminer si elle contient des visages humains et le cas échéant les extraire"*. C'est un problème appartenant au domaine de la vision par ordinateur. De nombreuses recherches ont été menées sur ce sujet en raison du vaste domaine d'applications possibles. Nous commencerons par étudier les enjeux de la résolution de ce problème. Nous verrons ensuite les difficultés auxquelles il faut faire face. Pour terminer, nous présenterons les différentes techniques de détection de visages qui ont été proposées par la communauté scientifique. Cette section est basée sur l'article : "Detecting Faces in Images : A Survey" [4] ainsi que sur les informations trouvées sur le site "The Face Detection Homepage" [5].

2.1 Enjeux de la détection de visages

De nombreux problèmes en vision par ordinateur sont étroitement liés à celui de la détection de visages. Citons les principaux :

- **Détection des caractéristiques faciales** : le but est de déterminer l'emplacement des différentes caractéristiques faciales dans un visage telles que les yeux, le nez, la bouche, etc... [6].
- **Suivi d'un visage** : le but est de suivre en temps-réel un visage dans une vidéo [7].
- **Identification d'expression faciale** : Le problème est d'identifier les états affectifs et émotifs d'une personne en analysant ses expressions faciales [8].
- **Identification (ou Reconnaissance) de visage** : Étant donné une image de visage en entrée, on cherche à la faire correspondre à des modèles de visage contenus dans une base de données [9]. On retourne les modèles ressemblants au visage en entrée le cas échéant.
- **Authentification (ou vérification) de visages** : Le but du problème est de permettre l'identification d'un individu à partir de son visage. Dans ce cas, le système doit connaître à l'avance l'identité de l'individu et vérifier si le visage en entrée est associé à cette dernière [10].

Ainsi, sans l'existence de techniques de détection de visages, certains problèmes cités ci-dessus ne serait pas réalisable (par exemple, l'authentification de visages).

Au niveau des domaines d'applications intéressés par cette technologie, citons la vidéosurveillance et les systèmes d'interaction homme-machine. Un système de détection de visages intégré dans une application de vidéosurveillance pourrait ainsi enrichir cette dernière de fonctionnalités intéressantes. Par exemple, il pourrait servir à détecter si une personne non autorisée est entrée dans une pièce. Mais encore, couplé à un système de reconnaissance de visages, il pourrait permettre de repérer une personne recherchée par les autorités. Au niveau des systèmes d'interaction homme-machine, la détection de visages est un enjeu majeur dans le domaine de la robotique.

2.2 Difficultés associées au problème

Détecter un visage de manière automatique via un ordinateur n'est pas une tâche aisée car il faut prendre en compte un certain nombre de contraintes. Résumons-les dans la liste ci-dessous :

- **Orientation** : L'orientation d'un visage dans une image peut considérablement varier. On peut ainsi avoir des images de visage de face, de profil, à l'envers, inclinée à 45 degrés, ... De plus, certaines caractéristiques faciales telles que le nez ou un oeil peuvent ne pas apparaître suivant certaines orientations.
- **Présence ou absence de composants structuraux** : Le fait de porter une barbe, une moustache ou encore des lunettes rend la détection de visage encore plus complexe. De plus, ces composantes peuvent être de couleur, de forme et de taille différentes. Le problème vient du fait qu'elles peuvent masquer certaines caractéristiques faciales de base. Par exemple, le fait de porter une barbe peut cacher la peau, et par conséquent peut compliquer la tâche des algorithmes de détection se basant sur la couleur de la peau. De même, porter des lunettes rend le travail des algorithmes se basant sur la détection des caractéristiques faciales plus difficile. En effet, les verres des lunettes peuvent entraîner des reflets au niveau de la zone des yeux et ainsi fausser la détection de ces derniers.
- **Expression faciale** : L'aspect d'un visage est directement affecté par l'expression faciale de la personne (sourire, grimace, etc...)
- **Occultation** : Dans une image, un visage peut-être partiellement masqué par un objet ou un autre visage. Par exemple, dans une photo de groupe, il n'est pas rare de voir un visage en arrière plan partiellement recouvert par un visage au premier plan.
- **Angle de capture** : Les images de visages varient considérablement pour diférentes rotations autour de l'axe optique du dispositif de capture.
- **Conditions de prise de vue** : L'aspect d'un visage dans une image varie considérablement suivant les conditions d'éclairage (luminosité, contraste, ...) et la qualité du matériel de capture (résolution, lentilles, ...).

Dû à toutes ces contraintes, on comprend qu'il n'existe pas de solution uniforme au problème de la détection de visages. Au cours des dernières années, plusieurs techniques ont été proposées pour tenter d'apporter une solution la plus efficace possible. Nous allons les présenter brièvement dans la section suivante.

2.3 Les techniques de détection de visages

Plusieurs méthodes de détection de visages ont été proposées au cours des dernières années. On peut les classer selon quatre catégories décrites ci-dessous :

1. Méthodes basées sur les connaissances acquises.
2. Méthodes basées sur les caractéristiques invariantes.
3. Méthodes basées sur la mise en correspondance.
4. Méthodes basées sur l'apparence.

Etudions chacune d'elle un peu plus en détail.

2.3.1 Méthodes basées sur les connaissances acquises

Les méthodes de cette catégorie se base sur la connaissance des propriétés du visage et des relations existantes entre les différentes caractéristiques faciales. Voici quelques exemples qui peuvent être appliqués sur une image pour tenter de déterminer la présence d'un visage :

- La partie centrale d'un visage a des valeurs d'intensité lumineuse uniforme.
- Un visage apparaît souvent avec deux yeux qui sont symétriques, un nez et une bouche.
- Les rapports existants entre les différentes composantes faciales peuvent être représentés par leurs distances et positions relatives.

On peut citer comme exemple la méthode de G. Yang et T.S. Huang [11] basée sur une stratégie de multirésolution.

Le problème de ces méthodes est la difficulté de traduire la connaissance des propriétés du visage humain en des règles bien définies. De plus, il est difficile d'étendre cette approche pour détecter des visages dans différentes orientations, car énumérer tous le cas possibles ne semble pas réalisable. Toutefois, ces méthodes sont efficaces pour détecter des visages de face.

2.3.2 Méthodes basées sur les caractéristiques invariantes

Les algorithmes contenus dans cette catégorie se basent sur les caractéristiques faciales ne changeant pas même quand l'orientation du visage, l'angle de prise de vue ou les conditions d'éclairage varient. On trouve ainsi dans cette catégorie des techniques basées sur :

- la détection de la couleur de la peau
- la détection des différentes caractéristiques faciales
- etc...

Citons comme exemple la méthode développée par S. McKenna, S. Gong, et Y. Raja se basant sur la détection de la couleur de la peau [12].

Le problème des méthodes de cette catégorie est qu'elles ne sont pas efficaces si les images en entrée sont de faible luminosité ou si certaines parties d'un visage sont masqués. De plus, le fait qu'un visage soit ombragé rend l'extraction des caractéristiques faciales très difficiles.

2.3.3 Méthodes basées sur la mise en correspondance

Les techniques employées ici se font à l'aide de motifs de visage préalablement créés et judicieusement choisis. Le principe est de calculer la corrélation entre des zones de l'image d'entrée et les motifs de visages. On peut citer comme exemple les travaux de A. Lanitis, C.J. Taylor et T.F. Cootes basés sur des modèles de visage déformables [13].

L'avantage de ces techniques est qu'elles sont simples à implémenter. Toutefois, les premières proposées ne pouvaient pas faire face aux changements d'orientation, de forme et de taille des visages. C'est pourquoi des méthodes basées sur des modèles de visage multirésolution, multi-échelles et déformables ont été proposées par la suite.

2.3.4 Méthodes basées sur l'apparence

Les méthodes de cette catégorie reviennent à traiter le problème de la détection de visages comme un problème de classification. Le but est de déterminer si, étant donné une image, si ce qu'elle représente appartient à la classe des visages ou à celle des non-visages. Afin de séparer ces dernières, on utilise des techniques d'apprentissage automatique supervisé. Pour cela, on utilise un jeu de données d'apprentissage, ici d'une part des images représentant des visages et d'autre part des images ne représentant pas des visages, qui permettra de fixer une règle de séparation entre les deux classes. Un exemple de méthode appartenant à cette catégorie est celle développée par H. Rowley, S. Baluja et T. Kanade basée sur des réseaux de neurones [14].

Ce sont à l'heure actuelle les techniques les plus efficaces et les plus adéquates pour une détection temps-réel.

Notre projet s'appuie sur une technique appartenant à cette catégorie. Nous allons par conséquent détailler son fonctionnement dans la partie suivante.

Chapitre 3

Analyse de l'existant

Notre projet se base sur les travaux de L. Carminati [1], qui dans le cadre de sa thèse a développé un système de détection de visages dans un flux vidéo en quasi temps-réel. Ce système est basé sur les machines à vecteur de support et a pour point de départ les travaux de E. Osuna, R. Freund et F. Girosi [15]. Cette technique sépare des portions d'image en deux classes complémentaires : visage et non-visage. De plus, pour optimiser cette classification, des traitements supplémentaires sont appliqués sur l'ensemble des portions d'image reconnus comme étant des visages dans le but d'éliminer celles qui n'en sont pas.

Dans la suite, nous commencerons par introduire le principe des machines à vecteurs de support puis nous expliquerons comment elles peuvent être utilisées pour détecter un visage dans une image.

3.1 Les machines à vecteurs de support

Cette présentation des machines à vecteurs de support a été réalisée à l'aide du chapitre 9 du livre de A. Cornuéjols et L. Miclet [16] et de l'article "Une nouvelle méthode d'apprentissage : les SVM. Séparateurs à vaste marge" [17].

3.1.1 Introduction

Une machine à vecteurs de support (Support Vector Machine ou SVM) est une méthode de *classification* binaire par *apprentissage supervisé*, introduite par V. Vapnik en 1995 [18]. Elle repose sur l'existence d'un classificateur linéaire dans un espace approprié. Le but est de séparer des données selon deux classes. Pour cela, elle fait appel à un jeu de données d'apprentissage pour apprendre les paramètres du modèle. Dans cette étude, nous commencerons par énoncer le principe de fonctionnement général des SVM. Nous étudierons ensuite le cas où les données en entrée à classifier ne sont pas linéairement séparables, cas le plus répandu dans la vie réelle, et les solutions proposées pour y remédier. Nous terminerons en détaillant les différents principes mathématiques sur lesquels repose SVM. Dans la suite, nous représenterons les données par des points dans un plan.

3.1.2 Principe de fonctionnement général

On dispose d'un ensemble de données d'apprentissage (appelées exemples d'apprentissage) représentables dans un espace d'une certaine dimension et distribuées selon deux classes.

Le but de SVM est de trouver un séparateur linéaire entre ces deux classes d'exemples qui maximise la distance entre ces dernières. Dans le cas de SVM, ce séparateur est un *hyperplan*. Un hyperplan est la généralisation à n dimensions d'une ligne (1 dimension) séparant un espace à 2 dimensions, ou d'un plan (2 dimensions) séparant un espace à 3 dimensions. Une fois, cet hyperplan déterminé, il est utilisé pour déterminer la classe à laquelle appartient une donnée non rencontrée dans les exemples d'apprentissage. Le schéma ci-dessous illustre ces propos.

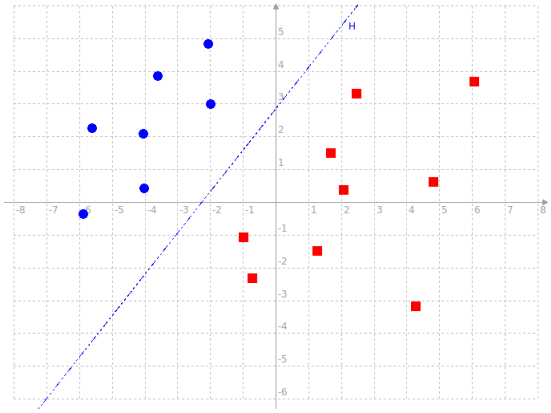

FIG. 3.1 – Séparation de deux classes de données par un hyperplan H

Les points des deux classes les plus proches de l'hyperplan les séparant sont appelés *vecteurs de support*. Le schéma ci-dessous illustre ce concept.

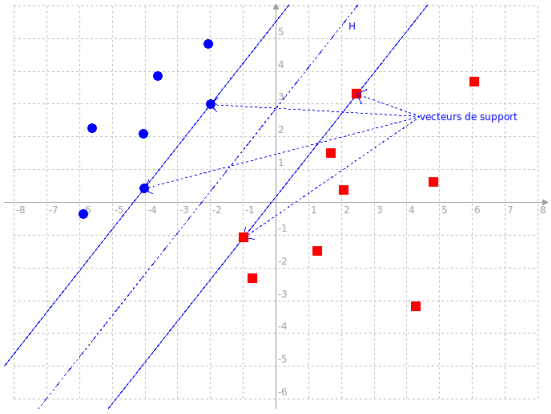

FIG. 3.2 – Hyperplan et vecteurs de support

Evidemment, il existe une infinité d'hyperplan valide séparant les deux classes mais le but de SVM est de déterminer un hyperplan optimal. Cela revient à chercher un hyperplan valide qui passe "au milieu" des points des deux classes d'exemples et qui maximise la distance minimale entre ces dernières. On appelle cette distance *marge* entre l'hyperplan et les distances. L'hyperplan optimal est donc celui qui maximise la marge. Le fait d'avoir une marge la plus large possible procure plus de sécurité lorsque l'on cherche à classer une donnée inconnue (i.e. ne faisant pas partie des exemples d'apprentissage). Ce phénomène est illustré dans la figure ci-dessous. Sur la partie droite, on a un hyperplan optimal qui implique la classification

correcte d'un nouvel exemple. Tandis que sur la partie gauche, l'hyperplan choisi possède une faible marge et le précédent exemple se voit mal classé.

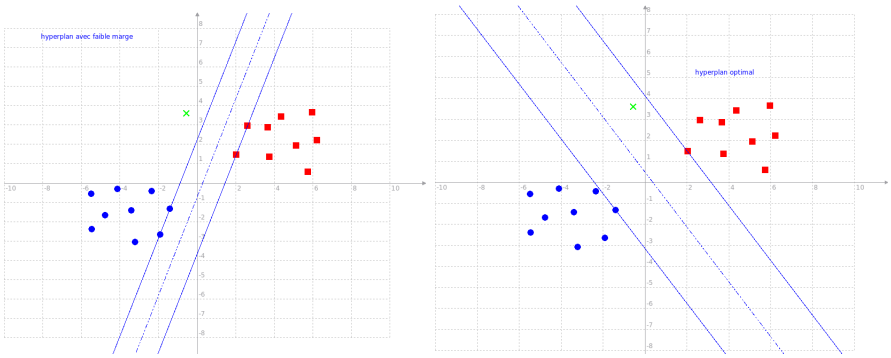

FIG. 3.3 – Ilustration de l'intérêt de choisir l'hyperplan de marge maximale

Enfin, de façon générale, la classification d'un nouvel exemple inconnu se fait suivant sa position par rapport à l'hyperplan optimal. Dans la figure ci-dessous le nouvel élément, représenté par une croix verte, sera classé dans la catégorie des "ronds".

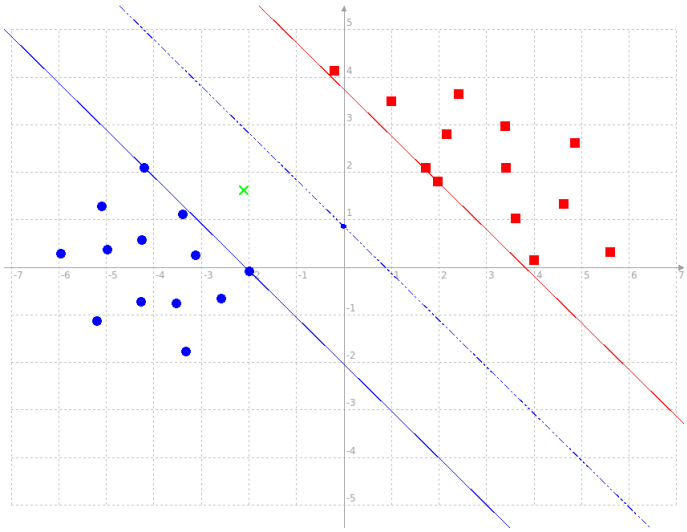

FIG. 3.4 – Ilustration de la classification d'un nouvel exemple

3.1.3 Problèmes linéaires et non-linéaires

Il existe deux cas au problème SVM, celui où les données d'apprentissage en entrée sont linéairement séparables dans leur espace de représentation et celui où elles ne le sont pas. Le premier cas est simple car il est alors facile de trouver un séparateur linéaire entre les deux classes d'exemples. Mais dans la plupart des problèmes réels, les données ne sont pas linéairement séparables et il n'est donc pas possible de déterminer un hyperplan pour les classifier.

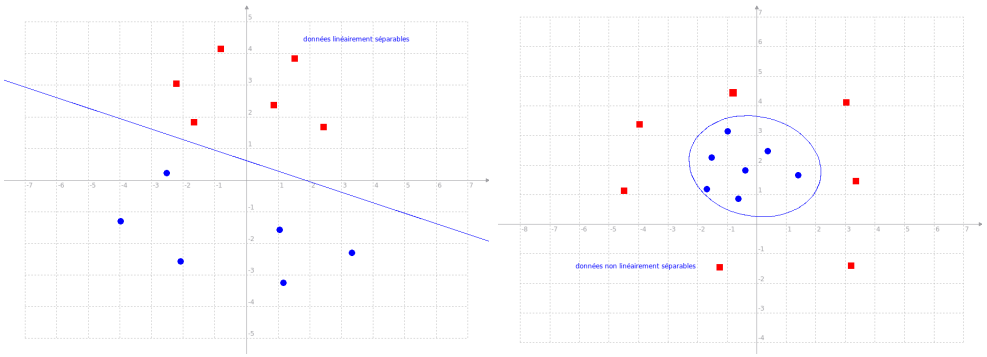

FIG. 3.5 – Séparateurs linéaires et non-linéaires

Pour palier à ce problème, l'idée de l'algorithme SVM est de changer l'espace de représentation des données d'apprentissage. On reconsidère donc le problème dans un espace de dimension supérieure à celui d'origine, appelé *espace de redescription*, via une fonction de transformation non linéaire appelé *fonction noyau*. Plus la dimension de l'espace de redescription est grande, plus la probabilité de trouver un hyperplan séparateur est grande.

3.1.4 Traduction mathémathique

Pour formaliser mathématiquement les principes énoncés plus haut, on considère que les deux classes des exemples d'apprentissage sont les exemples "positifs" et les exemples "négatifs". Nous allons rapidement voir comment se traduit le problème de la maximisation de la marge ainsi que celui du changement d'espace lors des cas de non-séparation linéaire dans l'espace d'origine.

Maximisation de la marge

Commençons par formaliser le concept d'hyperplan. Un hyperplan étant un séparateur linéaire, l'équation le représentant est de la forme :

$$h(x) = w.x + b$$

où w est le vecteur définissant l'hyperplan, perpendiculaire à ce dernier, et b un vecteur de décalage permettant d'augmenter la marge. Sans ce dernier, l'hyperplan serait obligé de passer par l'origine de l'espace de description ce qui restreindrait la solution au problème.

Attaquons nous maintenant au problème de la maximisation de la marge. On dipose d'un ensemble de données d'apprentissage de la forme :

$$\{(x_1, c_1), (x_2, c_2), ..., (x_m, c_m)\}$$

où les c_i sont des constantes prenant les valeurs 1 ou -1, dénotant la classe à laquelle appartiennent les vecteurs x_i. Supposons que ces exemples soient linéairement séparables dans leur espace d'origine. Comment détermine-t-on alors l'hyperplan optimal les séparant ? Le schéma ci-dessous servira à illustrer nos propos.

Dans la recherche de l'hyperplan optimal, on utilise deux hyperplans parallèles, passant chacun par un certain nombre de vecteurs d'une des deux classes (les fameux vecteurs de

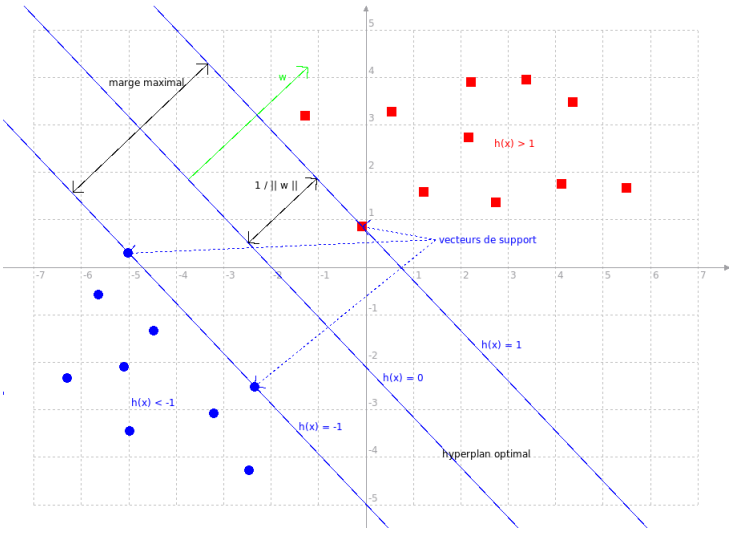

FIG. 3.6 – Illustration de la recherche de l'hyperplan optimal

support). Comme nos données sont linéairement séparables, on peut choisir ces hyperplans de telles sortes qu'il n'y ait aucune données entre. Il est démontrable que l'équation de ces deux hyperplans sont :

$$\begin{cases} h(x)= \ \ 1 \longleftrightarrow w.x+b=1 \\ h(x)=-1 \longleftrightarrow w.x+b=-1 \end{cases}$$

La distance entre ces deux hyperplans vaut $\frac{2}{||w||}$. On cherche à la maximiser. De plus, on doit s'assurer qu'aucun exemple ne se trouve entre les deux hyperplans. Cela revient donc à minimiser $||w||$ sous certaines contraintes, soit à résoudre le problème d'optimisation suivant portant sur les paramètres w et b :

$$\begin{cases} \text{Minimiser} & \frac{1}{||w^2||} \\ \text{sousles contraintes} & c_i(w.x_i+b) \geq 1, \ 1 \leq i \leq m \end{cases}$$

Cette écriture du problème est appelée *formulation primale* et implique le réglage de $d+1$ paramètres, où d est la dimension de l'espace des données d'apprentissage. Le problème peut être résolu pour des valeurs de d assez petites mais devient difficilement envisageable pour des valeurs de d dépassant quelques centaines. Heureusement il existe une transformation de ce problème en une *formulation duale* que l'on peut résoudre en pratique.

La formulation duale du problème s'écrit :

$$\begin{cases} Max_\alpha \left\{ \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j c_i c_j (x_i.x_j) \right\} \\ \alpha_i \geq 0, \ 1 \leq i \leq m \\ \sum_{i=1}^m \alpha_i c_i = 0 \end{cases}$$

C'est un problème quadratique. Les coefficients α_i sont appelés *multiplicateurs de Lagrange*. Dans ce cas, l'hyperplan solution correspondant peut alors être écrit sous la forme :

$$h(x) = (w^*.x) + b^* = \sum_{i=1}^{m} \alpha_i^* u_i.(x.x_i) + b^*$$

où les α_i sont les solutions du problème dual et b^* est obtenue en utilisant n'importe quel exemple critique (x_c, c_c), où x_c est un vecteur de support, dans l'équation :

$$\alpha_i[u_i((x_i.w^*) + b) - 1] = 0$$

Changement d'espace de description et fonctions noyau

On note Φ une transformation non linéaire de l'espace d'entrée X en un espace de redescription $\Phi(X)$:

$$x = (x_1, x_2, ..., x_d) \xrightarrow{\Phi} \Phi(x) = (\Phi_1(x_1), \Phi_2(x_2), ..., \Phi_d(x_d), ...)$$

où x_k est la kième composante du vecteur x. De façon générale, le vecteur image $\Phi(x)$ est de dimension supérieure à d, d'où les points de suspension à la fin de la liste de ses composantes.

Le problème d'optimisation se traduit dans ce cas par :

$$\begin{cases} Max_\alpha \left\{ \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j c_i c_j \langle \Phi(x_i), \Phi(x_j) \rangle \right\} \\ \alpha_i \geq 0, \ 1 \leq i \leq m \\ \sum_{i=1}^m \alpha_i c_i = 0 \end{cases}$$

où $\langle ., . \rangle$ désigne le produit scalaire dans le nouvel espace.

L'équation de l'hyperplan séparateur dans cet espace est :

$$h(x) = (w^*.x) + b^* = \sum_{i=1}^{m} \alpha_i^* u_i \langle \Phi(x), \Phi(x_i) \rangle + b^*$$

où les coefficients α_i^* et b^* sont obtenus comme précédemment.

Le problème est que le produit scalaire $\langle \Phi(x), \Phi(x_i) \rangle$ devient impossible à calculer quand la dimension de $\Phi(X)$ augmente. Heureusement, on peut dans certains cas s'arranger pour court-circuiter le passage par les calculs dans l'espace de redescription. En effet, il existe des fonctions bilinéaires symétriques positives $K(x, y)$, appelées *fonctions noyau*, faciles à calculer et dont on peut démontrer qu'elles correspondent à un produit scalaire $\langle \Phi(x), \Phi(y) \rangle$ dans un espace de grande dimension.

Lorsqu'une telle correspondance est exploitable, le problème d'optimisation précédent est équivalent au problème suivant :

$$\begin{cases} Max_\alpha \left\{ \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j c_i c_j K(x_i, x_j) \right\} \\ \alpha_i \geq 0, \ 1 \leq i \leq m \\ \sum_{i=1}^m \alpha_i c_i = 0 \end{cases}$$

L'équation de l'hyperplan séparateur recherché devient par conséquent :

$$h(x) = (w^*.x) + b^* = \sum_{i=1}^{m} \alpha_i^* u_i K(x, x_i) + b^*$$

Voici quelques exemples de fonctions noyau :

- Linéaires : $K(x, y) = x.y$
- Polynomiales : $K(x, y) = (x.y)^n ou (x.y + c)^n$
- Gaussiennes : $K(x, y) = e^{-\frac{||x-y||^2}{2\sigma^2}}$
- Sigmoïdes : $K(x, y) = tanh(a(x.y - b))$

3.2 Application des SVM à la détection de visages

Le principe de l'application des SVM pour la détection de visages, dans la thèse de L. Carminati, est de définir le vecteur caractéristique d'un visage dans une image comme un ensemble de valeurs de luminance choisies sur une fenêtre de taille prédéfinie. Sur l'image ci-dessous, une fenêtre de taille NxN pixels définit un vecteur X, à N^2 composantes, représentant un visage.

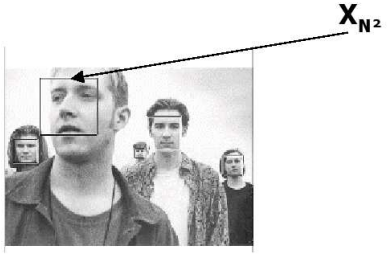

FIG. 3.7 – Représentation d'une image de visage par un vecteur dont les composantes correspondent au valeur de luminance des pixels

On rappelle que la luminance Y est calculée en tenant compte de la sensibilité de l'oeil en fonction des 3 couleurs primaires : Rouge, Vert et Bleu, selon la formule :

$$Y = 0.59V + 0.30R + 0.11B$$

Les exemples d'apprentissage à fournir à l'algorithme SVM seront donc des images de taille fixe qui représenteront soit des visages, soit autre chose que des visages. L'image ci-dessous permet de se faire une idée de ces données d'apprentissage.

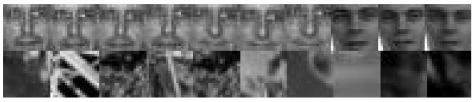

FIG. 3.8 – Echantillon d'exemples d'apprentissage qui serviront à déterminer une séparation entre les classes visage/non visage

A partir de ces exemples, l'algorithme SVM va donc essayer de déterminer un séparateur linéaire entre la classe d'images représentant des visages et son complémentaire, comme schématisé ci-dessous.

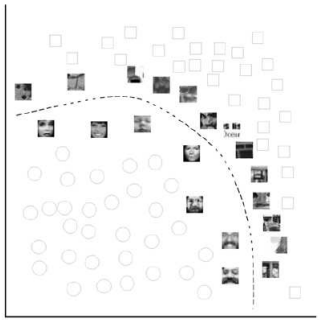

FIG. 3.9 – Etablissement d'une séparation entre les classes visage/non visage par SVM

Chapitre 4

Définition du cahier des charges

Le but du projet est de développer une application de vidéosurveillance intégrant une fonctionnalité de détection de visages. Nous disposons pour point de départ des travaux du docteur L. Carminati. Dans le cadre de sa thèse [1], il a ainsi développé une application de détection de visages en temps réel dans une vidéo. D'après les attentes de notre client, notre travail consiste dans un premier temps à étudier le fonctionnement de cette application et de faire une refonte du code existant[1]. Plus précisemment, nous devrons en repenser l'architecture ainsi qu'améliorer la lisibilité du code. De plus, nous aurons à faire un tri parmi les bibliothèques car beaucoup ont été utilisées lors de la conception de ce programme, et certaines ne sont plus utilisées dans la dernière version. Cette première étape aura pour avantage d'alléger et de rendre moins complexe la structure du programme. Dans un second temps, il nous est demandé d'ajouter de nouvelles fonctionnalités au programme.

Rentrons maintenant dans les détails du cahier des charges de l'application que nous devons développer en énonçant les besoins non fonctionnels et fonctionnels.

4.1 Besoins non fonctionnels

Les besoins non fonctionnels que nous avons retenus sont les suivants :

- **La détection de visages doit se faire en temps réel**

Le premier impératif que devra respecter notre programme sera d'assurer que la détection de visages se fait en temps réel, comme dans l'application de L. Carminati. Pour cela, nous aurons à prendre en compte les caractéristiques des caméras que nous utiliserons, telles le débit de capture et la résolution.

- **L'application doit pouvoir gérer plusieurs flux vidéos**

Il est courant d'installer plusieurs caméras dans le lieu que l'on cherche à surveiller, afin d'éviter les angles morts et avoir un meilleur champ de vision. Ainsi, notre programme se devra d'offrir la possibilité de travailler à partir de plusieurs flux vidéo provenant de différentes caméras.

- **L'application doit être développé de façon à être réutilisable et extensible**

[1] refactoring

Au terme du développement, notre application se devra d'être facilement réutilisable par quiconque en exprimant le besoin. Cela implique de bien définir les règles d'utilisation (en particulier le cadre légal) ainsi qu'avoir une interface avec l'utilisateur adaptée. De même, l'application se devra d'être extensible afin de faciliter son optimisation et l'ajout de nouvelles fonctionnalités. Par conséquent, l'architecture du logiciel devra être pensée de façon à faciliter une éventuelle reprise de développement.

- **Le code devra être clair et compréhensible**

Une des requêtes de notre client, est de produire un code facilement assimilable par des personnes extérieures au projet. Pour cela, nous devrons réaliser un code lisible et bien commenté. Dans cette optique, il sera bon d'introduire certaines conventions de programmation (choix intelligent des identificateurs de variables et de procédures, espacement, indentation, ...) et de s'y tenir.

4.2 Besoins fonctionnels

Après concertation avec notre client, nous avons défini les besoins fonctionnels suivants :

- **La détection de visages devra se faire par faible luminosité**

La première fonctionnalité que nous aurons à mettre en place sera l'amélioration du taux de détection par faible luminosité (dans les limites du raisonnable). En effet, pour le moment en cas de faible luminosité, les visages ne sont pas détectés. Pour être concret, dans la salle où sont installées les caméras, la détection de visage n'est opérationnelle que lorsque tous les néons sont en marche. Il nous faut donc mettre en place un système permettant la détection même lorsque la moitié des néons sont éteints.

- **L'application devra être capable de suivre un visage**

Pour améliorer le taux de détection, nous implémenterons un principe de tracking. Actuellement, la détection de visage ne se fait que sur les zones de l'image en mouvement, or une personne qui rentre dans le champ de la caméra et qui arrête de bouger ne sera plus détectée ! On imagine donc qu'entre deux images d'une vidéo, une personne peut difficilement disparaître. On cherchera donc si, dans la zone de l'image où un visage a été précédemment détecté, un visage n'est pas toujours présent.

- **L'application se devra d'informer l'utilisateur quand elle détectera une intrusion**

Pour finir et pour se rapprocher de la vidéosurveillance, nous créerons une fonction d'alerte par e-mail qui avertira le ou les utilisateurs de l'application de la détection d'une personne dans la salle surveillée. On pourra aussi imaginer qu'une alerte pourra être envoyée, dans le cas de la détection de mouvements, sans nécessairement la détection d'un visage. Un problème technique restera à envisager : la panne du serveur d'envoi de mail (smtp, imap,...)
En cas de détection, l'image du ou des visages seront sauvegardées sur un serveur sécurisé[2]. Ces dernières seront conservées pendant un certain délai fourni lors de l'autorisation préfectorale[3], ce dernier ne pouvant excéder un mois. Passé ce délai, elles seront automatiquement

[2]Code pénal 226-17
[3]Code pénal 226-20

supprimées. Un lien vers le serveur sécurisé sera alors envoyé par mail. L'utilisateur devra s'authentifier et fournir un mot de passe pour avoir accès aux images.

- **Fonctionnalité à priorité basse : Détection de visage de profil**

  Un des problèmes de l'application actuelle est la non détection de visage de profil. Il nous est donc demandé de réfléchir, une fois toutes les autres fonctionnalités implémentées, à l'amélioration du modèle de détection actuel pour qu'il prenne en compte les visages de profil. L'amélioration du modèle n'est pas une chose facile, et ne sera sans doute pas réalisable dans le temps imparti par le projet.

Chapitre 5

Tests préparatoires et tests de validation

5.1 Résultats et Interprétations des Tests préparatoires

Dans cette partie, nous présenterons les différents tests effectués sur le programme existant.

5.1.1 Tests de la détection en temps réel

Pour valider le fonctionnement de l'application existante en temps réel, nous avons testé la reconnaissance de visage sur des temps relativement longs, et en modifiant les conditions de la scène.
Nous avons commencé par travailler sur une scène avec une personne en mouvement, la luminosité dans la pièce est élevée, nous avons alors observé la détection au cours du temps (plusieurs minutes). Dans ce cas, on observe que la détection s'effectue bien en temps réel. Néanmoins, nous avons constaté un ralentissement de la machine et du programme (détection quasi temps réel).
Nous sommes alors passé à l'analyse d'une scène avec plusieurs personnes (4 personnes), on observe alors des ralentissements dans la détection de visage. En effet, sur certaines images, tous les visages ne sont pas détectés.
Pour finir, nous avons analysé une scène avec un visage, où une grande proportion de la scène est en mouvement. On observe alors comme à l'analyse précédente, que le visage n'est pas détecté à toutes les images.

5.1.2 Tests sur des images

Le programme fonctionnant correctement en temps réel, il nous reste à tester la détection de visage sur des images de natures différentes. Pour cela, nous avons adaptés le programme de L. Carminati afin de pouvoir tester son comportement sur des images. Il nous est maintenant possible de faire les différents tests ci-dessous :

- **Tests sur la luminosité des images** : A l'aide d'une image de référence sur laquel nous faisons varier la luminosité, nous allons relever le nombre de détection de visages en fonction de la luminosité de l'image.

- **Tests sur la résolution des images** : Ces tests permettent de savoir, en diminuant progressivement la résolution de l'image, la résolution minimum à utiliser pour une détection maximum de visages.

- **Tests sur le contraste des images** : Nous relèverons le nombre de visages détectés par rapport aux contrastes des images.

- **Tests sur des images flous** : Simple test pour visualiser le comportement du programme face à des images floues.

- **Tests sur des images bruités** : dans cette partie, nous étudierons le nombre de détections faites sur des images bruitées. Les tests seront faits sur des images auxquelles nous avons préalablement ajouté du bruit et de quantités différentes.

- **Tests sur la rotation des images** : L'image de départ ne contient que des visages de face et droits. En faisant varier l'inclinaison des visages, nous pourrons connaître l'angle à partir duquel le programme ne détecte plus de visage.

- **Tests sur des visages avec accessoires** : ces tests vont nous dire si le programme détecte encore des visages lorsque la personne porte des lunettes, un chapeau, une casquette, une cagoule, ...

Les tests sont effectués sur la même image sur laquelle nous ferons varier des paramètres tels que la luminosité, la résolution, le contraste, ... Cet image est de résolution 640 x 480 , elle est en couleur au format jpeg et comporte neuf visages. L'intensité moyenne des pixels des 3 composantes est de 89 sur 255 niveaux.

Tests sur la luminosité des images

La luminosité d'une image est fonction de l'intensité des pixels. Comme nous travaillons sur des images couleurs, l'intensité moyenne des pixels des trois composantes de l'image est prise pour abscisse. En faisant varier celle-ci (augmentation de la luminosité à droite et diminution de celle-ci à gauche), on obtient un graphe (FIG. 5.1, page 35).

Sur ce graphique, on peut voir que le taux de détection de visage chute rapidement pour des images très lumineuses (intensité moyenne des pixels augmentée de plus de 100 par rapport à l'image de départ). Même constatation pour les images peu lumineuses (intensité moyenne des pixels diminués de plus de 80 par rapport à l'image de départ). Nous devons améliorer la détection des visages dans les images sombre, ces tests nous montrent que nous pourrons sûrement augmenter ce taux de détection pour ce type d'images. En l'occurrence il ne faut pas nier que nous travaillerons à partir d'images acquises par une webcam et donc les images seront de moins bonnes qualités (peu de pixels représentés dans les faibles niveaux des trois composantes).

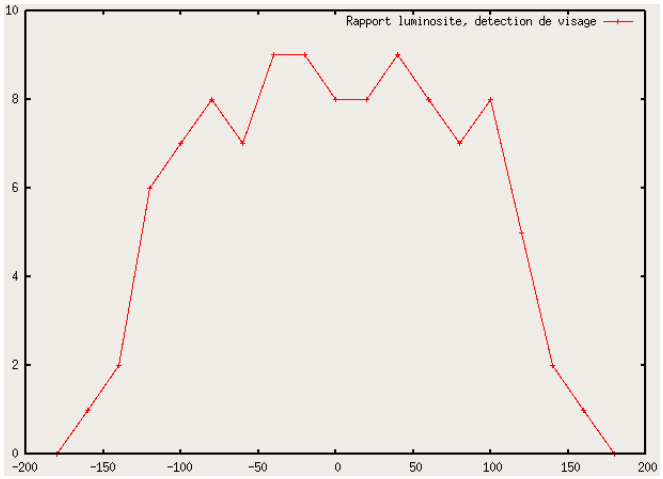

FIG. 5.1 – graphe du nombre de visages détectés en fonction de la luminosité de l'image

Tests sur la résolution des visages

Nous partirons d'une image de résolution de 640 x 480 pixels, en redimensionnant cette image par pas de 50 pixels sur la largeur, nous pourrons déterminer la résolution minimum à utiliser. En diminuant la taille de l'image on obtient le graphe (FIG. 5.2, page 35), seulement la largeur en pixel de l'image est notée en abscisse du graphe.

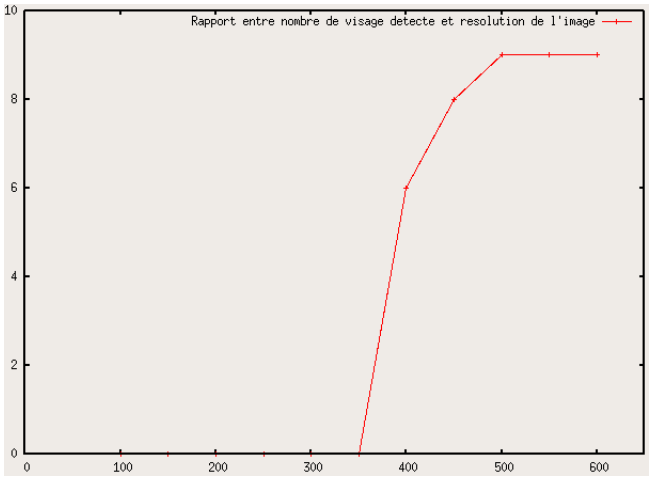

FIG. 5.2 – graphe du nombre de visages détectés en fonction de la résolution de l'image

On peut voir sur ce graphe que le nombre de visages détectés chute considérablement dés que la résolution est en dessous de 450 pixels en largeur. Dans l'image de résolution 500 x 390 pixels, quasiment tous les visages sont détectés et ces visages sont environs de taille 40 x 40 pixels. On ne pourra pas utiliser de séquence où les visages ont une résolution inférieure pour avoir une détection optimale.

Tests sur le contraste des images

Le contraste d'une image est donné par les bornes inférieures et supérieures de l'histogramme, nous allons faire varier le contraste et en présenter les résultats sur un graphe, la borne inférieure sera réduite vers la gauche du graphe et la borne supérieur sera réduite vers la droite du graphe (FIG. 5.3, page 36).

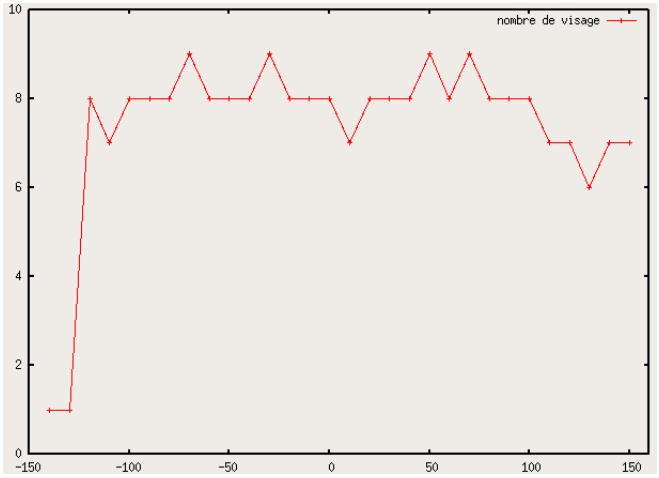

FIG. 5.3 – graphe du nombre de visages détectés en fonction du contraste de l'image

L'observation du taux de détection en fonction d'une valeur de contraste n'est pas très parlante. En effet, à par une nette rupture quand le contraste est très faible, les autres valeurs sont assez semblables. On peut donc en conclure que le contraste d'un visage devra être suffisant pour que le taux détection soit bon.

Tests sur le flou des images

Nous voulions voir le comportement du programme sur des images floues. Pour cela nous appliquerons successivement un filtre moyenneur classique avec pour un masque de taille 3 x 3 (FIG. 5.4, page 37).

Le graphe obtenu nous montre que les images légèrement floues permettent une meilleure détection. Les contours des yeux, du nez et de la bouche étant flous, le programme détecte plus facilement ces parties significatives.

Tests sur le bruit des images

Nous utiliserons un bruit uniforme (type poivre et sel) pour ces tests (FIG. 5.5, page 38).

On peut voir que la détection est résistante aux bruits. Il ne sera donc pas nécessairement indispensable de les filtrer avant la phase de détection (SVM).

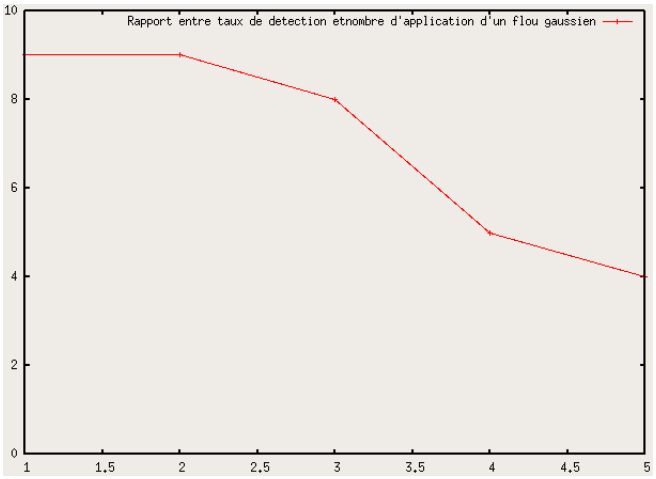

FIG. 5.4 – graphe du nombre de visages détectés en fonction du nombre de filtre moyenneur 3 x 3 appliqué à l'image

Tests sur la rotation des images

L'inclinaison des visages est testée ici, en pivotant l'image selon un angle en abscisse (FIG. 5.6, page 39).

Sur la courbe représentant le rapport entre le nombre de détection de visage et l'angle du visage, par rapport à la verticale, on observe que lorsqu'au delà de 20 degré la détection est nulle. On observe aussi que la détection est de bonne qualité quand le visage n'a pas un angle très important, inférieur à 10 degré.

Tests sur des visages avec accessoires

Nous avons testé avec différents accessoires, la détection est correcte avec des objets couvrants le crane (chapeaux, casquettes, bérets, cagoule), mais aussi avec les lunettes (testé seulement sur des lunettes de vue sans miroir teinté).Donc ces tests sont concluants.

5.2 Conclusion sur les tests préparatoires

Les tests préparatoires nous ont permis de connaître les points à améliorer et surtout si ces points peuvent être significativement améliorables ou peu améliorables. Les tests sur la résolution montrent qu'il sera difficile de détecter des visages de taille inférieure à 40 x 40 pixels. Au niveau de la détection par faible luminosité, elle sera améliorable mais pas significativement car il ne faut pas oublier de prendre en compte les capacités du matériel d'acquisition. Le contraste ne sera pas une caractéristique fondamentale des images acquises pour une bonne détection. Un filtre moyenneur pourra éventuellement être utilisé comme prétraitement à l'image acquise pour améliorer la détection à condition que ceci donne de bons résultats (pas de non visages détectés) et reste dans le cadre d'une détection en temps réel.

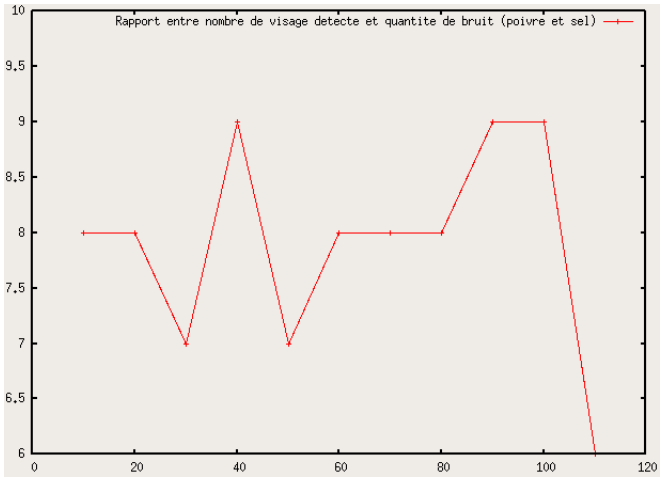

FIG. 5.5 – graphe du nombre de visages détectés en fonction du taux de bruit uniforme dans l'image

5.3 Tests de validation

Au cours du développement, pour chaque besoin fonctionnel qui aura été implémenté, nous devrons effectuer un test de validation. Suivant les résultats de ce dernier, nous pourrons ainsi décider si l'implémentation du besoin fonctionnel associé est correcte ou si elle doit être approfondie. De plus, au terme du développement, nous aurons à mettre en place différents tests de validation finaux qui permettront de vérifier la bonne interaction entre les différents composants de l'application et de s'assurer que cette dernière remplit bien les fonctionnalités définies dans le cahier des charges.

Dans cette partie, nous allons donc détailler comment nous pensons effectuer ces différents tests de validation.

5.3.1 Test de validation pour le besoin fonctionnel : "détection de visages par faible luminosité"

Actuellement, l'application développée par L. Carminati dans le cadre de sa thèse [1] ne détecte pas de visages si la scène n'est pas fortement éclairée. En effet, dans la salle du LaBRI où est déployée l'application, quand les néons sont pleinement allumés la détection se fait sans problème, mais si on baisse leur intensité lumineuse de moitié alors plus aucun visage n'est détecté. Ainsi pour valider la "détection de visage par faible luminosité", nous testerons notre application en prenant cette salle comme référence. Si notre application détecte des visages quand l'intensité lumineuse des néons est diminuée de moitié, nous pourrons ainsi affirmer que la détection a été améliorée. Nous verrons ensuite jusqu'à quel seuil de luminosité la détection se fait et déterminerons si le résultat est acceptable ou si des améliorations peuvent être apportées.

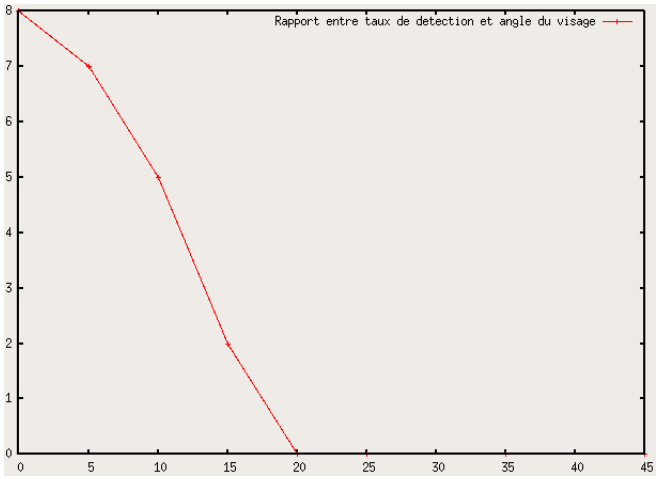

FIG. 5.6 – graphe du nombre de visages détectés en fonction de l'angle d'inclinaison des visages

5.3.2 Test de validation pour le besoin fonctionnel : "suivi de visage"

L'application existante ne détecte pas les visages qui ne sont pas "en mouvement". Ainsi quand un visage "en mouvement" entre dans le champ de la caméra il est détecté, mais à partir du moment où sa position est fixe il ne l'est plus. De plus, l'application n'a actuellement pas les moyens de déterminer si un visage détecté au temps t a été détecté au temps $t-1$. Ainsi, pour démontrer le bon fonctionnement de la fonctionnalité de suivi de visage que nous aurons implémenté, nous procéderons de la façon suivante. Nous ferons évoluer une personne dans le champ de la caméra et observerons le résultat de la détection. Si le visage de la personne est détecté à un temps t et qu'il continue à l'être au temps $t+1$ sans que notre application ne le considère comme un nouveau visage et ce même si le visage est immobile alors ce test sera probant. De plus, pour tester les limites du suivi de visage nous aurons à prendre en compte plusieurs paramètres tels le nombre de personnes dans le champ de la caméra, la vitesse de déplacement d'un visage à l'image, l'orientation du visage, ...

5.3.3 Test de validation pour le besoin fonctionnel : "envoi d'un e-mail d'alerte lors de la détection d'une intrusion"

L'application que nous devons développer devant servir à des fins de vidéosurveillance, il est primordial qu'elle garantisse d'avertir ses usagers qu'une intrusion a été détectée dans le lieu surveillé. La solution à l'heure actuelle qui a été retenue est une alerte par e-mail. Pour tester cette fonctionnalité, nous ferons évoluer une personne dans le champ de la caméra et, à partir du moment où elle est détectée, vérifierons qu'un e-mail d'alerte a bien été envoyé vers une adresse spécifiée dans la configuration de l'application. Pour respecter la législation française, les images des visages détectés devront être stockées sur un serveur sécurisé. Nous aurons donc à tester la bonne interaction de notre application avec ce serveur et garantir aux utilisateurs la confidentialité des images qui y seront stockées.

5.3.4 Test de validation pour le besoin fonctionnel facultatif : "détection des visages de profil"

Si nous parvenons à trouver le temps d'implémenter cette fonctionnalité, le test de validation consistera à vérifier que le visage orienté de profil d'une personne apparaissant dans le champ de la caméra soit bien détecté. De plus, pour valider cette fonctionnalité, les visages de profil-gauche et profil-droit devront être détectés tout en assurant que la détection des visages de face soit effective.

5.3.5 Tests de validation finaux

Au terme du développement, nous devrons tester si notre application répond bien aux demandes formulées par notre client. Pour cela, nous élaborerons divers scénarios d'intrusion dans la salle où est actuellement déployée l'application de L. Carminati et analyserons les réactions de notre application. Nous testerons donc le comportement de notre application en faisant varier plusieurs paramètres, tels :

- le nombre de personnes dans le champ des caméras
- la luminosité
- l'orientation des personnes dans le champ des caméras : de face, de profil, de dos...

En fonction du nombre d'intrusions détectées et des alertes associées, nous pourrons tirer des conclusions sur les qualités fonctionnelles de notre application, en déterminer les conditions optimales d'utilisation ainsi que dresser une liste des défauts.

Chapitre 6

Planning

6.1 Planning initial

Lors de la rédaction de notre planning initial, nous partions sur une refonte profonde, mais pas radicale, du code. Pour cela, nous avions considéré que cette tâche nécessiterait toutes nos forces pendant 2 semaines. Une fois cette étape réalisée, et seulement après sa finition totale, nous pensions démarrer, en deux groupes, la réalisation de deux besoins fonctionnels (amélioration de la détection par faible luminosité, et réalisation d'un embryon de "suiveur"), en affectant environ deux semaines à ces deux tâches. Enfin, pour finir la partie implémentation, nous avions décidé de se mettre ensemble à la réalisation d'un notifieur par envoi de courriel. Le but étant de se conserver une semaine pour rédiger le mémoire final, et finir les tests de validation. Un planning plutôt chargé, nécessitant un travail d'équipe au début, et un travail plus individuel par la suite.

Nous allons maintenant expliquer comment nous avons procédé réellement lors de la conception de ce programme.

6.2 Planning réel

6.2.1 Analyse du code existant

Avant de débuter la programmation, nous avons commencé par analyser dans le détail le code qui nous été fourni. Nous avons eu beaucoup de difficultés à intégrer ce code. Nous avons donc pris la décision, après une semaine d'errance et de doute, d'abandonner le code existant, pour répartir sur des bases plus saines. Il ne s'agissait pas là d'abandonner les concepts que nous avions compris lors de l'analyse du code de L. Carminati, mais de reprendre l'implémentation, en éliminant les librairies inutiles. Nous avons aussi décidé, en accord avec notre client, lors de la deuxième semaine, d'utiliser la librairie *OpenCV* (décrite au chapitre 8.2 page 51).

6.2.2 Programmation de la base

Une fois le choix de repartir à zéro et le choix d'utiliser *OpenCV* effectués, il nous fallait réécrire la base du programme. Nous avons donc séparé notre travail en respectant l'architecture modulaire que nous avions défini.

Deux semaines furent donc consacrées à l'implémentation d'un **Captureur**, d'un **Afficheur**, d'un **Detecteur**[de]**Visages**, d'un **Selecteur**[de]**Zones**[d']**Interet** et d'une **Application** centrale. Ces différentes parties du programmes furent implémentées de manière indépendante, cela était possible puisque les différents modules ne devaient communiquer qu'avec **Application**.
Nous avons aussi gérer notre premier besoin fonctionnel lors de ces deux semaines. En effet, une des particularité de l'algorithme de détection de visages inclus dans *OpenCV* permet la détection de visage même par faible luminosité, à ceci, nous avons ajouté l'égalisation de l'histogramme de l'image, et le problème de la luminosité était réglé.

6.2.3 Suiveur

Un peu à l'image de notre premier besoin fonctionnel, le besoin de suivi a commencé à être gérer pendant la quatrième semaine, et pendant la cinquième semaine. L'implémentation d'un suiveur basique n'était pas très difficile en soit, puisque nous avions conçu l'architecture en le prenant en compte. Cependant le rendre fonctionnel dans une grande partie des cas d'utilisation était moins évident. Son implémentation a donc nécessité un, puis deux éléments du groupe pour être opérationnel.

6.2.4 Alerte Courriel

Pendant que deux personnes suivaient le suiveur (drôle de coïncidence !), une autre personne était affectée à la réalisation d'un notifieur, qui permette l'envoi de courriel. Pour cette tâche, une semaine a été nécessaire.

6.2.5 Stockage des images

Parallèlement aux deux tâches précédentes, la dernière personne du groupe a été responsable de la réalisation d'un enregistreur. La difficulté était d'enregistrer des images de tailles raisonnable, et surtout de le faire en respectant la loi, qui impose que le stockage d'image d'une personne soit réalisé de manière sécurisé (voir Code Pénal 226-17). Le temps a donc été séparé en deux parties, le stockage des images, et la sécurisation de ces stockages.

6.2.6 Application

Pendant l'avancement du **Notifieur** et du **Stockeur**, les deux éléments, qui étaient avant à l'élaboration du suiveur, ont amélioré le fonctionnement de l'**Application**. En utilisant des méthodes trouvées dans le programme de L.Carminati.

6.2.7 Tests de validation

A la fin de la réalisation de chaque module du programme, des tests unaires ont été effectués.
Et pour valider le programme final, la réalisation de test ont nécessité une personne pendant une semaine et demie. Et aussi l'utilisation du matériel central de cette application, un ordinateur et une caméra performante (situés au Labri).

6.2.8 Mémoire

Pendant l'avant dernière semaine, deux personnes ont été responsable de la rédaction du mémoire final. Les deux autres personnes ont rejoints la rédaction une fois leur tâche respective terminée.

6.3 Conclusion

Malgré des débuts un peu difficiles, les besoins fonctionnels ont été respectés. Le planning initial n'a pas été suivi pendant les premières semaines, puis nous avons réussi à retrouver notre planning initial.

Chapitre 7

Architecture et découpage modulaire

7.1 Contexte

Après avoir étudié l'architecture existante, nous nous sommes rendu compte que cette architecture ne nous permettrait pas d'implémenter nos besoins fonctionnels et de rendre le code source lisible. C'est la raison pour laquelle nous avons décidé de refaire l'architecture du programme. Par conséquent, nous avons créé une architecture permettant une meilleure extensibilité et une meilleure modularité tout en conservant un traitement en temps-réel. Nous avons ainsi décidé d'organiser notre architecture selon huit paquetages. Ces derniers sont intitulés **CaptureImage**, **ZonesInterets**, **DetectionVisage**, **Afficheur**, **Notifieur**, **StockeurVisage**, **Application**, et le dernier **FichierConfiguration**. Le schéma global de notre architecture, décrivant le contenu de chacun de ces paquetages et leurs relations, est représenté par la figure 7.1 page 45. Dans la suite, nous allons étudier les fonctionnalités qu'offriront chacun de ces paquetages.

7.2 Les paquetages

7.2.1 Application

Application est le point central de notre architecture. En effet, afin d'avoir le plus de modularité possible, nous avons choisi de n'avoir aucunes communications entre les paquetages. Ainsi, les paquetages sont totalement indépendant les uns des autres et n'ont de compte à rendre qu'à **Application** qui gère tous les paquetages.

Le fonctionnement est simple. En premier lieu, **Application** demande une image à **CaptureImage**. Par la suite, un premier traitement est demandé par **Application** à **ZonesInterets**, il s'agit de sélectionner les zones à analyser. Puis, si des zones sont renvoyées par ce dernier alors un autre traitement est effectué (Détection de visage), sinon **Application** demande une autre image à **CaptureImage**. Le second traitement, appelé par **Application**, est effectué par **DetectionVisage** qui retourne les zones comportant des visages. Si des zones d'image sont renvoyées alors des visages ont été détectés. Ensuite un affichage est effectué par **Afficheur**, et si des visages ont été détectés, alors une notification est envoyée par **Notifieur**, et les visages détectés sont stockés par **StockeurVisage**.
Pour faciliter la configuration du programme, nous avons mis en place un fichier de confi-

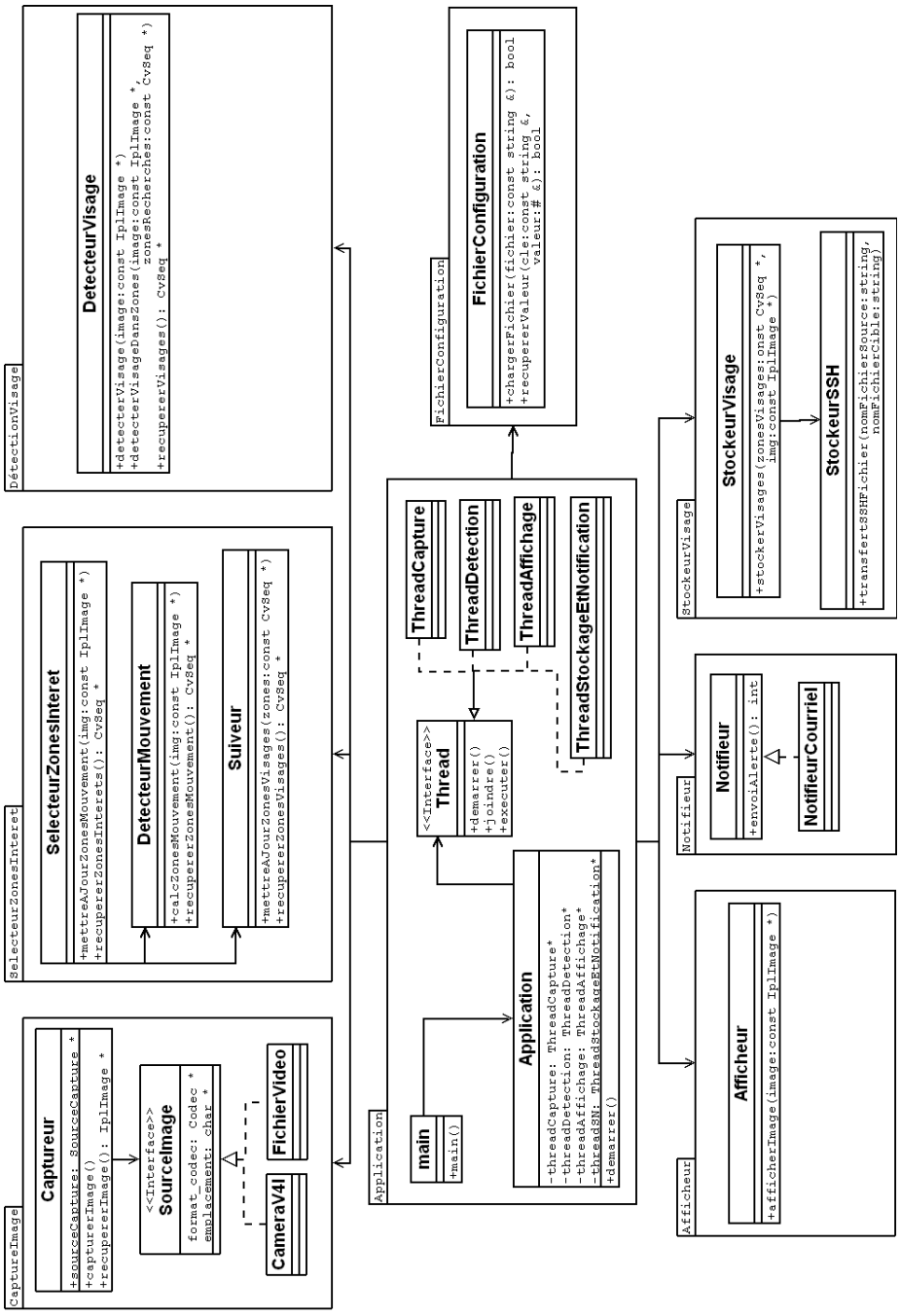

FIG. 7.1 – Schéma de l'architecture de notre application

guration, dans lequel seront stocké les différents paramètres de l'application (adresse de la camera, adresses de stockage, ...) et dont l'utilisation au sein de l'application est réalisée grâce à **FichierConfiguration**.

La figure 7.2 page 47 permet d'illustrer le fonctionnement de l'application.

Pour mieux illustrer le fonctionnement de l'application, nous avons réalisé un diagramme de séquence. Il permet d'observer les transitions entre les différents paquetages, ainsi que le déplacement des différents objets. (7.3 page 48)

7.2.2 CaptureImage

Captureur a pour rôle de capturer une image selon la source en entrée. Il utilise **SourceImage** qui est implémenté par différentes sources comme CameraV4l et FichierVideo. On peut alors lire des images de sources différentes. Les cameras compatibles avec *Video4Linux* 1 et 2 fonctionnent (avec l'installation préalable de le librairie *Video4linux* correspondante). Les fichiers vidéos supportés sont ceux compatible avec la librairie *OpenCV*, qui utilise pour cela la librairie *ffmpeg* (par exemple, les fichiers AVI non compressé, en RGB 24 ou 32 bits).

Captureur retourne donc une image (une IplImage au sens *OpenCV* du terme).

7.2.3 ZonesInterets

Le but est de réduire la zone de recherche des visages dans l'image pour conserver un fonctionnement en temps réel, ce paquetage répond au premier besoin de notre cahier des charges. Pour cela, nous sélectionnons les zones d'intérêts en fonction de plusieurs critères, qui sont les zones en mouvement et les zones issues de **Suiveur**(zones où des visages ont été détectés à l'image précédente). *OpenCV* dispose d'une structure cvSeq permettant de créer facilement une séquence d'objets, nos objets serons des cvRect représentant chacun une zone de l'image.

La classe **DetecteurMouvement** permet donc d'isoler des zones dans lesquelles des différences ont été constatées entre les quatre dernières images. Un seuillage est effectué, pour permettre l'élimination d'un bruit éventuel (la valeur de seuillage peut être modifiée dans le fichier de configuration).

La classe **Suiveur** permet de mémoriser les zones des visages détectés sur l'image précédente et ainsi pouvoir faire un suivi simplifié de visages en vérifiant la présence d'un visage dans ces zones aux traitements suivants. L'utilité de ce suiveur est de ne pas perdre un visage qui devient immobile. Pour augmenter la fiabilité de ce suiveur, les zones où des visages ont été détectés sont conservé et analyser sur plusieurs images suivantes.

7.2.4 DetectionVisage

La fonctionnalité de ce paquetage est de chercher des visages dans les zones passées en paramètre et de retourner les zones qui contiennent un visage.

Ce travail est effectué par une classe nommée **DetecteurVisage**.

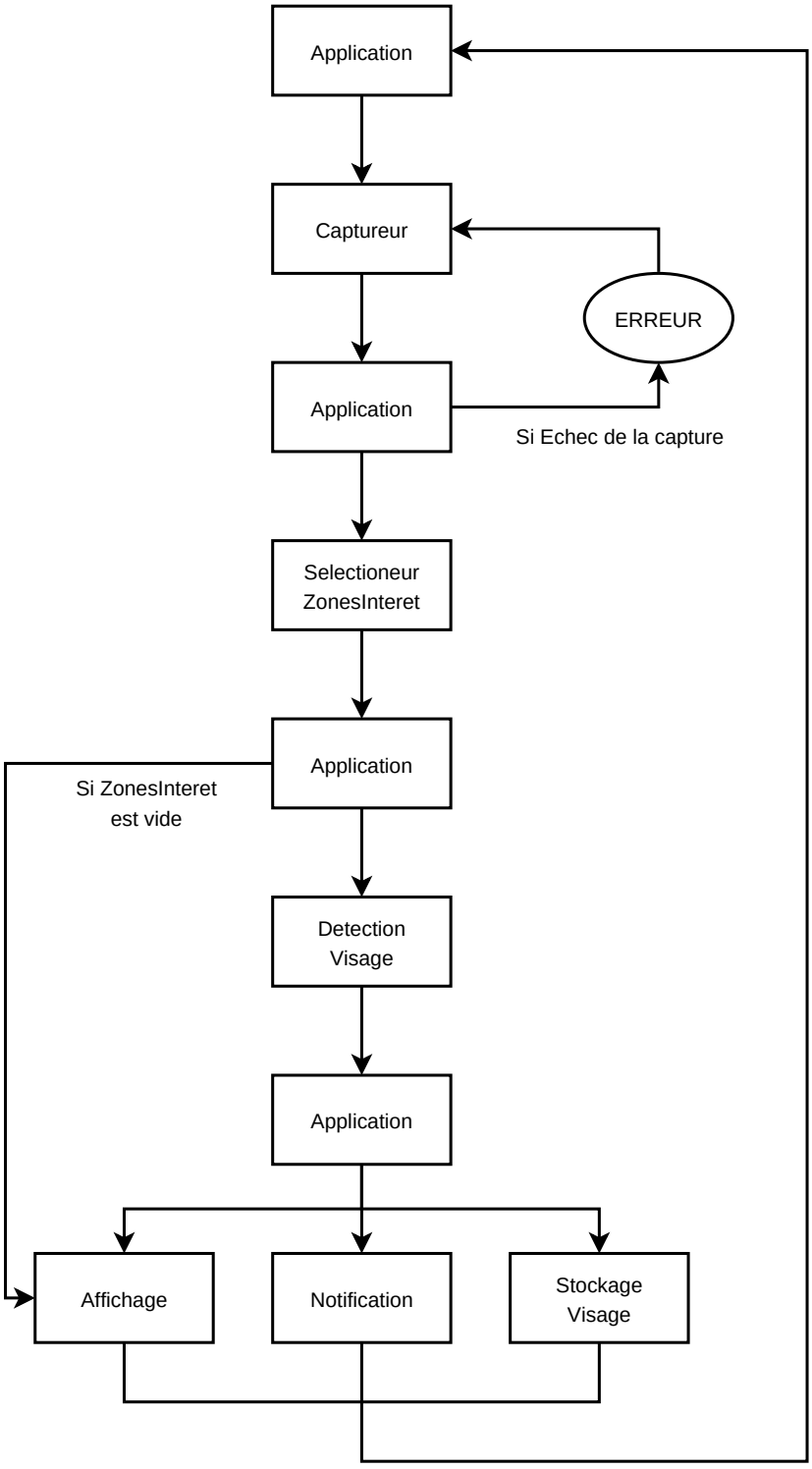

FIG. 7.2 – Diagramme de fonctionnement de notre application

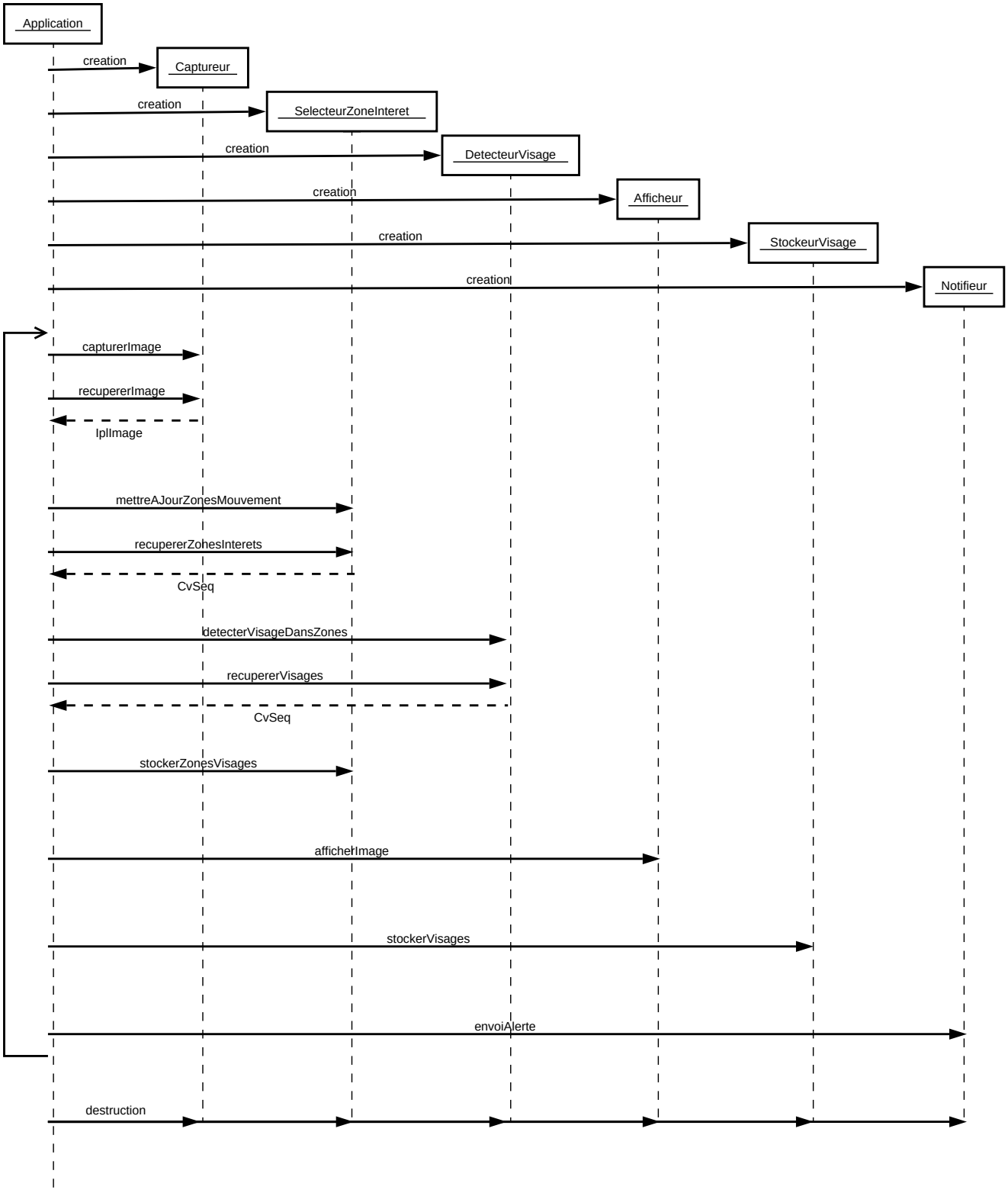

FIG. 7.3 – Diagramme de séquence simplifié de notre application

7.2.5 Affichage

La classe **Afficheur** permet d'afficher l'image capturée et de cibler une ou plusieurs zones, dans notre cas des visages, à l'aide d'un rectangle qui est dessiné autour de celui-ci.

7.2.6 Notifieur

Le notifieur à pour rôle d'informer l'utilisateur. Dans notre cas, lorsqu'un visage est détecté, nous envoyons une alerte à l'utilisateur pour avertir d'une intrusion dans la pièce où se trouve la caméra. Pour cela, un courriel est envoyé à l'utilisateur. Avoir une interface **Notifieur** permet une extensibilité car ici nous implémenterons un **NotifieurCourriel** mais on peut imaginer un NotifieurSMS, NotifieurTelephone, etc...

7.2.7 StockeurVisage

StockeurVisage reçoit des images à sauvegarder sur un serveur sécurisé. La loi française impose une limite dans le temps de conservation des images, cette contrainte devra être gérée par l'utilisateur final. (cf Chapitre sur les lois) **StockeurSSH** implémente une sauvegarde distante utilisant le protocole de transfert SSH, pour sécuriser le transfert.

Chapitre 8

Choix de conception

Dans ce chapitre, nous allons détailler les différents choix de conception pour lesquels nous avons opté afin d'implémenter notre application.

8.1 Introduction

Notre point de départ était une application existante développée par L. Carminati et à l'époque de la rédaction du mémoire intermédiaire, nous avions décidé que la première étape de notre développement serait de faire une refonte du code existant. Lorsque nous nous sommes pleinement lancés dans l'étude du code de cette application, nous avons jugé préférable de ne pas le réutiliser et de repartir sur une base vierge. Les raisons qui nous ont poussé à faire ce choix sont les suivantes :

- L'aspect très "brouillon" du code
- Le trop grand nombre de bibliothèques utilisées
- Le manque de documentation autour de certaines bibliothèques primordiales, comme celle utilisée pour la détection de visages à proprement parler, qui nous aurait pénalisé pour le bon déroulement de l'implémentation
- Le manque de commentaires dans certaines parties importantes du code qui rendait ainsi difficile sa compréhension

Bien que nous ayons décidé de tout réimplémenter, nous nous sommes quand même largement inspiré du fonctionnement de l'application existante pour mener à bien notre développement. Nous nous sommes donc en grande partie inspirée de l'architecture de cette application pour définir la nôtre. Nous avons donc repris l'idée d'intégrer à notre architecture les modules suivants, à savoir un module de capture d'images, un module pour la détection de visages et un module d'affichage des images après traitement. Nous avons également retenu l'idée de faire une détection de mouvement en amont de la détection de visages pour restreindre les zones de recherche sur l'image à analyser. Enfin, nous avons conservé l'idée d'encapsuler le travail de chaque module dans un thread pour optimiser les performances globales de l'application.

Le langage que nous avons retenu pour développer notre application est le C++ en raison de sa rapidité d'éxécution, de sa couche objet qui facilite le découpage modulaire, et du grand nombre de bibliothèques disponibles. Pour mener à bien notre implémentation, nous avons

commencé par effectuer un travail de recherche afin de déterminer les différentes bibliothèques que nous utiliserons. Nous avons retenu les suivantes :

- La bibliothèque OpenCV d'Intel pour tout ce qui concerne la manipulation et l'analyse d'images
- La bibliothèque LibSSH pour implémenter le stockage des images de visage sur une machine distante et de façon sécurisée

Dans la suite, nous allons présenter ces différentes bibliothèques et justifier pourquoi nous avons décidé de les utiliser.

8.2 Présentation de la bibliothèque OpenCV

La bibliothèque OpenCV[1] (Open Source Computer Vision Library) est une bibliothèque gratuite de traitement d'images et de vision par ordinateur proposée par Intel. Elle est de surcroît optimisée et multi-plateforme. Le domaine d'application visé par cette bibliothèque est principalement la vision par ordinateur en temps-réel, comme par exemple les systèmes d'interaction Homme-Machine, l'identification et la reconaissance d'objets, la reconaissance et le suivi de mouvements, la robotique,...

OpenCV regroupe une collection d'opérations bas-niveau aux performances accrues appliquables sur des images. Voici une liste non exhaustive des fonctionnalités proposés par cette bibliothèque :

- Création, allocation et destruction d'images. Macro d'accès rapide aux pixels
- Recherche, manipulation, traitement de contours
- Pyramides d'image
- Analyse morphologique
- Dessins de primitives géométriques (ligne, rectangle, ellipse, polygone, ...)
- Opérations de seuillage (binaire, binaire inversé, à zero, à zéro inversé, ...)
- Calcul de statistiques sur une image (valeur moyenne, mimimum et maximum des pixels, calcul de moments, ...)
- Création et utilisation d'histogramme
- Changement d'espace de couleurs (RGB, HSV, YCrCb, ...)
- Suppression d'arrière plan
- Transformation de distances
- Analyse de mouvement (flot optique, image d'historique de mouvement)
- Détection de visages
- Calibrage de caméras
- Opérateurs d'estimation (Kalman et Condensation)
- Contours actifs
- Algorithme Mean Shift

[1] Le site officiel de la bibliothèque OpenCv est `http ://www.intel.com/technology/computing/opencv/overview.htm`. Elle est disponible en téléchargement gratuit à l'adresse : `http ://sourceforge.net/projects/opencvlibrary/`

- Suivi d'objet 3D avec plusieurs caméras

OpenCV met également à disposition de son utilisateur une large collection de structures de données, ainsi que les opérations qui y sont associées, dont voici quelques exemples :

- Tableaux
- Matrices
- Piles, Files et Listes génériques
- Ensembles
- Graphes
- Arbres

Enfin, la bibliothèque inclue également une sous-bibliothèque nommée HighGui permettant de réaliser facilement une interface utilisateur dont voici les principales fonctionnalités :

- Lecture/écriture d'images sous de nombreux formats (JPEG, PNG, PPM, ...)
- Affichage d'images à l'écran dans une fenêtre
- Capture d'images à la volée directement depuis un fichier vidéo ou une caméra

Les raisons qui nous ont poussé à choisir cette bibliothèque pour implémenter notre application sont multiples. Premièrement parce qu'elle contient des fonctionnalités d'entrée/sortie dont nous avions besoin, à savoir une interface de capture d'images depuis une caméra ainsi qu'une interface pour afficher les images capturées à l'écran. Deuxièmement pour le grand nombre d'opérations de traitement et de manipulation d'images disponibles. Ainsi, OpenCV renferme des implémentations d'algorithmes de détection de mouvement ainsi qu'une implémentation d'un algorithme de détection de visages particulièrement robuste et efficace, ce qui a considérablement renforcé notre intérêt vis à vis de cette bibliothèque. Enfin, et pour revenir à l'algorithme de détection de visages implémenté dans la bibliothèque, nous avons pu nous rendre compte en effectuant des tests, que ce dernier conserve un très bon taux de détection même quand la luminosité est très faible contrairement à celui utilisé dans l'application existante.

8.3 Présentation de la bibliothèque LibSSH

La LibSSH[2] est une bibliothèque C permettant aux programmeurs d'exploiter les fonctionnalités du protocole SSH dans leurs applications comme par exemple éxécuter un programme à distance ou transférer des fichiers de façon sécurisée grâce au protocole SFTP. Les principales fonctionnalités de cette bibliothèque sont listées ci-dessous :

- manipulation d'une connection SSH coté client
- compatibilité avec les protocoles SSH1 et SSH2
- sessions entièrements configurables
- création de serveur
- support des systèmes de cryptage suivants : AES-128, AES-192, AES-256, blowfish, 3des
- support du SFTP

[2]Le site officiel de la bibliothèque est consultable à l'adresse :
`http ://0xbadc0de.be/wiki/doku.php ?id=libssh :libssh`

- support des clés publiques RSA et DSS
- support de la compression (à l'aide de zlib)

Nous avons décidé d'utiliser cette bibliothèque pour implémenter un stockage des images de visages détectés vers une machine distante de façon sécurisée. En effet, au vu des conditions légales et du respect du droit à l'image, nous étions obligé de fournir une telle fonctionnalité dans le cadre d'une application de vidéosurveillance.

Chapitre 9

Description des algorithmes

9.1 Algorithme de Détection de mouvements

Nous allons décrire le fonctionnement de l'algorithme de détection de mouvements dans un flux vidéo implémenté dans la bibliothèque OpenCv. Il repose sur les travaux de J. Davis , G. Bradski et A. Bobick [19] [20]. Le principe est de générer en fonction des images capturées une image d'historique de mouvement qui peut être utilisée pour déterminer rapidement où un mouvement a eu lieu dans l'image et dans quelle direction. Nous allons ainsi détailler les deux procédés que nous utilisons dans notre application, à savoir la génération de l'image historique de mouvement et celui de l'extraction des zones en contenant à partir de cette image.

9.1.1 Génération de l'image d'historique de mouvement

L'image d'historique de mouvement est une image dont les pixels sont à valeurs flottantes. Le principe de l'algorithme de la mise à jour de cet historique est le suivant. Il prend en entrée une image qui vient d'être capturée ainsi que le temps en millisecondes qui s'est écoulé depuis le démarrage de l'application utilisant cet algorithme. On commence par calculer l'image de la différence absolue entre l'image qui vient d'être capturée et celle précedemment capturée. On effectue ensuite un seuillage binaire de cette image pour éliminer les zones qui ont très peu variées entre les deux images capturées. On obtient ainsi une image comportant une ou plusieurs silouhettes d'objets ou de personnes en mouvement. On met alors à jour l'image d'historique de mouvement en fonction du temps d'éxécution de référence ainsi qu'une durée maximale d'historique de mouvement. Le détail de cet algorithme est présenté figure 9.1 page 55.

La figure 9.2 page 56 permet de se faire une idée de l'aspect de l'image d'historique de mouvement générée à partir d'une capture d'images depuis une caméra fixe.

9.1.2 Extraction des zones de mouvement

A partir de l'image d'historique de mouvement, il est facile d'extraire les zones de l'image en contenant. Le principe est de parcourir l'historique de mouvement et de déterminer les zones contenant un grand nombre de pixels ayant des valeurs positives. La figure 9.3 page 56 illustre le résultat de l'application de ce procédé.

Entrées :
- $ImgCap$: une image qui vient d'être capturée
- t_{ref} : le temps en millisecondes qui s'est écoulé depuis le démarrage de l'application utilisant cet algorithme

Données :
- ► $PrecImgCap$: la précédente image capturée
- ► $ValSeuil$: une valeur de seuillage
- ► Ihm : l'image d'historique de mouvement initialisée à zéro au premier appel de l'algorithme
- ► $DureeHistorique$: la durée en secondes de l'historique de mouvement

1. Calculer l'image $Silh$, différence absolue des images $ImgCap$ et $PrecImgCap$:
   Pour $x = 1, \ldots, L$, L étant la largeur de l'image capturée faire
   Pour $y = 1, \ldots, H$, H étant la hauteur de l'image capturée faire
   $Silh(x,\ y) = abs(ImgCap(x,\ y) - PrecImgCap(x,\ y))$
2. Effectuer un seuillage binaire de l'image $Silh$ pour éliminer les zones de l'image qui ont très peu variées entre deux captures :
   Pour $x = 1, \ldots, L$ faire
   Pour $y = 1, \ldots, H$ faire
   Si $Silh(x, y) > ValSeuil$ alors $Silh(x, y) = 1$
   Sinon $Silh(x, y) = 0$
3. Mettre à jour l'image d' historique de mouvement :
   Pour $x = 1, \ldots, L$ faire
   Pour $y = 1, \ldots, H$ faire
   Si $Silh(x, y) > 0$ alors $Ihm(x, y) = t_{ref}$
   Sinon Si $Silh(x, y) == 0$ et $Ihm(x, y) < t_{ref} - DureeHistorique$ alors $Ihm(x, y) = 0$
   Sinon $Ihm(x, y)$ n'est pas modifiée
4. $PrecImgCap = ImgCap$ et réitérer ce processus avec la prochaine image capturée

FIG. 9.1 – Algorithme de génération d'une image d'historique de mouvement à partir d'images capturées à intervalle régulier

FIG. 9.2 – A gauche, la fenêtre affichant les images capturées. A droite, celle affichant l'historique de mouvement. Un mouvement de gauche à droite a été effectué et est visible sur l'historique de mouvement. Les zones les plus ternes correspondent aux silouhettes de mouvement les plus anciennes dans le temps, les plus lumineuses aux plus récentes.

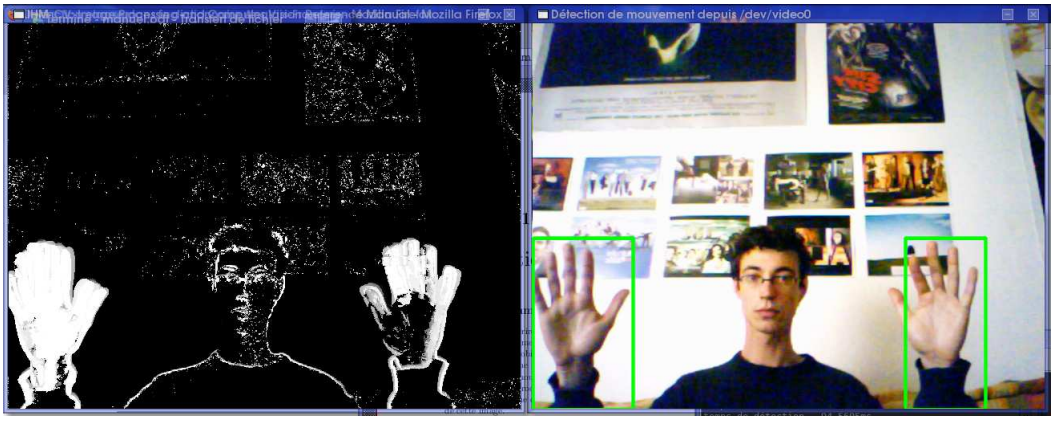

FIG. 9.3 – Illustration de l'extraction des zones de mouvement à partir de l'image d'historique de mouvement. Du mouvement est généré en agitant les mains. On peut voir que les zones de mouvement extraites sont celles de l'historique contenant un très grand nombre de pixels à valeurs positives.

9.2 Algorithme de Détection de visages

Nous allons maintenant décrire en détail l'algorithme au centre de notre application, soit celui de détection de visages implémenté dans la bibliothèque OpenCV. Cet algorithme permet en fait de détecter n'importe quel type d'objets dans une image mais il est particulièrement adapté à la détection de visages. Il a été initialement élaboré par P. Viola et M. Jones [21] puis amélioré et implémenté par R. Lienhart [22]. Notre application reposant principalement sur l'utilisation de leur algorithme, il nous semblait important de faire un état de l'art de leurs travaux.

Cet algorithme de détection de visages est en réalité un algorithme de classification binaire qui analyse différentes zones d'une image et qui classe ces dernières suivant deux classes : la classe des images représentant l'objet que l'on veut détecter et celle des images ne le représentant pas, de la même manière que les machines à vecteur de support présentées dans l'un des premiers chapitres de ce mémoire. Pour mener à bien une détection d'objets avec cet algorithme, on doit en premier lieu entraîner un classifieur à partir d'un ensemble d'images représentant l'objet à détecter (par exemple une voiture ou dans notre cas un visage), appelées exemples positifs, et d' un ensemble d'images arbitraires appelées exemples négatifs. Tous ces exemples doivent avoir exactement la même taille (par exemple 20x20 pixels) et leur nombre doit être conséquent (de l'ordre de plusieurs milliers).

OpenCV est livré avec plusieurs classifieurs (sous la forme de fichiers XML) permettant d'effectuer une détection de visages dans une image. Chaque classifieur est en réalité une "cascade de classifieurs dopés basés sur les descripteurs de Haar" (*cascade of boosted classifiers working with Haar-like features*). Le terme "cascade" signifie que le classifieur est constitué de plusieurs classifieurs simples (appelés étages) qui sont appliqués les uns à la suite des autres sur une région d'intérêt d'une image. Le terme "dopé" (*boosted*) signifie que les classifieurs de chaque étage de la cascade sont égalements construits à partir de plusieurs classifieurs, dits faibles ou basiques, à l'aide de techniques d'apprentissage basées sur une combinaison de décisions.
Dans cette étude, nous commencerons par introduire le concept d'apprentissage supervisé par combinaison de décisions (le *boosting*). Nous verrons ensuite comment est réalisé le détecteur de visages à l'aide de cette technique en utilisant les fameux descripteurs de Haar (*Haar-like features*). Nous verrons ensuite le fonctionnement général de l'algorithme de détection.

9.2.1 L'apprentissage par combinaison de décisions : le *boosting* (dopage)

Principe

Cette section a été réalisée avec l'aide du chapitre 11 du livre de A. Cornuéjols et L. Miclet sur l'apprentissage artificiel [23]. Le *boosting* est un domaine de l'apprentissage automatique supervisé (une branche de l'intelligence artificielle). Le principe est de combiner plusieurs classifieurs faibles afin d'obtenir un classifieur fort. Ce concept a été introduit par R. Schapire [24] au début des années 90. On rapelle qu'un classifieur (également appelé hypothèse) est une fonction séparant un ensemble de données en plusieurs classes (usuellement deux). Un classifieur est qualifié de faible s'il est capable de reconnaître deux classes au moins aussi bien que ne le ferait le hasard, c'est à dire qu'il ne se trompe pas plus d'une fois sur deux en moyenne. On qualifie un classifieur de fort s'il est très précis, c'est à dire que les erreurs de classification

sont rares. Pour mener, à bien cette tâche, on utilise ce que l'on appelle un apprenant faible qui est un algorithme fournissant un classifieur faible en fonction d'un ensemble d'exemples d'apprentissage et d'une distribution de probabilité sur ces exemples.

Le fonctionnement général d'un algorithme de *boosting* est le suivant. De façon itérative, on va construire un ensemble de N classifieurs complémentaires à partir d'un ensemble d'exemples d'apprentissage distribués selon une loi de probabilité. A chaque itération, un classifieur faible sur ces exemples est fourni par l'apprenant faible suivant la distribution de probabilité. Ce dernier est alors pondéré par la qualité de sa classification globale : plus il classe bien, plus il aura de l'importance dans le classifieur final. Les exemples d'apprentissage sont alors repondérés de manière a ce que ceux qui ont été mal classés soit "boostés" en importance pour qu'à la prochaine itération, l'apprenant faible en tienne compte pour générer un classifieur faible adéquat. Le classifieur final revient à une combinaison linéaire pondérée des classifieurs faibles déterminés à chaque itération.

L'algorithme de *boosting* le plus connu s'appelle AdaBoost (pour Adaptive Boosting) et c'est une de ses variantes qui a été utilisée pour générer le classifieur utilisé par le détecteur de visages d'OpenCV.

L'algorithme AdaBoost

AdaBoost est un algorithme élaboré par Y. Freund et R. Schapire en 1995 [25]. C'est le plus populaire et le plus important historiquement des algorithmes de boosting. Le détail de cet algorithme est présenté à la figure 9.4 page 59.

9.2.2 Le détecteur de visages de la bibliothèque OpenCV

Le détecteur de visages de la bibliothèque OpenCV est basé sur les travaux de recherches de P. Viola et M. Jones [21] et a été amélioré et implémenté par R. Lienhart [22].C'est le plus populaire actuellement. Il consiste en une cascade de classifieurs "boostés". Les classifieurs de chaque étage de cette cascade ont été construits à partir d'une combinaison de classifieurs faibles basés sur des descripteurs de Haar (Haar-like features). Nous commencerons donc par présenter les descripteurs de Haar. Nous verrons ensuite l'algorithme qui a été utilisé pour construire la cascade de classifieurs à l'aide de ces derniers.

Les descripteurs de Haar

Un descripteur de Haar permet d'extraire des informations d'une région d'une image. Le principe de ce type de descripteur est de calculer une combinaison pondérée de sommes de valeurs de pixels contenus dans des régions rectangulaires d'une image. Il existe 3 types principaux de descripteurs de Haar représenté sur la figure 9.5 page 60 :

- Les descripteurs type "deux rectangles" (exemples (A) et (B) de la figure 9.5 page 60) qui calculent la différence de la somme des valeurs d'intensité de pixels de deux régions rectangulaires adjacentes de l'image.
- Les descripteurs type "trois rectangles" (exemple (C) de la figure 9.5 page 60) qui calculent la différence entre la somme des valeurs d'intensité des pixels de deux régions rectangulaires et celle d'une troisième adjacente aux deux premières.

Entrées :
- Un ensemble de N exemples d'apprentissages étiquetés $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ avec $y_i = \{+1, -1\}$
- Une distribution de probabilité D sur les N exemples
- Un apprenant faible **ApprFaible**
- Un nombre d'itérations T

Initialisation : $D_1(i) = D(i)$ pour $i = 1, \ldots, N$
Pour $t = 1, \cdots, T$ faire :

- Apprendre un classifieur faible h_t grâce à l'algorithme **ApprFaible** en fonction de la distribution de probabilité D_t
- Calculer l'erreur apparente ϵ_t de h_t sur S :

$$\epsilon_t = Pr_{D_t}(h_t(x_i) \neq y_i) = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{N} D_t(i)$$

- Calculer $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
- **Pour** $i = 1, \ldots, m$ faire :
  - $D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{-\alpha_t}$ si $h_t(x_i) = y_i$ (i.e. si h_t classe bien x_i)
  - $D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{+\alpha_t}$ si $h_t(x_i) \neq y_i$ (i.e. si h_t classe mal x_i)

  (Z_t est une valeur de normlisation telle que $\sum_{i=1}^{N} D_{t+1}(x_i) = 1$)

Sortie : Un classifieur fort $H(x) = signe(\sum_{t=1}^{T} \alpha_t(x) h_t(x))$

FIG. 9.4 – Détail de l'algorithme AdaBoost

- Les descripteurs “quatre rectangles” (exemple (D) de la figure 9.5 page 60) qui calculent la différence de la somme des valeurs d'intensité des pixels de deux paires diagonales de rectangles.

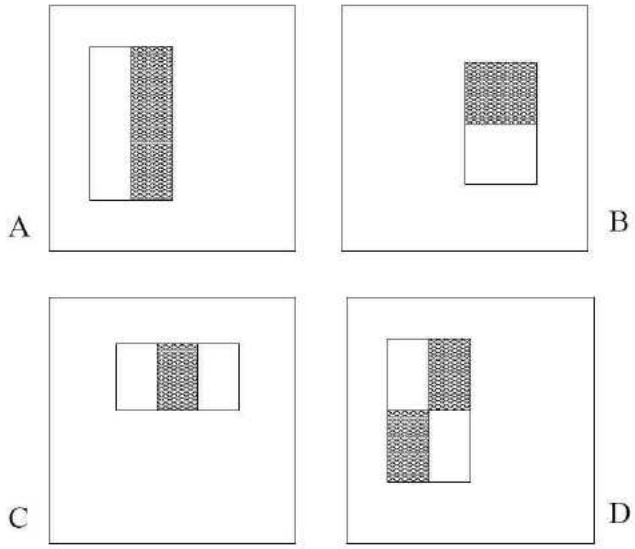

FIG. 9.5 – Exemples de descripteurs de Haar encapsulés dans une fenêtre de détection. La somme des valeurs d'intensité de pixels contenus dans les régions rectangulaires blanches est soustraite à celle des régions grisées.

L'intérêt d'utiliser ce type de descripteurs est que leur calcul est très rapide à effectuer en utilisant une représentation intermédiaire de l'image traitée appelée “image intégrale”. La valeur de l'image intégrale au pixel de coordonnées (x, y) est la somme des valeurs des pixels situés dans la zone au dessus à gauche (voir figure 9.6 page 61) :

$$ii(x, y) = \sum_{\substack{x' \leq x \\ y' \leq y}} i(x', y')$$

où $ii(x, y)$ est la valeur de l'image intégrale et $i(x, y)$ la valeur d'intensité du pixel (x, y) de l'image originale.

En utilisant la paire de récurrence suivante :

$$s(x, y) = s(x, y - 1) + i(x, y)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

(où $s(x, y)$ est la valeur de la somme des intensités des pixels de la ligne d'indice x jusqu'à la colonne d'indice y, $s(x - 1, y) = 0$ et $ii(-1, y) = 0$), on peut calculer la valeur de l'image intégrale en un passage sur l'image originale.

En stockant les valeurs de l'image intégrale dans un tableau, toute somme de valeurs d'intensité de pixels d'une zone rectangulaire de l'image originale peut être calculé avec au

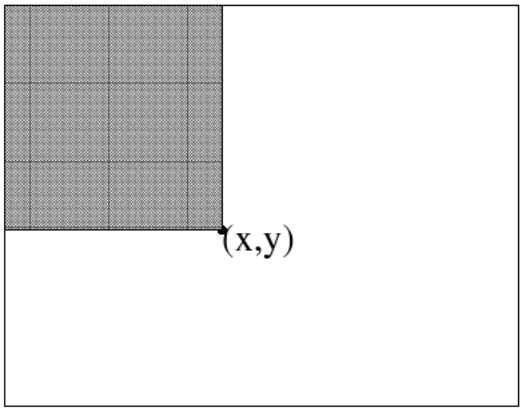

FIG. 9.6 – La valeur de l'image intégrale au point (x, y) est la somme de tous les pixels contenues dans la zone grisée

plus 4 valeurs du tableau (voir la figure 9.7 page 61). Comme les descripteurs de Haar consistent à calculer des différences de sommes de valeurs d'intensité de pixels de régions rectangulaires adjacentes, leur calcul est très simple à effectuer. Ainsi, la valeur d'un descripteur de type "deux rectangles" peut être calculé avec 6 valeurs du tableau, un descripteur de type "trois rectangles" avec 8 et un descripteur de type "quatre rectangles" avec 9.

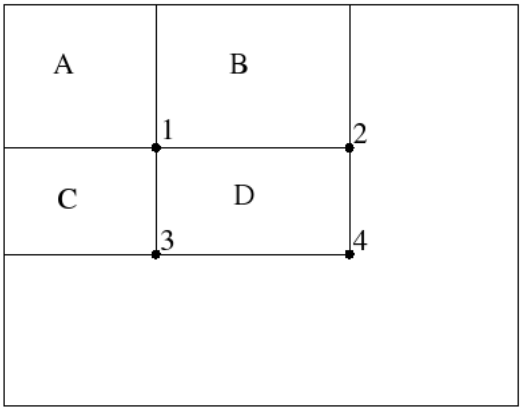

FIG. 9.7 – La somme des pixels du rectangle D peut être calculée avec quatre valeurs du tableau stockant celles de l'image intégrale. La valeur de l'image intégrale à la position 1 est la somme des pixels du rectangle A. La valeur à la position 2 est $A + B$, à la position 3 $A + C$ et à la position 4 $A + B + C + D$. La somme des pixels du rectangle D est obtenue en calculant 4 + 1 - (2 + 3).

Les prototypes des descripteurs de Haar utilisés dans l'algorithme de détection de visages d'OpenCV sont présentés à la figure 9.8 page 62. Ils sont au nombre de 14 et sont des types "deux rectangles" et "trois rectangles". Les versions des descripteurs orientés de 45 ° par rapport à ceux présentés plus haut ont été ajoutés par R. Lienhart [22] quand il a repris les travaux de P.

Viola et M. Jones[21]. Il a élaboré un algorithme permettant de calculer leur valeur facilement, toujours à partir des images intégrales, que nous ne détaillerons pas. Ces prototypes permettent de créer une grande batterie de descripteurs (de l'ordre de plusieurs dizaines de milliers) en faisant varier leur position dans la fenêtre de détection ainsi que leur taille horizontalement et verticalement. La taille de la fenêtre de détection est égale à la taille des images qui seront utilisées comme exemples d'apprentissage dans le processus de création du classifieur que nous allons maintenant détailler.

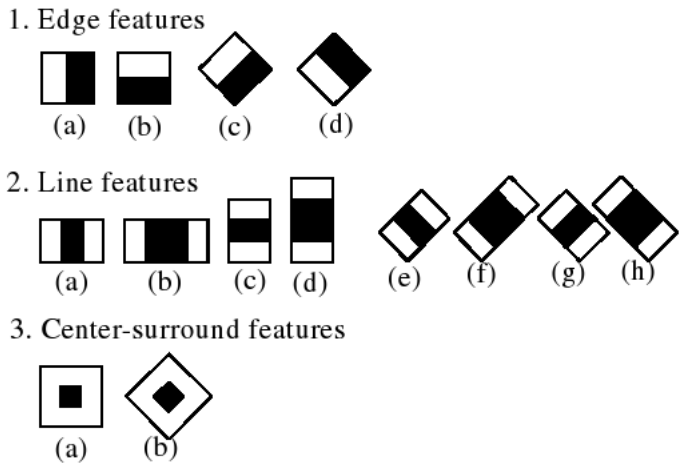

FIG. 9.8 – Prototypes des descripteurs de Haar utilisés dans la construction du classifieur utilisé par le détecteur de visages d'OpenCV. Les zones blanches ont des poids positifs et les zones noires des poids négatifs

Construction d'une cascade de classifieurs

Nous allons maintenant expliquer le processus d'apprentissage qui a été mis en oeuvre pour construire le classifieur utilisé par le détecteur de visages. Comme indiqué plus haut, il consiste en une cascade de classifieurs "boostés" qui peut être assimilé à un arbre de décision. Cette cascade a été construite à l'aide d'un vaste ensemble d'exemples d'apprentissage, à savoir des images de taille fixe (de l'ordre de 20x20 pixels) séparées en deux classes. La classe des exemples positifs contenant toutes les images représentant un visage de face(la figure 9.9 page 63 présente un échantillon des images de cette classe) et celles des exemples négatifs contenant celles n'en représentant pas un.

Le principe de l'algorithme d'apprentissage est le suivant. A chaque étage, un classifieur utilisant les descripteurs de Haar présentés dans la section précédente est entrainé à partir des exemples d'apprentissage de telle sorte que le taux minimum de bonnes détections soit égale à une certaine valeur f et que le taux maximum de fausses détections soit égal à une certaine valeur d. Le taux de bonnes détections est le pourcentage des exemples positifs qui ont bien été classés. Le taux de fausses détections est le pourcentage des exemples négatifs qui ont été classés comme positifs. Le classifieur de l'étage suivant sera alors entrainé avec la totalité des exemples positifs ainsi que le sous-ensemble des exemples négatifs qui ont mal été classés par le classifieur qui vient d'être appris. Ce processus est schématisé sur la figure 9.10 page 64.

FIG. 9.9 – Echantillon d'exemples d'apprentissage positifs utilisés pour l'entrainement de la cascade de classifieurs

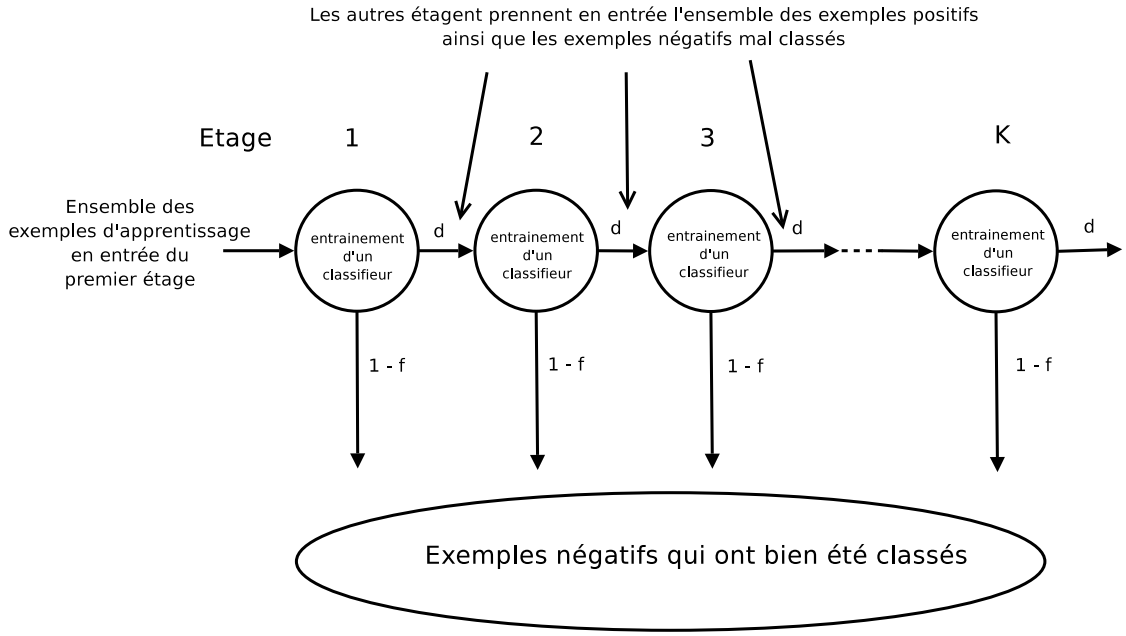

FIG. 9.10 – Processus d'apprentissage d'une cascade de classifieurs de K étages. A chaque étage, un classifieur est entrainé de telle sorte qu'il ait un taux minimum de bonnes détections d et un taux maximum de fausses détections f

Au final, si la cascade comporte K étages, le taux de bonnes détections D du classifieur global sur les exemples d'apprentissage sera au moins égal à :

$$D = \prod_{i=1}^{K} d_i$$

où d_i est le taux de bonnes détections du ième classifieur de la cascade sur l'ensemble des exemples à partir duquel il a été entrainé. Le taux de fausses détections F quant à lui sera au plus égal à :

$$F = \prod_{i=1}^{K} f_i$$

où f_i est le taux de fausses détections du ième classifieur de la cascade sur les exemples à partir duquel il a été entrainé.

Le détail de l'algorithme d'apprentissage utilisé pour la construction de la cascade est présenté à la figure 9.11 page 65. Pour la réalisation du classifieur utilisé couramment dans la bibliothèque OpenCV, le taux de bonnes détections minimum par étage a été fixé à 99.9% et celui maximum de fausses détections à 50%. La cascade résultante comporte 20 étages et le temps de calcul nécessaire à son apprentissage fut de l'ordre de plusieurs semaines.

L'étape la plus important de l'algorithme d'apprentissage de la cascade est bien sûr l'entrainement à chaque étage d'un classifieur basé sur des descripteurs de Haar grâce à AdaBoost.

- On fixe les valeurs f, le taux acceptable maximum de fausses détections par étage et d le taux acceptable minimum de bonnes détection par étage.
- On fixe $F_{attendu}$, le taux maximum de fausses détection que l'on veut avoir au final
- P = ensemble des exemples d'apprentissage positifs
- N = ensemble des exemples d'apprentissage négatifs
- $F_0 = 1.0$; $D_0 = 1.0$
- $i = 0$
- **Tant que** $F_i > F_{attendu}$ faire
  - $i \leftarrow i + 1$
  - $n_i \leftarrow 0; F_i = F_{i-1}$
  - **Tant que** $F_i > f \times F_{i-1}$ faire
    * $n_i \leftarrow n_i + 1$
    * Utilser P et N pour entrainer un classifieur utilisant n_i descripteurs de Haar à l'aide de l'algorithme AdaBoost
    * Tester la cascade de classifieurs courante sur l'ensemble des exemples d'apprentissage pour déterminer F_i et D_i
    * Ajuster le classifieur obtenu de telle sorte que le taux de bonnes détections de la cascade de classifieurs courante soit au moins égal à $d \times D_{i-1}$ (cela aura également pour effet de modifier la valeur de F_i)
  - $N \leftarrow \emptyset$
  - **Si** $F_i > F_{attendu}$ alors tester la cascade de classifieur courante sur l'ensemble des exemples négatifs et mettre dans l'ensemble N ceux qui sont faussement détectés (i.e. qui sont classés comme étant positifs)

FIG. 9.11 – Algorithme d'apprentissage utilisé pour construire une cascade de classifieurs

La construction d'un classifieur binaire à partir d'un descripteur de Haar est très simple. Si on désigne par f la fonction calculant la valeur d'un descripteur de Haar sur une image x et T une valeur de seuillage, alors le classifieur h associé est :

$$h(x) = \begin{cases} +1 \text{ si } f(x) > T \\ -1 \text{ sinon} \end{cases}$$

Ainsi, pour chaque étage de la cascade, un algorithme d'apprentissage basé sur AdaBoost construit un classifieur utilisant un nombre fixé de descripteurs de Haar. Dû aux grands nombres de descripteurs disponibles, cet algorithme d'apprentissage effectue également une sélection de ceux qui permettront d'effectuer la classification avec le taux d'erreur le plus bas. Le détail de cet algorithme est présenté figure 9.12 page 66.

Entrées :
- Un ensemble de N exemples d'apprentissage $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ où $y_i \in \{+1, -1\}$ (exemples positifs et négatifs)
- Un nombre T de descripteurs de Haar à utiliser pour construire le classifieur

- Initialiser le poids des exemples : $\omega_{i,1} = \frac{1}{2m}, \frac{1}{2l}$ pour $y_i = +1, -1$ respectivement, avec m le nombre d'exemples positifs et l le nombre d'exemples négatifs
- **Pour** $t = 1, \ldots, T$ faire
  1. Normaliser les poids pour que ω_t devienne une distribution de probabilité :
  $$\omega_{t,i} \leftarrow \frac{\omega_{t,i}}{\sum_{j=1}^{N} \omega_{t,j}}$$
  2. Pour chaque descripteur de Haar j disponible, entrainer un classifieur h_j et évaluer l'erreur de classification globale ϵ_j en fonction de la distribution ω_t :
  $$\epsilon_j = \sum_{\substack{1 \leq i \leq N \\ h_j(x_i) \neq y_i}} \omega_{t,i}$$
  3. Sélectionner le classifieur h_t avec l'erreur de classification ϵ_t la plus faible
  4. Mettre à jour le poids des exemples :
  $$\omega_{t+1,i} = \omega_{t,i} \times \beta_t^{1-e_i}$$
  où $e_i = 0$ si x_i est classé correctement, $e_i = 1$ sinon et $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

Sortie : Le classifieur global suivant :

$$h(x) = \begin{cases} 1 \text{ si } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ -1 \text{ sinon} \end{cases}$$

FIG. 9.12 – Algorithme d'apprentissage basé sur AdaBoost utilisé pour entrainer les classifieurs de chaque étage de la cascade

Afin de donner un exemple concret de l'utilisation des descripteurs de Haar dans un processus de détection de visages dans une image, présentons ceux qui ont été sélectionnés pour

construire le classifieur du premier étage de la cascade. Ce dernier utilise deux descripteurs et obtient un taux de 100% de bonnes détections et un taux de 40% de fausses détections. Le premier descripteur sélectionné utilise la propriété que la région des yeux est souvent plus sombre que celle du nez et des joues. Le second se base sur la propriété que les yeux ont une couleur plus sombre que le nez. Ces deux descripteurs sont présentés à la figure 9.13 page 67. Les autres étages de la cascade utilisent beaucoup plus de descripteurs dû au fait que la classification à effectuer est plus complexe.

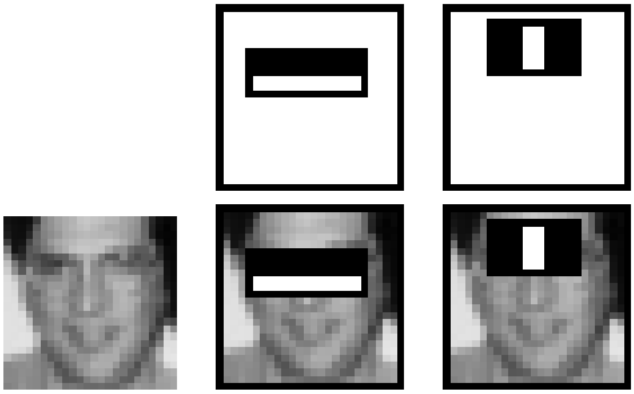

FIG. 9.13 – Les deux descripteurs de Haar utilisés pour constuire le classifieur du premier étage de la cascade. Le premier mesure la différence d'intensité entre la région des yeux et celle du nez et des joues. Le second compare l'intensité des régions des yeux et celle de la région du nez.

9.2.3 Fonctionnement de l'algorithme de détection

Le fonctionnement global de l'algorithme de détection est le suivant. Etant donné une image en entrée, une fenêtre de détection va alors parcourir l'image sous plusieurs échelles et la sous-image contenue dans cette fenêtre est analysée par la cascade de classifieurs. Si le premier étage de la cascade indique que cette image pourrait représenter un visage, il déclenche l'analyse de cette dernière par le second étage de la cascade. Et ainsi de suite jusqu'à ce que l'image ait traversée tous les étages (elle est alors reconnue comme étant un visage) ou qu'elle soit rejetée à un certain étage (elle n'est pas reconnue comme étant un visage). Ce fonctionnement est schématisé sur la figure 9.14 page 68.

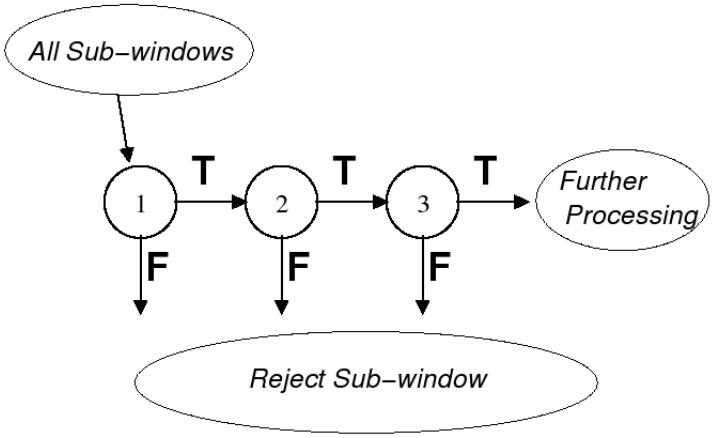

FIG. 9.14 – Schéma de fonctionnement de l'algorithme de détection de visages. L'ensemble des sous-fenêtres de l'image traitée qui traversent l'ensemble des étages de la cascade de classifieurs sont identifiées comme étant un visage. Sinon ils sont rejetés

Chapitre 10

Résultats des tests de validation et de fonctionnement

Nous allons vous montrer les différences entre le logiciel de départ implémenté par Mr Carminati et notre logiciel. Ainsi nous verrons les conditions limites d'utilisation de ce logiciel. Pour cela, nous allons réaliser les tests de validation et de fonctionnement à l'aide des tests "système", de tests en "boîte noire" et pour terminer un test en "boîte blanche".
Tous ces tests ont été réalisé depuis la salle visu du labri, avec les caméras Axis 213 PTZ. De plus, elles disposent d'un projecteur de lumière infrarouge pour la vision de nuit. Par ailleurs, la résolution des images est de 640*480 pour notre logiciel contre une résolution de seulement 384*288 pour celle de Mr Carminati. Enfin, les tests ont été réalisé dans l'état actuel de la caméra sans que le zoom ou d'autres options ne soient modifiées.
Lors de nos illustrations, la mention "taux de détection de visages" apparaîtra souvent sur les graphiques. Cette notion représente la proportion en pourcentage de visages détectés. Anisi, un taux de 50% montre que la détection de visages se fait une fois sur deux environ et un taux de 100% : une détection systématique. Enfin nos images sont des images miroirs de la réalité, ce qui n'a aucune incidence sur la détection.

10.1 Les tests de validation et de fonctionnement

10.1.1 Les tests "système"

Les tests "système" vont nous aider à vérifier globalement la fiabilité et les performances du logiciel en ce qui concerne les besoins fonctionnels.

Test de la luminosité :

La luminosité est le premier point des besoins fonctionnels à traiter. Au travers d'images, de graphes et d'analyses, nous pouvons distinguer de nombreuses différences obtenues par le nouveau logiciel et celui mis en place jusqu'alors. Nous avons tester le comportement des deux logiciels avec une lumière uniformément répartie dans la salle mais aussi en cas de contrastes saisissants.

Une luminosité uniforme dans la salle : La figure 10.1 présentée page 70 est un graphe qui montre de grosses différences de performance. Nous avons utilisé le trait plein pour repré-

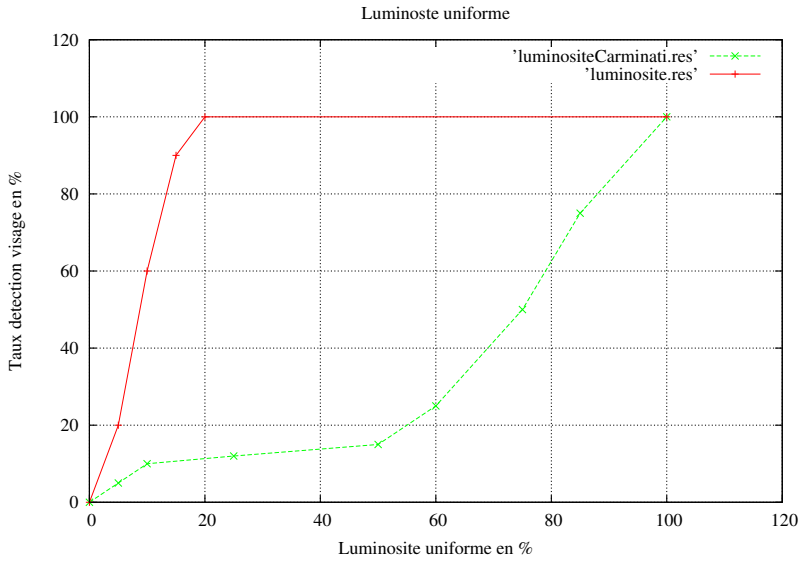

FIG. 10.1 – Graphe de luminosité uniformement répartie dans la salle

senter les performances de notre logiciel et le trait discontinu pour celles de Mr Carminati. Notre logiciel permet d'avoir une détection de visages de plus en plus fiable et au fur et à mesure de l'importance de la luminosité émise.
Pour que tous les visages de face présents dans le champ de la caméra soient détectés, il faut une luminosité minimale d'environ 20% dans la pièce contre une luminosité totale de 100% avec l'ancien logiciel !
Passons aux résultats visuels des applications.
En premier lieu, figure 10.2, page 71 nous avons des images capturées en pleine lumière dont une, prise avec l'application de Mr Carminati à gauche et une prise avec notre logiciel à droite. Nous pouvons constater qu'avec une luminosité maximale, la detection de visages se déroule parfaitement bien dans les deux cas.

En second lieu, figure 10.3, page 71 nous avons des images prises avec la plus faible luminosité possible dont une, prise avec l'application de Mr Carminati à gauche et une prise avec notre logiciel à droite. Nous pouvons constater qu'avec une trés faible luminosité, notre logiciel détecte parfois un visage alors qu'avec l'ancien logiciel, on ne detecte quasiment plus rien, si pas du tout.

Une luminosité non uniforme : Nous avons testé le comportement des logiciels avec une forte luminosité en avant plan et de l'obscurité en arrière plan et vice-et-versa. Nous avons fait ces tests en inversant progressivement la luminosité et nous arrivons à la figure 10.4, page 72.
"-50" est un chiffre arbitraire représentant une forte luminosité en avant plan seulement, "0" à une lumière uniforme dans la pièce et enfin "50" une forte luminosité en arrière plan.
Notre application "souffre" trés peu des changements de luminosité, la détection de visages est d'environ 75% dans le pire des cas ce qui reste une donnée fiable. Par contre, celle du logiciel

FIG. 10.2 – Images capturées en pleine lumière

FIG. 10.3 – Images prisent en basse lumière

existant montre des faiblesses évidentes.
Pourtant, il reste une question : Que se passe t-il en cas d'obscurité totale de la salle ?
Notre logiciel est incapable de détecter un visage en cas d'obscurité mais il est pourtant efficace dans la détection de mouvements s'ils ont lieu : C'est à dire qu'il est capable de les cibler : voir figure 10.5, page 73.
Ceci donne lieu à des extensions possibles du logiciel décrites au chapitre 11, page 78.

Test du suivi des visages

Dans l'application de Mr Carminati, suite à une détection de mouvement dans le champ visuel de la caméra, un visage peut être détecté. Le problème est qu'en absence de mouvement, le visage "présent" n'est plus détecté.
Par conséquent, nous avons implémenté un suiveur permettant de corriger ce problème. Anisi, lorsqu'un visage est détecté, il le sera jusqu'à ce qu'il quitte le champ de la caméra ou qu'il soit dans les conditions limites d'utilisation décritent page 75.
Ainsi nous obtenons à la figure 10.6, page 73 une illustration des prises possibles, avec à gauche

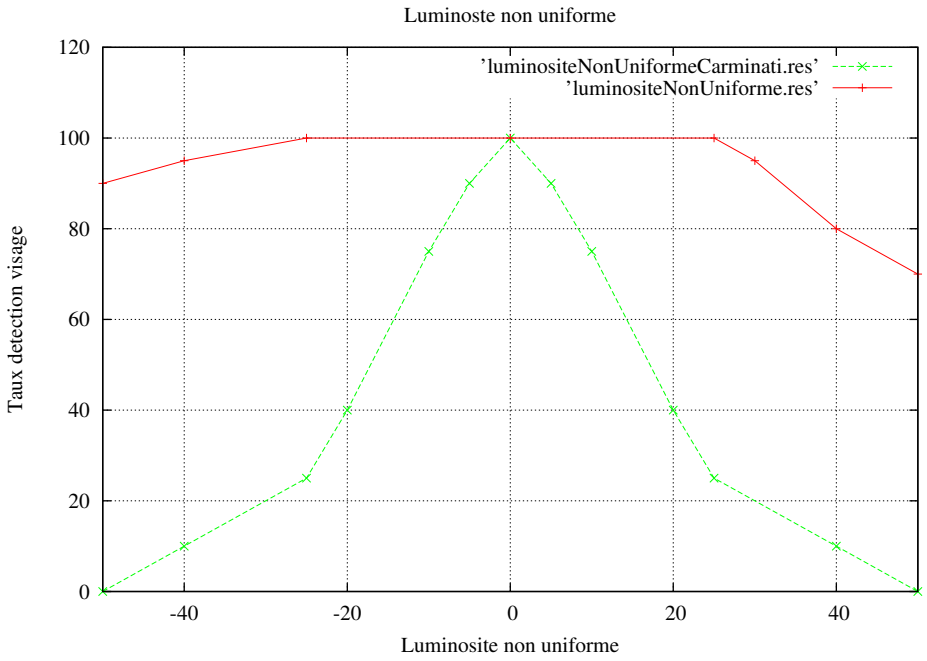

FIG. 10.4 – Graphe d'une luminosité non uniformément répartie dans la salle

le résultat de l'application existante et à droite, la nôtre.

Test de la notification d'alerte lors de la détection d'une intrusion

En rapport avec le code pénal 226-17, le traitement automatisé d'informations doit être à accès sécurisé. C'est pourquoi, nous sauvegardons une image dont un exemple est disponible figure 10.7, page 74 sur une installation sécurisée.
Nous envoyons de plus, un courriel dont l'adresse du destinataire est, entre autres, paramétrable, qui aura l'aspect de la figure 10.8, page 74.

10.1.2 Les tests en "boîte noire"

Nous avons effectué des tests en "boîte noire". C'est à dire, nous ne nous préoccupons que des résultats et non pas de comment ils ont été obtenu. Nous nous attachons à tester les besoins non-fonctionnels du cahier des charges.

Test sur la détection en temps réel

Notre application, malgré le rajout de nombreuses fonctionnalités, devrait être aussi bonne que celle de Mr Carminati en ce qui concerne le traitement des images en temps réel. Nous définissons le traitement d'images en temps réel comme un traitement de 25 images par seconde. Suite au cahier des charges établi avec notre client, nous considérons que nous sommes en temps réel si nous traitons 10 images par seconde dans le pire des cas.
Nous avons donc sur la figure 10.9 à la page 75 un graphe montrant la durée de traitement d'une image (en ms) en fonction de l'activité dans le champ de la caméra.

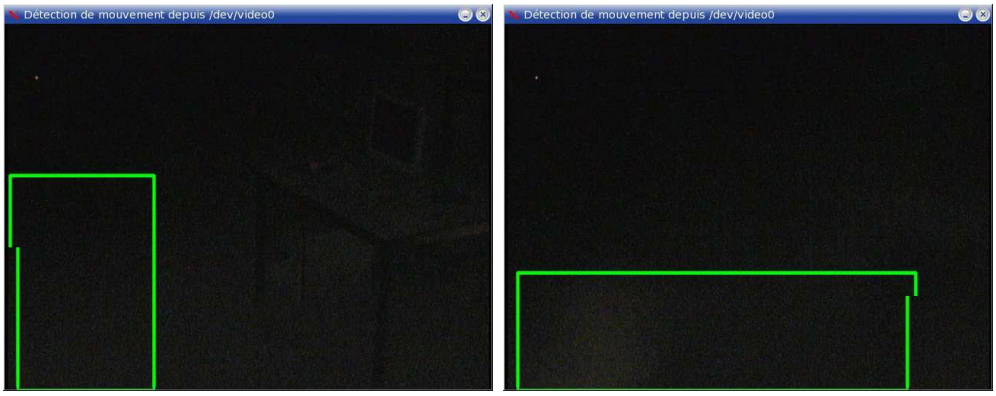

FIG. 10.5 – Images de la détection de mouvement dans l'obscurité

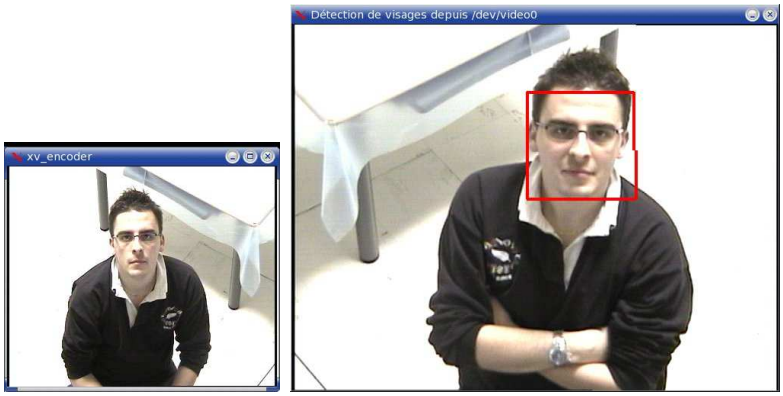

FIG. 10.6 – Suivi de visage

Nous voyons que plus il y a d'activités dans le champ de la caméra plus le traitement serait long. Ce qui compte est l'activité c'est à dire le mouvement qu'il y a dans ce champ.
En effet, le rajout de visages à détecter ou la variation de la luminosité modifie très sensiblement le temps de traitement de l'image.
Dans le cas où il n'y a aucun mouvement, il n'y a pas de traitement qui s'effectue donc nous sommes dans un temps de traitement de 0 ms. Dans une moyenne d'activité de 50% dans le champ de la caméra, nous sommes dans des performances de l'ordre de 40 ms. Plus précisement, le logiciel traite environ 25 images par seconde ! ! !
Enfin lorsqu'il y a du mouvement dans tout le champ de la caméra, toute l'image est traité afin de détecter d'éventuel visage. Nous sommes alors dans des temps de traitement de l'ordre de 85 ms : Le logiciel est capable de traiter un peu moins de 12 images par seconde.
Nous sommes parfaitement en accord avec le cahier des charges.
Cependant, il est important de noter que le suivi d'un visage fixe prend 5 à 6 ms pour traiter l'image.

FIG. 10.7 – Image sauvegardé par le stockeur

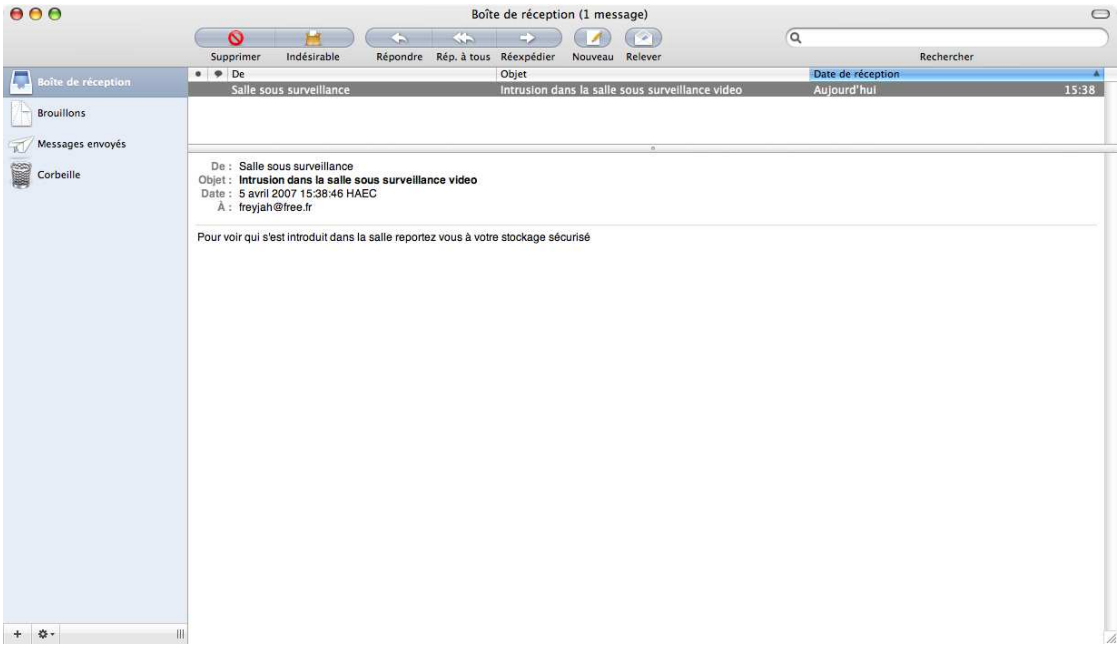

FIG. 10.8 – Courriel envoyé lors d'intrusion

Test de gestion de plusieurs flux vidéo

Ce besoin non-fonctionnel a été traité mais pour des raisons techniques, nous ne pouvons pas avoir de tests certifiant nos avancés car la deuxième caméra en place est défectueuse. Ainsi, nous avons testé ce besoin avec nos webcams ; le test ne se déroule pas comme il devrait : En effet, il y a plusieurs ralentissement dans le traitement des images et nous pensons que c'est essentiellement dû aux trop grand nombre de threads, ainsi qu'à leurs la non communication. Il y a donc multiplication par deux du stockage et de la notification. Ces points peuvent être résolus dans une extension possible du logiciel comme décrit au chapitre 11, page 78.

Concernant les deux autres points des besoins non fonctionnels, à savoir la clarté et la compréhension du code ainsi que le developpement de façon réutilisable et extensible du code sont difficilement testables. Nous vous renvoyons donc à notre architecture et à la lecture de notre code pour vous faire une idée.

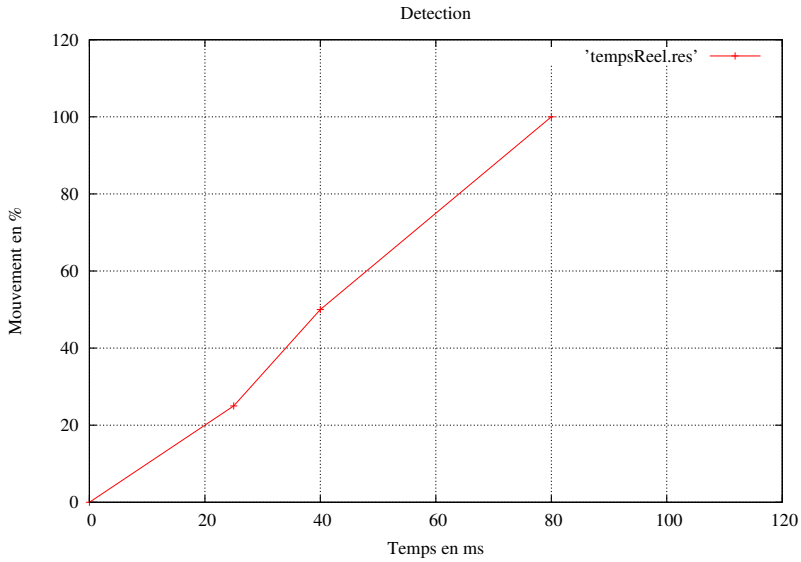

FIG. 10.9 – Evolution du temps en fonction du mouvement dans le champ de la caméra

10.2 Test en "boîte blanche" : Valgrind

Au début de l'implémentation de notre logiciel, nous avons eu de grosses fuites mémoires, on atteignait les 2 Go en 10 minutes pris sur notre disque dur. Puis, lors de la première utilisation de valgrind avec les options "–tool=memcheck" et "–leak-check=yes", nous nous sommes aperçu que certaines fonctions ne détruisaient pas des objets alloués ce qui faisait l'objet de petites fuites mémoires. Nous avons aussi constaté que des fuites de mémoire ont lieu depuis notre librairie, à savoir OpenCv.
Cependant, il se trouve que tout est correctement libéré, car aucun octet n'est définitivement perdu.
Voici le résumé de valgrind :
==7619== LEAK SUMMARY :
==7619== definitely lost : 0 bytes in 0 blocks.
==7619== possibly lost : 53939 bytes in 149 blocks.
==7619== still reachable : 9133256 bytes in 3703 blocks.
==7619== suppressed : 0 bytes in 0 blocks.
==7619== Reachable blocks (those to which a pointer was found) are not shown.

10.3 Limites du logiciel

Notre logiciel a malheureusement des limites dans son utilisation. Sans compter une meilleure résolution ou un meilleur zoom, un visage ne peut être détecté que s'il rentre entièrement dans le champ de la caméra. C'est à dire, à partir de 50 cm environ, et ne peut dépasser une certaine distance qui est de 11 m environ. Les performances de notre logiciel sont sensiblement les même que pour celui de Mr Carminati.

La figure 10.10 page 76 illustre parfaitement ces propos.

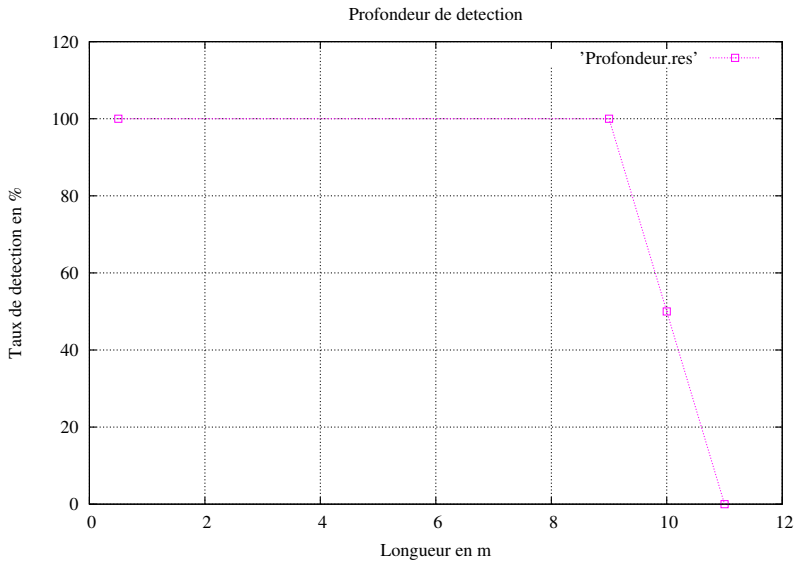

FIG. 10.10 – Distance pour une détection de visage

De plus, un visage ne peut avoir un angle maximum que de 25 ° avec la caméra pour l'application existante contre 45 ° pour la nôtre. La preuve en est donnée image 10.11 page 76.

FIG. 10.11 – Visage avec un angle maximum par rapport à la caméra pour qu'il soit détecté

Par suite, il était intéressant de savoir dans quel mesure, notre visage serait détecté. Nous avons cherché à savoir quel était le maximum de partie caché tout en étant détecté. Une illustration est donnée à la figure 10.12, page 77.
Les différents tests sur les deux logiciels mettent en exergue des performances supérieures de notre logiciel par rapport au logiciel existant tout en respectant le cahier des charges.

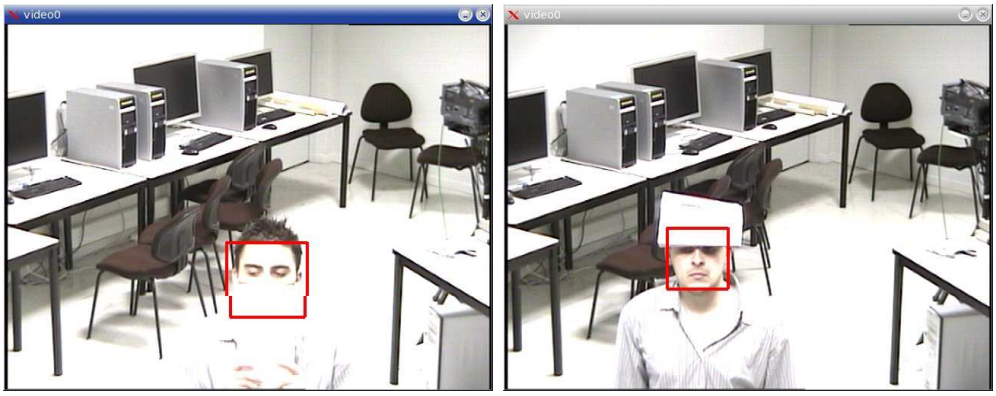

FIG. 10.12 – Detection de visages avec certaines parties cacher

Chapitre 11

Description technique des extensions possibles du logiciels

On peut imaginer plusieurs extensions possibles de notre programme. Ces extensions viseraient à améliorer l'affichage, à automatiser certaines manipulations, faciliter la modification des entrées du programme ou offrir plus de services à l'utilisateur. Nous vous proposons ci-dessous, les extensions utiles qui nous viennent à l'esprit.

11.1 Extensions du module de stockage

Suite aux recherches documentaires sur le domaine de notre projet de programmation, des modifications du cahier des charges ont été effectuées. Les recherches concernant l'aspect légal du projet montrent que le stockage des visages détectés par vidéosurveillance est soumis à des règles. La loi stipule que l'espace de stockage doit être sécurisé et que le temps de stockage des images ne doit pas excéder une période d'un mois (avec accord de la Préfecture de Police). Mais encore, s'il y a transfert des images vers un espace distant, il est impératif d'utiliser un protocole de transfert sécurisé. Après en avoir informer notre client, certains points du cahier des charges ont donc été revus. Ainsi, dans notre implémentation, nous utilisons le protocole de transfert sécurisé SSH pour le stockage des images des visages détectés sur le compte distant.
Par contre, le contrôle du temps de sauvegarde des images n'étant pas prise en compte de manière automatisée par notre application, nous en laissons l'entière responsabilité à l'utilisateur du programme. Une extension possible du module **StockeurVisages** consisterait donc à comparer la date d'enregistrement d'une image à la date actuelle et de prendre les dispositions nécessaires c'est à dire la suppression de celles-ci en cas d'échéance du délai. Ce traitement devra être appliqué sur la totalité des images stockées (stockage local et sur le compte distant).
Dans le même ordre d'idée, une autre extension possible de ce module est la gestion de l'espace disponible. Il suffirait de vérifier l'espace libre sur la session avant de procéder à la sauvegarde d'une image.

11.2 Extension du module de notification

La notification d'intrusion est aujourd'hui assuré par l'envoi d'un courriel à une adresse donnée en paramètre. L'utilisateur est alors avertit d'une détection lorsque celui-ci relève ses courriels et donc pas forcement au moment où l'intrusion est détectée. La personne à contacter pourrait être notifié par des moyens plus rapides, tels que le téléphone ou un sms. Le module de notification serait composé d'une interface et de plusieurs implémentations de moyens de communication. L'utilisateur choisirait le moyens d'envoi de l'alerte qu'il souhaite utiliser.

11.3 Extension du suiveur

Afin de réduire le temps de calcul de l'algorithme de détection de visages, qui constitue la partie la plus couteuse en temps de notre programme, nous utilisons des zones d'intérêts. En effet, si l'on procède à la détection de visages sur la totalité des images capturées, notre programme ne peut traiter le minimum de 10 images par seconde, que nous devons respecter. Pour cela nous avons créé un module **SelecteurZonesInteret** définissant les zones susceptibles de contenir des visages (les zones d'intérêts). Ces zones sont principalement composées des zones en mouvement mais aussi des zones des visages détectés à l'image précédente. Les anciennes zones assurent le suivi des visages, car lorsque que la personne reste immobile face à l'objectif de la caméra, aucun mouvement n'est détecté.Le suiveur résout ce défaut.
Une extension possible de notre programme résiderait à suivre les visages en question en calculant leurs déplacements. L'algorithme consisterait à extraire des points caractéristiques du visages détectés pour ensuite les comparer à ceux des visages détectés à l'image précédente. Il existe des fonctions de suivi d'objets implémentées dans la bibliothèque *Opencv*. Ces fonctions de haut niveau permettent le suivi de zones dans une séquence d'images. Cependant les tests de ces fonctionnalités montre que malgré le fait que le suivi d'objet soit convainquant, leurs utilisations ne permet pas un traitement en temps réel. Le temps de calcul de ce suivi évolué de visages ajouté à celui de notre programme ne permet que le traitement de quelques images par seconde.

11.4 Création d'un module de pilotage des caméras

Les caméras installées au LABRI étant motorisées, il est envisageable de les diriger dans notre application. On peut imaginer des caméras "intelligentes" qui suivent les personnes afin de détecter un maximum de visages. Avec un suiveur évolué comme présenté précédemment, on pourrait associer un vecteur de déplacement à chaque visages et donc changer l'orientation des caméras. Dans l'idéal, les caméras devront être positionnées de façon à obtenir un maximum de personnes dans leurs champs. Beaucoup de cas sont à considérer dans cette extension, comme par exemple le cas où deux visages sont détectés et partent dans des directions opposées. Encore plus loin dans le pilotage des caméras, l'application adapterait le zoom des caméras en fonctions de la taille des visages détectés. Un zoom (optique ou numérique) sur un visage éloigné (capturé ou non) offrirait une image d'une meilleure résolution. Le pilotage des caméras consisterait à avoir le maximum de zones en mouvement sur l'image capturée pour faciliter la détection de visages et obtenir des images de visages de tailles maximum.

11.5 Création d'un gestionnaire de fenêtres

De plus, les diverses fenêtres d'affichage ne sont pas disposées de façon à éviter le chevauchement de celles-ci. La création d'un module de gestion de sorties permettrait de régler cela. En connaissant le nombre de fenêtres à créer, le gestionnaire les disposerait équitablement à l'écran.

11.6 Extensions concernant la vidéosurveillance

Imaginons qu'on soit en présence de mouvement dans la pièce mais que l'application ne détecte aucun visage dans les images capturées, plusieurs formes d'actions et d'alertes peuvent être déclenché dans ce cas. Notamment l'envoi d'une notification indiquant la présence de mouvement dans la zone surveillée. Ou encore procédé à l'enregistrement des images susceptibles de contenir de des informations utiles.

Chapitre 12

Manuel utilisateur

12.1 A propos de ce manuel

12.1.1 Objectif du manuel

Ce manuel détaille les fonctions de service du **détecteur temps-réel de visages dans une séquence vidéo**. Mais aussi la présentation générale, qui permet de comprendre les objectifs et de connaître les conditions d'utilisation. Ensuite, nous expliquerons comment installer ce programme sur votre machine et comment l'utiliser. Des copies d'écran sont associées à la description des manipulations permises par le logiciel.

12.2 Présentation générale

12.2.1 Fonctions du service

Le détecteur temps-réel de visages est un programme permettant de détecter des visages dans une séquence vidéo en temps réel. Ce logiciel est exclusivement compatible Unix/linux. De nombreuses options sont disponibles dans cette application. Pour le choix des sources, le programme peut aussi bien détecter des visages à partir d'une caméra que d'un fichier vidéo. Un courriel de notification d'intrusion peut-être envoyé. Il est aussi proposé de stocker les images en local (sur la machine) ou sur un compte distant acceptant toutefois les connexions de type SSH (compte compatible et accessible avec le protocole SSH). On peut afficher le résultat de la détection ainsi que le mouvement détecté dans la vidéo.

12.2.2 Conditions d'utilisations

L'utilisation de ce logiciel est soumise à des lois.

- Votre dipositif de vidéosurveillance doit être déclaré à la CNIL (voir Code Pénal 226-18 et 226-16).
- Vous devez avertir clairement les personnes entrant dans la ou les zone(s) surveillée(s) de la présence du système de surveillance (voir Code Pénal 226-18 et 226-22).
- Il est formellement interdit de divulguer ou d'utiliser les images résultats de la détection (voir Code Pénal 226-19 et 226-22).

- Les images doivent être conservées au maximum un mois avec accord de la Préfecture de Police (voir Code Pénal 226-20).
- Les images doivent impérativement être stockées sur un espace sécurisé (voir Code Pénal 226-17).
- Il est interdit de ce servir des images sans accord écrit de la personne (voir Code Pénal 226-1 et 226-8).

12.2.3 Définition du temps-réel pour notre application

Dans ce manuel nous appellerons temps-réel, la capacité du programme à traiter au minimum 10 images par seconde.

12.2.4 Format des dates

Les dates sont affichées sous la forme JJ-MM-AA—HH-mm-ss. Ce format de date est utilisé pour le nommage des images sauvegardées. Comme plusieurs visages peuvent être sauvegardés avec la même date (intervalle de temps inférieur à une seconde), un compteur est alors concaténé à la suite de la date pour donner un unique nom de fichier pour chacunes des images.

12.2.5 Redimensionnement et positionnement des fenêtres

Les fenêtres ne sont pas redimensionnables car la résolution est fixée par l'utilisateur dans le fichier de configuration. Cependant, la disposition de celles-ci est libre. L'utilisateur peut répartir à sa guise les différentes fenêtres à l'écran.

12.2.6 Confidentialité

Lorsque l'utilisateur utilise le stockage distant vers un compte compatible SSH, son login est à ajouter au fichier de configuration. A la première utilisation de ce service, l'application procédera à l'échange d'une clé nécessaire pour le transfert crypté des données vers le compte distant. Cette clé est ensuite sauvegardée pour ne plus être demandée par la suite. Votre login étant connu, juste votre mot de passe de connexion vous sera demandé au démarrage de l'application. Une fois la validité du nom d'utilisateur et la validité du mot de passe vérifiées, l'application démarrera normalement. Toutes anomalies constatées pendant la connexion sera affichées dans le terminal.

12.2.7 Configuration matérielle

Pour utiliser le service Détecteur temps réel de visages dans de bonnes conditions, un poste de travail répondant aux conditions minimales suivantes est nécessaire :

- connexion Internet haut débit.
- écran de 15 pouces avec une résolution 800 x 600 pixels.
- processeur Intel Pentium IV ou équivalent AMD.
- caméra compatible linux (driver *v4l* ou *v4l2*).

12.3 Installation et lancement de l'application

Avant d'installer l'application, veuillez vérifier que toutes les dépendances requises sont satisfaites, puis procéder à l'installation. Pour cela, placer vous dans le répertoire de l'application.

Ensuite, aller dans le répertoire *sources/src* :

$ cd sources/src

Puis, donner les droits d'éxécution à l'éxécutable *configure* :

$ chmod u+x configure

Enfin lancer le configure :

$ ./configure

Installer les dépendances manquantes.

Puis, compiler le programme sur la machine, en utilisant la commande *make* :

$ make

Enfin, terminer par la commande *make install* :

$ make install

Les exécutables *PdpDetectionVisages* et *ConfigurationApplication* sont alors créés.

Il ne reste qu'à paramètrer le fichier de configuration, ensuite vous pourrez lancer l'application comme ceci :

- Soit en utilisant le fichier de configuration par défaut (*PdpDetectionVisages.cfg*) :

  $ ./PdpDetectionVisages

- Soit en utilisant un autre fichier de configuration comme *Exemples.cfg* :

  $ ./PdpDetectionVisages -conf Exemples.cfg

12.4 Création ou édition d'un fichier de configuration

Le fichier de configuration permet le chargement simple de toutes les entrées relatives au programme. Ce fichier de configuration porte l'extension *.cfg* . Pour éditer un fichier de

configuration, vous pouvez utiliser un éditeur de textes ou utiliser l'interface graphique prévu à cet effet.

12.4.1 Méthode avec un éditeur de texte

ouvrer le fichier de configuration *PdpDetectionVisages.cfg* avec votre éditeur de texte préféré. Les lignes commençants par le caractère # sont des commentaires du fichier de configuration, vous pouvez en ajouter autant que vous le souhaitez. Dans ce fichier, des champs sont obligatoires alors que d'autres sont facultatifs.

Fichier de configuration de base

Plusieurs champs sont obligatoires dans le fichier de configuration, nous vous en proposons un dans l'exemple suivant nommé *PdpDetectionVisages.cfg* :

#Parametres de capture
typeSource = camera
fichierSourceCapture = /dev/video0
largeurCapture = 640
hauteurCapture = 480

 # Parametres de detection de visages
cheminClassifieur = ../data/haarcascades/haarcascade_frontalface_alt2.xml
largeurMinVisage = 10
hauteurMinVisage = 10
afficheResultatDetection = true
afficheMouvement = false

Les valeurs numériques sont des entiers naturels. Ce fichier engendrera une detection à partir des images capturées avec une caméra (fichier de la caméra */dev/video0*). La capture est faite sur des images de résolution *640 X 480*. Il est aussi possible de détecter des visages à partir d'une vidéo, voir l'exemple suivant d'un *fichierSource* de type *fichierVideo* ainsi que l'emplacement de la source :

typeSource = fichierVideo
fichierSourceCapture = /video/exemple.avi

Le champ *cheminClassifieur* est le chemin par défaut, néanmoins plusieurs classifieurs sont disponibles dans le dossier *../data/haarcascades/*. Les mots clés *largeurMinVisage* et *hauteurMinvisage* définissent les dimensions minimums pour les visages détectés (exemple : minimum de *10 X 10* pixels). Les champs *afficheResultatDetection* et *afficheMouvement* ont pour but respectifs d'afficher le résultat de la détection et afficher le résultat des zones en mouvement, leurs valeurs sont des booléens (*true* ou *false*). Avec ce fichier de configuration de base, il n'y a ni stockage, ni notification d'intrusion.

Utilisation de l'option de notification par courriel

Voici la partie à ajouter au fichier de configuration pour profiter de cette option.

 # Parametres de notification d'intrusion par courriel
adresseDestinataire = <une adresse mail valide>
serveurSMTP = <une adresse de serveur SMTP valide>
portSMTP = 25
nomLieuSurveille = salle sous surveillance

Vous devez remplir la partie *Parametres de notification d'intrusion par courriel* avec une adresse mail valide et une adresse de serveur SMTP valide. Vous pouvez aussi si nécessaire modifier le port SMTP. Le champ *nomLieuSurveille* permet de mettre une description du lieu surveillé, cette description figurera dans les courriels envoyés.

Utilisation de l'option stockage

Pour le stockage (local ou distant), votre fichier de configuration est le fichier de base présenté au dessus auquel nous allons ajouter des champs.

- Pour un stockage local des images. Il faut ajouter :

  # Parametres de stockage
  typeStockage = local
  dossierStockage = ./

  Le type de stockage est donc local (sur la machine) et les images seront stockées dans le répertoire courant. Il est possible de changer de répertoire comme par exemple :

  dossierStockage = /home/visages/

- Pour un stockage distant des images vers un compte compatible SSH, La partie suivante doit être ajoutée au fichier de configuration :

  # Parametres de stockage
  typeStockage = distant
  dossierStockage = /home/visages/
  serveurSSH = <l'adresse du serveur SSH où le compte est accessible>
  portSSH = 22
  login = <votre nom d'utilisateur>

  En effet, le type de stockage est maintenant *distant*. Vous pouvez mettre dans le champ *dossierStockage* le chemin du dossier où les images des visages seront sauvegardées sur le compte distant. Vous pouvez modifier le port de connexion SSH si besoin. Enfin, on termine en mettant son nom d'utilisateur. Pour des raisons de sécurité et de confidentialité, le mot de passe sera uniquement demandé au lancement de l'application et ne sera jamais stocké.

12.4.2 Méthode avec l'Assistant de configuration

Le logiciel comporte une interface graphique pour l'édition plus conviviale du fichier de configuration. Vous pouvez lancer cette assistant de la manière suivante :

$ ./ConfigurationApplication

Vous pouvez modifier le nom du fichier généré en bas de l'application. Toutefois, le nom par défaut est aussi le fichier chargé par défaut, si vous changez le nom il faudra alors passer le fichier en paramètre à l'exécution (voir partie sur l'installation et l'exécution du programme). Pour sauvegarder et quitter, vous disposez des boutons de mêmes noms en bas à droite. Cette interface est composé de quatre onglets :

- Capture : pour y renseigner le type de sources et les différents paramètres de capture.
- Détection de visages : pour y renseigner le classifieur à utiliser et les affichages.
- Stockage des visages : pour y renseigner les informations relatives à l'option de stockage.
- Notification d'intrusion par courriel : pour y renseigner les informations relatives à l'envoi du courriel.

Onglet Capture

La figure 12.1, page 86 présente l'onglet *Capture*.

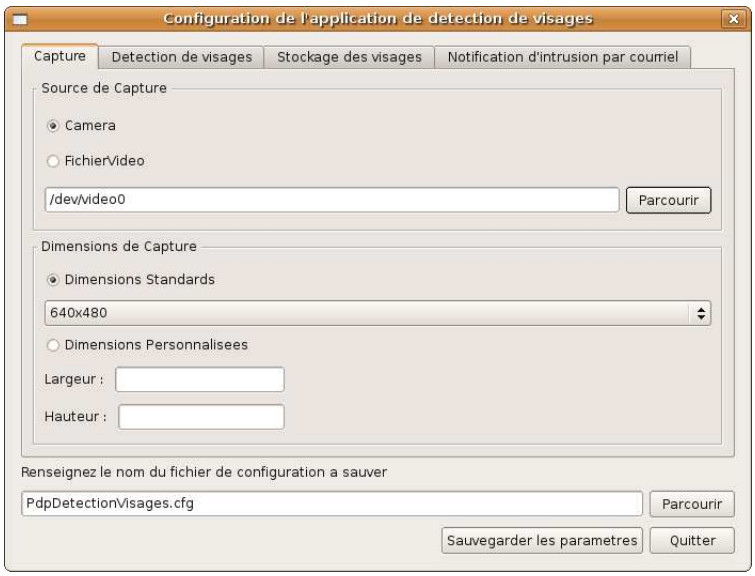

FIG. 12.1 – Onglet Capture de l'interface graphique configuration de l'application

La première chose à faire est de renseigner la source de capture (un fichier caméra ou un fichier video). Dans le cas par défaut, c'est à dire si vous désirez faire une détection à partir d'une caméra et que vous possèdez qu'une seule caméra, vous n'avez normalement rien

à modifier. Si toutefois l'emplacement de la source est différent, utiliser le bouton *Parcourir*. Même chose pour travailler à partir d'un fichier vidéo. Ensuite, il faut choisir une résolution. Il est conseillé d'en choisir une parmi les résolutions standards. Une mauvaise résolution peut engendrer une augmentation considérable du temps de calcul et donc des ralentissements du programme.

Onglet Détection de visages

La figure 12.2, page 87 présente l'onglet *Detection*.

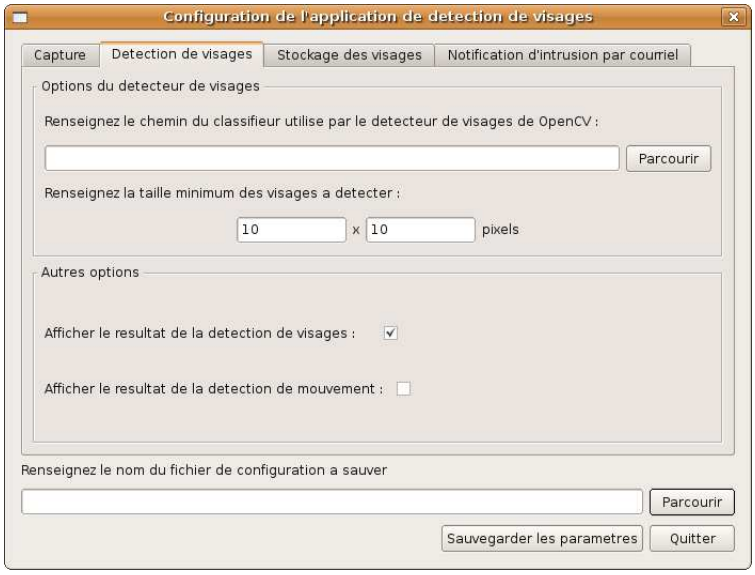

FIG. 12.2 – Onglet Détection de visages de l'interface graphique configuration de l'application

Dans cet onglet, le chemin du classifieur est demandé. Le classifieur est un outil nécessaire pour la détection de visages. Il est conseillé de laisser le classifieur par défaut, vous pouvez néanmoins en utiliser un autre en cliquant sur le bouton *Parcourir*. La taille minimum des visages détectés permet d'éviter de fausses détections ou d'avoir des images trop petites. Elle est par défaut de $10 * 10$ pixels. Vous pouvez augmenter légèrement cette valeur si vous êtes ammené à utiliser le programme dans une zone peut profonde (cela diminue le temps de calcul et amèliore ainsi la fluidité des affichages). Ensuite, nous avons par défaut l'affichage du résultat et pas l'affichage du mouvement.

Onglet Stockage des visages

La figure 12.3, page 88 présente l'onglet *Stockage*.

Il est possible de ne faire aucun stockage des visages en décochant la case en haut à gauche. Sinon vous avez le choix entre un stockage local(sur la machine) et un stockage distant. Pour le stockage local, seulement le repertoire d'accueil des images est demandé. Pour le stockage distant plusieurs champs obligatoires sont à renseigner. Le premier champ concerne le dossier

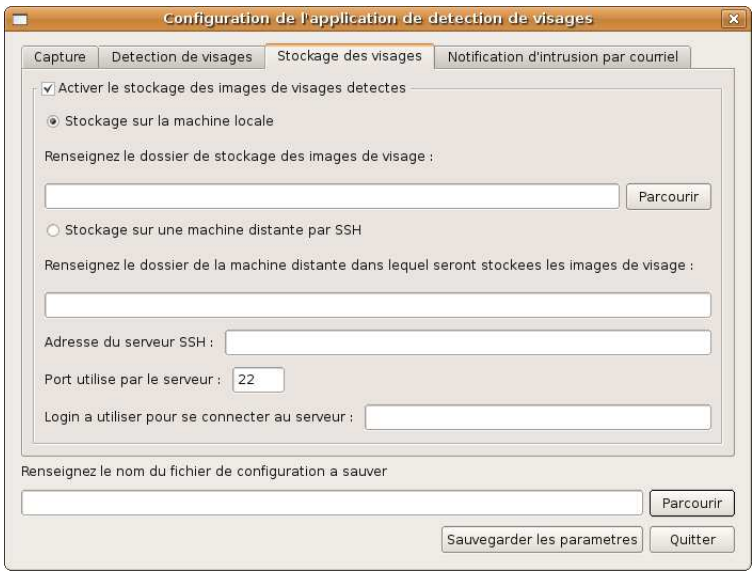

FIG. 12.3 – Onglet Stockage de visages de l'interface graphique configuration de l'application

où les images seront sauvegardées sur le compte distant. Le deuxième et troisième champs concernent respectivement le serveur SSH utilisé et le login. Si vous ignorez ces informations, contacter l'administrateur du compte. Le mot de passe de connexion ne sera demandé qu'au lancement de l'application.

Onglet Notification d'intrusion par courriel

La figure 12.4, page 89 présente l'onglet *Notification d'intrusion par courriel.*

Si vous ne souhaitez pas envoyer d'alertes par courriel, il suffit de désactiver cette option. Sinon une adresse courriel valide et l'adresse du serveur SMTP sont requises. Il est également possible de changer le port SMTP. Le nom du lieu surveillé est demandé, il figurera dans tous les courriels envoyés.

12.5 Exemples d'utilisation

Une fois le programme lancé avec un fichier de configuration valide, il n'y aura aucune intéraction entre vous et le programme. Si vous avez choisi d'afficher le résultat de la détection de visages, une fenêtre de taille la résolution choisie est ouverte. Dans cette fenêtre on peut voir les captures, c'est à dire les images de la sources vidéo (de 10 à 25 images par secondes) et aussi des rectangles rouges encadrants les visages détectés.

voir pour exemple la figure 12.5, page 89.

Si vous avez choisi d'afficher le résultat de la détection de mouvement. Une deuxième fenêtre est présente. Dans celle-ci on retrouve les zones en mouvement qui sont encadrées en vert (voir exemple figure 12.6, page 90, ce sont les zones où le détecteur de visages va principalement chercher.

Pour terminer, voici un exemple d'image sauvegardée (figure 12.7, page 90).

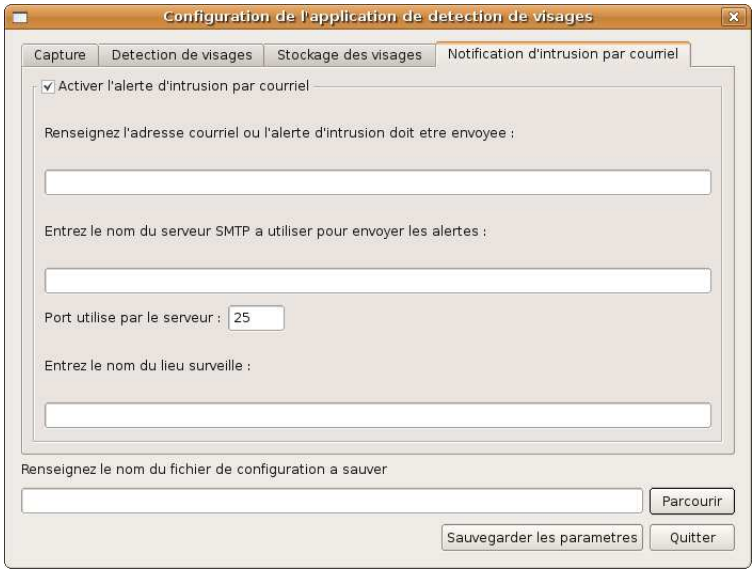

FIG. 12.4 – Onglet Notification d'intrusion par courriel de l'interface graphique configuration de l'application

FIG. 12.5 – Capture de la fenêtre de détection de visage sur le fichier caméra /dev/video0

Ces images sont consultables dans le dossier de sauvegarde choisi à la configuration, elles sont au format *jpeg* .Le nom des images est la date concaténé à un compteur représentant le nombre de visages détectés depuis le lancement de l'application. L'image porte le nom *05-04-07-13–28–34-7-1.jpeg* . Le visage a été détecté le 5 avril 2007 à 13 heures 28 minutes et 34 secondes. Le 7-1 veut dire que c'était la 7^{eme} images à contenir des visages depuis le lancement de l'application. Le chiffre 1 signifie que c'était le premier visage détecté dans l'image.

FIG. 12.6 – Capture de la fenêtre de mouvement sur le fichier caméra /dev/video0

FIG. 12.7 – Exemple d'image sauvegardée par l'application.

12.6 Foire aux questions

12.6.1 Concernant la compilation

Des erreurs peuvent survenir à la compilation. Pas de panique : faites ces vérifications :

- Toutes les dépendances doivent être installées avant la compilation. Pour cela utiliser la commande *./configure* qui vous retournera la liste des paquets manquants. Les bibliothèques *Opencv*, *PosixThread*, *libssh* et *GTK2+x* sont requises. La bibliothèque *QT4* est également requise pour l'application *ConfigurationApplication*.
- vérifier aussi les droits des fichiers. L'exécutable *configure* doit avoir les droits d'exécution.

12.6.2 Concernant l'exécution

La compilation est correcte mais l'exécution du programme (*./PdpDetectionVisages*) échoue.

- Une erreur de chargement de librairies ou de chargement de bibliothèques. Cette erreur est généralement dûe aux variables d'environnements. Pour exemples :

  Soit XXX le chemin de la bibliothèque :

export PKG_CONFIG_PATH=$PKG_CONFIG_PATH :/XXX/lib/pkgconfig
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH :/XXX/lib

remarque : le chemin est souvent /usr/local/lib soit :

export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig :$PKG_CONFIG_PATH
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH :/usr/local/lib

- une erreur de type cvError (initialisation de matrices) : cette erreur est généralement causée par une résolution non compatible avec la camera ou au fait que la caméra ne fonctionne pas. Référez vous aux données constructeur de votre caméra pour identifier les résolutions compatibles. Modifier le fichier de configuration en conséquence.

  Verifier aussi la présence du fichier video. (logiquement vous devez avoir */dev/video0* présent sur votre disque). Vous pouvez tester la caméra avec le logiciel *camorama* par exemple.

12.6.3 Des erreurs apparaissent dans le terminal

- Vous avez un message indiquant que le stockage distant a échoué. Vérifier les points suivants :

  - Le compte distant est bien accessible en SSH.
  - Le repertoire dans lequel vous voulez stocker le résultat des détections existe.
  - Les bons paramètres ont été saisies lors de la création du fichier de configuration.
  - Il y a suffisamment d'espace disponible sur la machine et sur le compte distant.

- Problème d'envoi du courriel de notification d'intrusion. Vérifier les points suivants :

  - Vous avez bien accès au serveur SMTP, que vous avez spécifié, depuis cette machine.
  - Les bons paramètres ont été saisies lors de la création du fichier de configuration.
  - Le serveur peut vous bloquer temporairement si vous avez envoyé un trop grand nombre de courriel.

  Si l'erreur persiste contacter votre administrateur.

Annexe A

Extraits de textes de loi relatifs à l'utilisation de la vidéosurveillance

Dispositions relatives à la prévention de l'insécurité

A.1 Article 10 de la loi 95-73

Modifié par Loi n ° 2006-64 du 23 janvier 2006 art. 1 (JORF 24 janvier 2006).

I. - Les enregistrements visuels de vidéosurveillance répondant aux conditions fixées au II sont soumis aux dispositions ci-après, à l'exclusion de ceux qui sont utilisés dans des traitements automatisés ou contenus dans des fichiers structurés selon des critères permettant d'identifier, directement ou indirectement, des personnes physiques, qui sont soumis à la loi n ° 78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés.

II. - La transmission et l'enregistrement d'images prises sur la voie publique, par le moyen de la vidéosurveillance, peuvent être mis en oeuvre par les autorités publiques compétentes aux fins d'assurer la protection des bâtiments et installations publics et de leurs abords, la sauvegarde des installations utiles à la défense nationale, la régulation du trafic routier, la constatation des infractions aux règles de la circulation ou la prévention des atteintes à la sécurité des personnes et des biens dans des lieux particulièrement exposés à des risques d'agression ou de vol.

La même faculté est ouverte aux autorités publiques aux fins de prévention d'actes de terrorisme ainsi que, pour la protection des abords immédiats de leurs bâtiments et installations, aux autres personnes morales, dans les lieux susceptibles d'être exposés à des actes de terrorisme.

Il peut être également procédé à ces opérations dans des lieux et établissements ouverts au public aux fins d'y assurer la sécurité des personnes et des biens lorsque ces lieux et établissements sont particulièrement exposés à des risques d'agression ou de vol ou sont susceptibles d'être exposés à des actes de terrorisme.

Les opérations de vidéosurveillance de la voie publique sont réalisées de telle sorte qu'elles ne visualisent pas les images de l'intérieur des immeubles d'habitation ni, de façon spécifique,

celles de leurs entrées.

Le public est informé de manière claire et permanente de l'existence du système de vidéosurveillance et de l'autorité ou de la personne responsable.

III. - L'installation d'un système de vidéosurveillance dans le cadre du présent article est subordonnée à une autorisation du représentant de l'Etat dans le département et, à Paris, du préfet de police, donnée, sauf en matière de défense nationale, après avis d'une commission départementale présidée par un magistrat du siège ou un magistrat honoraire.

L'autorisation préfectorale prescrit toutes les précautions utiles, en particulier quant à la qualité des personnes chargées de l'exploitation du système de vidéosurveillance ou visionnant les images et aux mesures à prendre pour assurer le respect des dispositions de la loi.

L'autorisation peut prescrire que les agents individuellement désignés et dûment habilités des services de police et de gendarmerie nationales sont destinataires des images et enregistrements. Elle précise alors les modalités de transmission des images et d'accès aux enregistrements ainsi que la durée de conservation des images, dans la limite d'un mois à compter de cette transmission ou de cet accès, sans préjudice des nécessités de leur conservation pour les besoins d'une procédure pénale. La décision de permettre aux agents individuellement désignés et dûment habilités des services de police et de gendarmerie nationales d'être destinataires des images et enregistrements peut également être prise à tout moment, après avis de la commission départementale, par arrêté préfectoral. Ce dernier précise alors les modalités de transmission des images et d'accès aux enregistrements. Lorsque l'urgence et l'exposition particulière à un risque d'actes de terrorisme le requièrent, cette décision peut être prise sans avis préalable de la commission départementale. Le président de la commission est immédiatement informé de cette décision qui fait l'objet d'un examen lors de la plus prochaine réunion de la commission.

Les systèmes de vidéosurveillance installés doivent être conformes à des normes techniques définies par arrêté ministériel, à compter de l'expiration d'un délai de deux ans après la publication de l'acte définissant ces normes.

Les systèmes de vidéosurveillance sont autorisés pour une durée de cinq ans renouvelable.

La commission départementale instituée au premier alinéa peut à tout moment exercer, sauf en matière de défense nationale, un contrôle sur les conditions de fonctionnement des dispositifs autorisés en application des mêmes dispositions. Elle émet, le cas échéant, des recommandations et propose la suspension des dispositifs lorsqu'elle constate qu'il en est fait un usage anormal ou non conforme à leur autorisation.

Les autorisations mentionnées au présent III et délivrées antérieurement à la date de publication de la loi n° 2006-64 du 23 janvier 2006 relative à la lutte contre le terrorisme et portant dispositions diverses relatives à la sécurité et aux contrôles frontaliers sont réputées délivrées pour une durée de cinq ans à compter de cette date.

III bis. - Lorsque l'urgence et l'exposition particulière à un risque d'actes de terrorisme le requièrent, le représentant de l'Etat dans le département et, à Paris, le préfet de police peuvent délivrer aux personnes mentionnées au II, sans avis préalable de la commission départementale, une autorisation provisoire d'installation d'un système de vidéosurveillance, exploité dans les conditions prévues par le présent article, pour une durée maximale de quatre mois. Le président de la commission est immédiatement informé de cette décision. Il peut alors la réunir sans délai afin qu'elle donne un avis sur la mise en oeuvre de la procédure d'autorisation provisoire.

Le représentant de l'Etat dans le département et, à Paris, le préfet de police recueillent l'avis de la commission départementale sur la mise en oeuvre du système de vidéosurveillance conformément à la procédure prévue au III et se prononcent sur son maintien. La commission doit rendre son avis avant l'expiration du délai de validité de l'autorisation provisoire.

IV. - Hormis le cas d'une enquête de flagrant délit, d'une enquête préliminaire ou d'une information judiciaire, les enregistrements sont détruits dans un délai maximum fixé par l'autorisation. Ce délai ne peut excéder un mois.

V. - Toute personne intéressée peut s'adresser au responsable d'un système de vidéosurveillance afin d'obtenir un accès aux enregistrements qui la concernent ou d'en vérifier la destruction dans le délai prévu. Cet accès est de droit. Un refus d'accès peut toutefois être opposé pour un motif tenant à la sûreté de l'Etat, à la défense, à la sécurité publique, au déroulement de procédures engagées devant les juridictions ou d'opérations préliminaires à de telles procédures, ou au droit des tiers.

Toute personne intéressée peut saisir la commission départementale mentionnée au III de toute difficulté tenant au fonctionnement d'un système de vidéosurveillance.

Les dispositions du précédent alinéa ne font pas obstacle au droit de la personne intéressée de saisir la juridiction compétente, au besoin en la forme du référé.

VI. - Le fait d'installer un système de vidéosurveillance ou de le maintenir sans autorisation, de procéder à des enregistrements de vidéosurveillance sans autorisation, de ne pas les détruire dans le délai prévu, de les falsifier, d'entraver l'action de la commission départementale, de faire accéder des personnes non habilitées aux images ou d'utiliser ces images à d'autres fins que celles pour lesquelles elles sont autorisées est puni de trois ans d'emprisonnement et de 45000 euros d'amende, sans préjudice des dispositions des articles 226-1 du code pénal et L. 120-2, L. 121-8 et L. 432-2-1 du code du travail.

VI bis. - Le Gouvernement transmet chaque année à la Commission nationale de l'informatique et des libertés un rapport faisant état de l'activité des commissions départementales visées au III et des conditions d'application du présent article.

VII. - Un décret en Conseil d'Etat fixe les modalités d'application du présent article et notamment les conditions dans lesquelles le public est informé de l'existence d'un dispositif de vidéosurveillance ainsi que de l'identité de l'autorité ou de la personne responsable. Ce décret fixe également les conditions dans lesquelles les agents visés au III sont habilités à accéder aux enregistrements et les conditions dans lesquelles la commission départementale exerce son contrôle.

A.2 Article 10-1 de la loi 95-73

Créé par Loi n ° 2006-64 du 23 janvier 2006 art. 2 (JORF 24 janvier 2006).

I. - Aux fins de prévention d'actes de terrorisme, le représentant de l'Etat dans le département et, à Paris, le préfet de police peuvent prescrire la mise en oeuvre, dans un délai qu'ils fixent, de systèmes de vidéosurveillance, aux personnes suivantes :

- ◇ les exploitants des établissements, installations ou ouvrages mentionnés aux articles L. 1332-1 et L. 1332-2 du code de la défense ;
- ◇ les gestionnaires d'infrastructures, les autorités et personnes exploitant des transports collectifs, relevant de l'activité de transport intérieur régie par la loi n ° 82-1153 du 30 décembre 1982 d'orientation des transports intérieurs ;
- ◇ les exploitants d'aéroports qui, n'étant pas visés aux deux alinéas précédents, sont ouverts au trafic international.

II. - Préalablement à leur décision et sauf en matière de défense nationale, le représentant de l'Etat dans le département et, à Paris, le préfet de police saisissent pour avis la commission départementale instituée à l'article 10 quand cette décision porte sur une installation de vidéosurveillance filmant la voie publique ou des lieux et établissements ouverts au public.

Les systèmes de vidéosurveillance installés en application du présent article sont soumis aux dispositions des quatrième et cinquième alinéas du II, des deuxième, troisième, quatrième et sixième alinéas du III, du IV, du V, du VI et du VII de l'article 10.

III. - Lorsque l'urgence et l'exposition particulière à un risque d'actes de terrorisme le requièrent, le représentant de l'Etat dans le département et, à Paris, le préfet de police peuvent prescrire, sans avis préalable de la commission départementale, la mise en oeuvre d'un système de vidéosurveillance exploité dans les conditions prévues par le II du présent article. Quand cette décision porte sur une installation de vidéosurveillance filmant la voie publique ou des lieux ou établissements ouverts au public, le président de la commission est immédiatement informé de cette décision. Il peut alors la réunir sans délai afin qu'elle donne un avis sur la mise en oeuvre de la procédure de décision provisoire.

Avant l'expiration d'un délai maximal de quatre mois, le représentant de l'Etat dans le département et, à Paris, le préfet de police recueillent l'avis de la commission départementale sur la mise en oeuvre du système de vidéosurveillance conformément à la procédure prévue au III de l'article 10 et se prononcent sur son maintien.

IV. - Si les personnes mentionnées au I refusent de mettre en oeuvre le système de vidéosurveillance prescrit, le représentant de l'Etat dans le département et, à Paris, le préfet de police les mettent en demeure de procéder à cette installation dans le délai qu'ils fixent en tenant compte des contraintes particulières liées à l'exploitation des établissements, installations et ouvrages et, le cas échéant, de l'urgence.

V. - Est puni d'une amende de 150 000 Euros le fait, pour les personnes mentionnées au I, de ne pas avoir pris les mesures d'installation du système de vidéosurveillance prescrit à l'expiration du délai défini par la mise en demeure mentionnée au IV.

Annexe B

Convention de programmation.

B.1 Nom de fichier

Tous les fichiers du projet devront respecter les règles suivantes

- Les fichiers présentant une interface porteront l'extension .h
- Les fichiers d'implémentation auront eux l'extension .cc

B.2 Structure des fichiers

La taille des lignes ne devra pas être supérieure à 80 caractères.
Quand une instruction ne tient pas sur une ligne, les sauts de ligne seront à effectuer après une virgule ou avant un opérateur.
L'indentation utilisera 2 espaces.
Le fichier source sera organisé autour du modèle suivant

- Un commentaire présentant la licence, la version du fichier, et la date.
- Les inclusions, qui suivront l'ordre suivant :
  - ◇ Inclusions issue du noyau
  - ◇ Inclusion issue des bibliothèques
  - ◇ Inclusion locales
- Les constantes globales.
- Les méthodes, ou les procédures.

Les fichiers d'interface commenceront par une macro de protection contre les inclusions multiples.
Puis les prototypes des différentes méthodes, ou procédures, dont l'utilisation est possible en dehors du module, seront indiqués.

B.3 Convention de nommage

Les noms des classes, des méthodes, ou des procédures devront permettre de savoir de quoi traite cette entité, en respectant l'organisation des prototypes (voir B.4). Il pourra être composé de plusieurs mots. Pour séparer les mots, la première lettre du mot suivant sera en

majuscule.
Les noms des classes commenceront obligatoirement par une majuscule.
Tandis que les noms des variables ainsi que les noms des méthodes commenceront par une minuscule.
Les constantes seront écrites entièrement en majuscule, les différents mots seront séparés par des “souligné” (_).

B.4 Organisation des prototypes

Les noms des fonctions seront produit sous la forme
<NomDeLAction><ObjetDeLAction>[<InformationSuplementaire>]
L'ordre des paramètres, pour la plupart des fonctions, sera les entrées, les sorties, les options.

B.5 Déclarations

Les déclarations de types différents ne pourront pas être effectuées sur la même ligne.
Quand c'est possible, les variables seront initialisées lors de leur déclaration. Les déclarations de variables seront placées au plus près de leur utilisation.

B.6 Saut de ligne et espace

Un saut de ligne sera placé entre les différentes parties d'une méthode, entre des variables dont l'utilisation est différente, entre la déclaration des variables et le code.
Deux saut de ligne seront placés : entre deux méthodes ou deux procédures. Les points virgules n'indiquant pas la fin d'une ligne, ainsi que toutes les virgules, seront suivis par une espace.
Les opérateurs seront encadrés par des espaces.

B.7 Gestion des erreurs

Toutes les opérations d'entrée-sortie seront vérifiées, soit par la génération d'exceptions, soit par des retours de valeur. Un affichage de l'erreur produite devra être produit.

B.8 Gestion de la mémoire

Les allocations de mémoire seront faite grâce à la fonction **new**. Le désallocation de mémoire seront faite avec le fonction **delete**. L'utilisation des fonctions *malloc* (et assimilées) et *free* est à **proscrire**.

B.9 Mise en page

B.9.1 Structure du “If ... Then ... Else ...”

```
if ([condition]) {
  [code]
```

```
} else if ([condition]) {
  [code]
} else {
  [code]
}
```

B.9.2 Structure du “For ...”

```
for ([int i = 0]; [i < N]; i++) {
  [code]
}
```

B.9.3 Structure du “While ...”

```
while ([condition]) {
  [code]
}
```

Les boucles infinies seront réalisées avec des “while (0)”

B.9.4 Structure du “Do ... While ...”

```
do {
  [code]
} while ([condition])
```

B.9.5 Structure du “Switch”

```
switch ([variable]) {
  case ([cas1]) : {
    [code]
  }
  case ([cas2]) : {
    [code]
  }
  default : {
    [code]
  }
}
```

B.10 Commentaires

Ils doivent servir à expliquer le rôle des différents composants, et à expliquer les parties complexes du code.

Bibliographie

[1] L. Carminati. *Détection et suivi d'objets dans les scènes animées : Application à la vidéosurveillance*. PhD thesis, Université Bordeaux I, 2006.

[2] Jacques Georgel. *Les libertés de communication*, chapter 2, pages 45–61. 1996.

[3] P. Gauvin. Droit à l'image et droit de l'image, 2006. http ://savoirscdi.cndp.fr/rencontrelyon/gauvin/gauvin.htm , dernière mise à jour le 27 octobre 2006.

[4] Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja. Detecting Faces in Images : A Survey. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 24(1), 2002.

[5] Robert Frischholz. The Face Detection Homepage. http ://www.facedetection.com. A page focused on the task of detecting faces in arbitrary images. Last update : January 11th 2007.

[6] I. Craw, D. Tock, and A. Benett. Finding Face Features. *Proc. Second European Conf. Computer Vision*, pages 92–96, 1992.

[7] G.J. Edwards, C.J. Taylor, and T. Cootes. Learning to Identify and Track Faces in Image Sequences. *Proc. Sixth IEEE Int'l Conf. Computer Vision*, pages 317–322, 1998.

[8] I.A. Essa and A. Pentland. Facial Expression Recognition Using a Dynamic Model and Motion Energy. *Proc. Fifth IEEE Int'l Conf. Computer Vision*, pages 360–367, 1995.

[9] R. Chellappa, C.L. Wilson, and S. Sirohey. Human and Machine Recognition of Faces : A Survey. *Proc. IEEE*, 83(5) :705–740, 1995.

[10] C. Kotropoulos, A. Tefas, and I. Pitas. Frontal Face Authentification Using Variants of Dynamic Link Matching Based on Mathematical Morphology. *Proc. IEEE Int'l Conf. Image Processing*, pages 122–126, 1998.

[11] G. Yang and T. S. Huang. Human Face Detection in Complex Background. *Pattern Recognition*, 27(1) :53–63, 1994.

[12] S. McKenna, S. Gong, and Y. Raja. Modelling Facial Colour and Identity with Gaussian Mixtures. *Pattern Recognition*, 31(12) :1883–1892, 1998.

[13] A. Lanitis, C.J. Taylor, and T.F. Cootes. An Automatic Face Identification System Using Flexible Appearance Models. *Image and Vision Computing*, 13(5) :393–401, 1995.

[14] H. Rowley, S. Baluja, and T. Kanade. Neural Network-based Face Detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(1) :23–38, 1998.

[15] E. Osuna, R. Freund, and F. Girosi. Training Support Vector Machines : An Application to Face Detection. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[16] A. Cornuéjols and L. Miclet. *Apprentissage Artificiel : Méthodes et Algorithmes*, chapter 9 : Support Vector Machine, Séparateurs à vaste marge. Eyrolles, 2002.

[17] A. Cornuéjols. Une nouvelle méthode d'apprentissage : les SVM. Séparateurs à vaste marge. *Bulletin de l'AFIA*, Juin 2002.

[18] V. Vapnik. *The Nature of Statistica Learning Theory.* Springer-Verlag, 1995.

[19] G. Bradski and J. Davis. Motion Segmentation and Pose Recognition with Motion History Gradients. *IEEE WACV'00*, 2000.

[20] A. Bobick and J. Davis. The Representation and Recognition of Action Using Temporal Templates. *MIT Media Lab Technical Report 402*, 1997.

[21] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. *IEEE CVPR*, 2001. The paper is available online at http ://www.ai.mit.edu/people/viola/.

[22] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. *IEEE ICIP*, 1 :900–903, September 2002. This paper, as well as the extended technical report, can be retrieved at http ://www.lienhart.de/Publications/publications.html.

[23] A. Cornuéjols and L. Miclet. *Apprentissage Artificiel : Méthodes et Algorithmes*, chapter 11 : Apprentissage par combinaison de décisions : le dopage (boosting). Eyrolles, 2002.

[24] R.E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5 :197–227, 1990.

[25] Yoav Freund and Robert E. Schapire. *Computational Learning Theory : Eurocolt '95*, chapter A decision-theoretic generalization of on-line learning and an applicationto boosting, pages 23–37. Springer-Verlag, 1995.