كلية العلوم و التقنيات فاس
ⵜⴰⵖⵙⵓⵍⵜ ⵏ ⵜⵓⵙⵙⵏⵉ ⴷ ⵜⵙⵏⵙⵜⵉ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵜ ⵙⵏⴷ ⵎⵓⵃⴰⵎⴷ ⵙ ⵄⴰⴱⴷⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

LICENCE SCIENCES ET TECHNIQUES
Génie Electrique

RAPPORT DE FIN D'ETUDES

Intitulé :

Contribution au développement d'un système intelligent pour la reconnaissance de la main

Réalisé Par :

BOUZAMMOUR Btissam

Encadré par :

Pr **AHAITOUF Ali** **FST de Fès**

Pr **MANSOURI Anas** **Encadrant LERSI, ENSA de Fès**

Soutenu le 12 Juin 2019, devant le jury composé de :

Pr	**ERRAHIMI Fatima**	**Examinatrice**
Pr	**ESSBAI Najia**	**Examinatrice**
Pr	**MANSOURI Anas**	**Invité**
Pr	**AHAITOUF Ali**	**Encadrant**

Remerciements

Au terme de ce travail, je tiens à exprimer ma profonde gratitude à notre cher professeur et encadrant Mr. Ahaitouf Ali de la Faculté des Sciences et Techniques de Fès, pour son suivi et son énorme soutien qu'il n'a cessé de me prodiguer tout au long de la période du projet. Je suis reconnaissante pour sa bonne humeur, son dévouement, son temps accordé, et surtout pour l'opportunité d'effectuer mon stage au sein de son laboratoire LERSI, ce qui m'a permis de découvrir le monde de la recherche.

Je tiens à remercier également le professeur MANSOURI Anas de l'Ecole nationale des sciences appliquées de Fès pour ses conseils, ses directives, sa disponibilité, sa générosité et son soutien tout au long de la période du stage.

Je tiens à remercier, tout particulièrement, professeur GUENNOUNI Souhail de l'Ecole nationale de sciences appliquées de Fès pour ses directives, son aide, le partage de son expertise et son temps accordé.

J'adresse aussi mes vifs remerciements aux membres des jurys pour avoir bien voulu examiner et juger ce travail.

Enfin je tiens à remercier toutes les personnes qui ont contribué de près ou de loin à l'élaboration de ce travail. Qu'ils trouvent ici l'expression de ma profonde reconnaissance et mon profond respect.

Résumé

La reconnaissance automatique est la clé du domaine de la vision par ordinateur, elle peut concerner le corps humain, le visage, la reconnaissance des empreintes digitales ou la gestuelle. Son essor est dû à la disponibilité d'algorithmes puissants qui concernent principalement le traitement d'images.

Dans ce présent rapport, il s'agit de la détection de la main humaine et de sa gestuelle dans le but d'appréhender cette interaction homme-machine.

Pour ce cas de détection, nous avons choisi d'utiliser l'algorithme de Viola et Jones.

Cela passe par une compréhension de cet algorithme de traitement d'images, de la constitution d'une base de données et de la mise au point d'un programme d'un environnement adéquat de travail comprenant la caméra, la carte Raspberry et finalement d'un langage de codage, notre choix étant reporté sur python dans un environnement de Linux embarqué.

Notre but ultime étant d'aborder le domaine de la reconnaissance d'objets, l'apprentissage automatique et également la détection et la reconnaissance des gestes associé à la LSA (American Sign Langue).

Mots Clés

Traitement d'images, détection de la main, algorithme de Viola et Jones, caméra, carte Raspberry, Linux, reconnaissance des gestes, apprentissage automatique.

Sommaire

Liste des figures

Introduction générale

La détection de la main revient à trouver les coordonnées délimitant une main dans une image ou une vidéo. C'est un cas spécifique de détection d'objet, où l'on cherche à détecter la présence et la localisation précise d'une ou plusieurs mains dans une ou plusieurs images. C'est l'un des domaines de la vision par ordinateur parmi les plus étudiés avec de très nombreuses publications et de conférences spécialisées.

Dans les dernières années, il y a eu un intérêt accru chez les chercheurs pour utiliser ce type de détection pour l'interprétation des mots de la langue des signes. Une langue utilisée dans les communautés de sourds-muets, y compris les interprètes, les amis et les familles. Cependant, ces langues ne sont pas généralement connues en dehors de ces communautés, et donc les barrières de communication existent entre les sourds et les entendants. Trouver quotidiennement des interprètes qualifiés et expérimentés est une tâche très difficile et aussi inabordable. D'où l'intérêt et le besoin croissant des outils d'analyse automatiques de vidéo de langue des signes. Un tel système permettrait de réduire les obstacles à la communication entre les sourds-muets et les autres personnes.

Récemment, plusieurs approches de reconnaissance de signes ont été proposées. Toutefois, le processus de la détection gestuelle a besoin d'un système d'acquisition de l'information performant, rapide et éventuellement économique pour donner des bons résultats , tels que la méthode de Viola et Jones que nous étudierons en particulier.

Ce rapport comporte trois chapitres principaux :

Un premier relatif au contexte de l'étude à savoir la position du problème, la détection du mouvement de la main en étudiant les principales composantes de ce système, ainsi qu'une présentation de laboratoire de travail et du plan d'action

Un deuxième chapitre dont lequel nous avons présenté la méthode de Viola et Jones en décrivant son principe et le matériel nécessaire à la réalisation de notre projet.

Le troisième chapitre est consacré à la partie conception et implémentation de la méthode adoptée, où nous présenterons les étapes d'installation des systèmes d'exploitation et des logiciels d'environnement, la constitution de la base de donnée et les résultats obtenues.

On terminera notre mémoire par une conclusion générale.

Chapitre 1 :

Contexte et objectifs

1. Présentation du laboratoire de travail

Le Laboratoire Energies Renouvelables et Systèmes Intelligents (LERSI) à la FST de Fès, a été créé en 2014, accueille 14 enseignants-chercheurs, 35 doctorants, constitué de 3 équipes de recherche dont les axes sont :

- Microélectronique & Systèmes Embarqués (MSE)
- Systèmes à Energies Renouvelables : Intégration & Gestion Intelligence (SERIGI)
- Réseaux, Télécommunication et Ingénierie Logicielle (RLT)

J'ai effectué mon stage au sein d'équipe Microélectronique et systèmes embarqués, sous la direction du Pr. A. Mansouri, professeur à l'ENSA de Fès et le Pr. A.AHAITOUF de la FST de Fès.

2. Problématique

Les gestes sont un moyen naturel de communication. Dans la vie courante, ils viennent ponctuer ou renforcer l'expression orale entre personnes. Avec la langue des signes, ils constituent le moyen de communication privilégié pour de nombreux malentendants.

L'acquisition des gestes de la main pour la reconnaissance de la langue des signes nécessite un système de détection qui analyse l'information contenue dans des images fixes ou en temps réel. Ce dernier est le plus souvent construit par l'apprentissage automatique.

A cet égard, je propose une approche par l'algorithme AdaBoost, un algorithme robuste et rapide basé sur la densité d'images, qui combine des descripteurs simples pour un classificateur fort.

La détection gestuelle reste un sujet complexe, dû à la grande variabilité d'apparence de la main dans des conditions sans contraintes comme la variabilité intrinsèque (couleur, taille, forme) et les conditions d'illumination.

Il est essentiel de définir la qualité d'une détection qui se caractérise par trois types :

- Le premier est nommé « détection positive » qui pointe effectivement une main.
- Le deuxième est nommé «détection fausse positive » où la zone détectée ne correspond pas une main.
- Le troisième type est nommé « non-détection » où il existe un ou plusieurs mains non détectés dans l'image

3. Plan d'action

Ce diagramme montre l'échelonnement temporel et exécutif que j'ai mis en place pour réaliser ce projet :

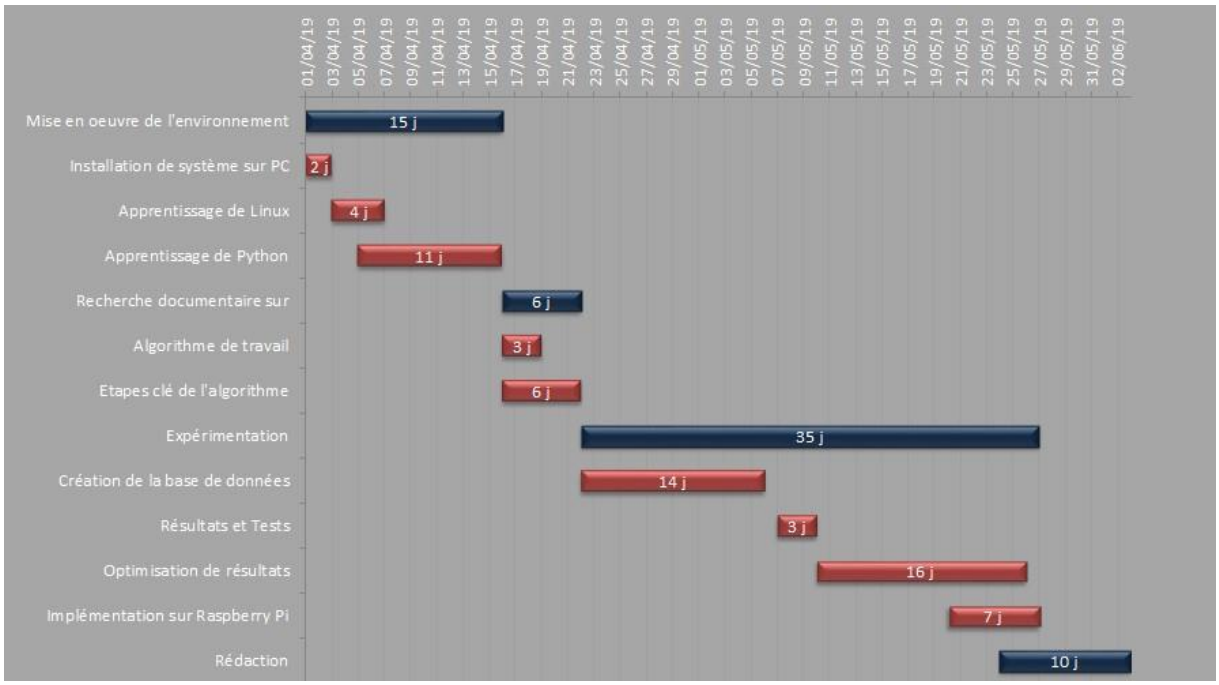


Figure 1 : Organisation temporelle du projet

Chapitre 2 :

La détection du mouvement par la méthode de Viola et Jones

1. Introduction

Après la présentation du laboratoire de travail et du plan d'action, ce deuxième chapitre vient pour détailler toutes les bases théoriques et techniques nécessaires à la réalisation de ce projet.

2. Algorithme de Viola et Jones

La méthode de Viola et Jones [1] est une méthode de détection d'objet dans une image numérique, elle fait partie des toutes premières méthodes capables à détecter efficacement et en temps réel des objets dans une image.

Cette méthode est une approche basée sur l'apparence, qui consiste à parcourir l'ensemble de l'image en calculant un certain nombre de caractéristiques dans des zones rectangulaires qui se chevauchent. Elle a la particularité d'utiliser des caractéristiques très simples mais très nombreuses. Une première innovation de la méthode est l'introduction des images intégrales, qui permettent le calcul rapide de ces caractéristiques. Une deuxième innovation importante est la sélection de ces caractéristiques par boosting, en interprétant les caractéristiques comme des classificateurs. Enfin, la méthode propose une architecture pour combiner les classificateurs boostés en un processus en cascade.

En tant que méthode d'apprentissage supervisé, la méthode de Viola et Jones est divisée en deux étapes :

- une étape d'apprentissage qui nécessite de quelques à plusieurs milliers d'exemples de l'objet que l'on souhaite détecter pour entraîner un classificateur.
- une phase de détection par application de ce classificateur pour détecter la présence éventuelle de l'objet dans une image en parcourant celle-ci à toutes les positions et dans toutes les tailles possibles.

Cette méthode bénéficie d'une implémentation sous licence BSD dans OpenCV, la bibliothèque utilisée dans notre application.

3. Principe

La technique utilise les caractéristiques pseudo-Haar. Cette technique consiste à définir des zones rectangulaires et adjacentes comme montre la figure 2. On calcule ensuite la somme des intensités des pixels de l'image dans ces zones. La différence entre les rectangles noirs et blanc donne la caractéristique pseudo-Haar.

La méthode de Viola et Jones consiste à balayer une image à l'aide d'une fenêtre de détection de taille initiale 24px par 24px (dans l'algorithme original). Lorsque l'image a été parcourue entièrement, la taille de la fenêtre est augmentée et le balayage recommence, jusqu'à ce que la fenêtre fasse la taille de l'image. L'augmentation de la taille de la fenêtre se fait par un facteur multiplicatif de 1.25.

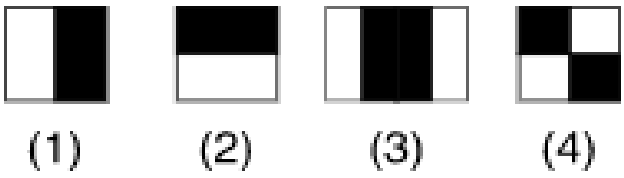


Figure 2 : Caractéristiques de Viola et Jones

Le balayage, quant à lui, consiste simplement à décaler la fenêtre d'un pixel. Ce décalage peut être changé afin d'accélérer le processus, mais un décalage d'un pixel assure une précision maximale [2].

La renommée de cette approche est faite sur trois concepts :

- L'image intégrale.
- Algorithme d'apprentissage basé sur AdaBoost.
- Cascade de classificateur.

3.1. L'image intégrale

Pour calculer rapidement et efficacement les caractéristiques pseudo-Haar sur une image, Viola et Jones proposent une nouvelle méthode, qu'ils appellent « image intégrale ». C'est une représentation sous la forme d'une image, de même taille que l'image d'origine, qui en chacun de ses points contient la somme des luminances des pixels situés au-dessus de lui et à sa gauche.

Le calcul de la somme des valeurs des pixels appartenant à une zone rectangulaire s'effectue donc en accédant seulement à quatre pixels de l'image intégrale : Soit un rectangle ABCD (figure 3) dont les sommets sont nommés dans le sens des aiguilles d'une montre en commençant par le sommet supérieur gauche. La somme des pixels dans le rectangle D peut être calculée avec quatre opérations. La valeur de l'image intégrale au point 1 est la somme des pixels dans le rectangle A. La valeur à la position 2 est A + B, à la position 3 est A + C, et à la position 4 est A + B + C + D. la somme dans D peut être calculée comme 4 +1-(2 +3)

Une caractéristique de Haar étant une combinaison linéaire de tels rectangles ABCD, son calcul se fait alors en un temps indépendant sa taille.

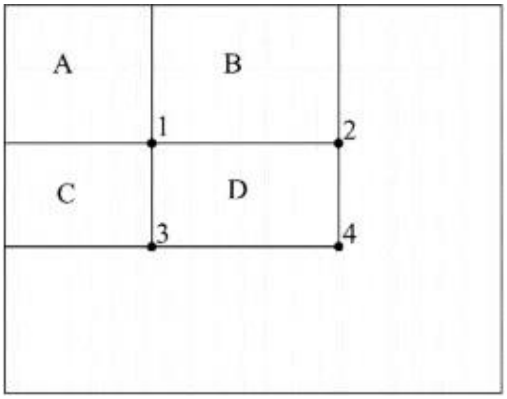


Figure 3 : Calcul de la somme du rectangle D avec l'image intégrale

3.2. Algorithme d'apprentissage basé sur AdaBoost

L'algorithme de boosting utilisé est une version modifiée d'AdaBoost (Adaptive Boosting), une méthode qui vise à combiner plusieurs classificateurs faibles pour obtenir un classificateur fort plus efficace en réduisant le taux d'erreur.

Ce principe a été initialement conçu par Robert Schapire et Yoav Freund en 1995 [4] et adapté par Viola et Jones en assimilant une caractéristique à un classificateur faible, en construisant un classificateur faible qui n'utilise qu'une seule caractéristique. L'apprentissage du classificateur faible consiste alors à trouver la valeur seuil de la caractéristique qui permet de mieux séparer les exemples positifs des exemples négatifs. Le classificateur se réduit alors à un couple (caractéristique, seuil).

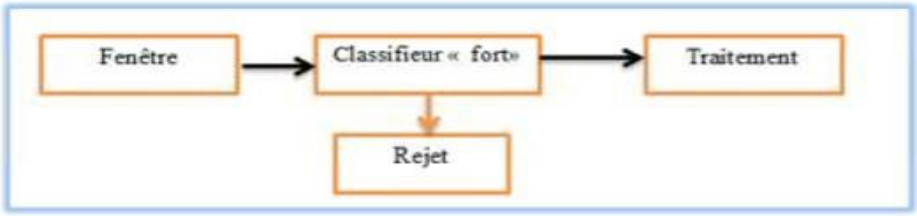


Figure 4 : Sélection par boosting

En d'autres termes, l'algorithme d'AdaBoost, appelle à chaque itération, un algorithme d'apprentissage qui entraînent les instances à classifier. Par la suite, AdaBoost définit une nouvelle distribution de probabilité pour les instances d'apprentissage en fonction des résultats de l'algorithme à l'itération précédente tout en augmentant le poids des instances qui ne sont pas correctement classées (figure 5). A la fin, AdaBoost combine les données faibles par un vote pondéré pour en déduire un classificateur fort.

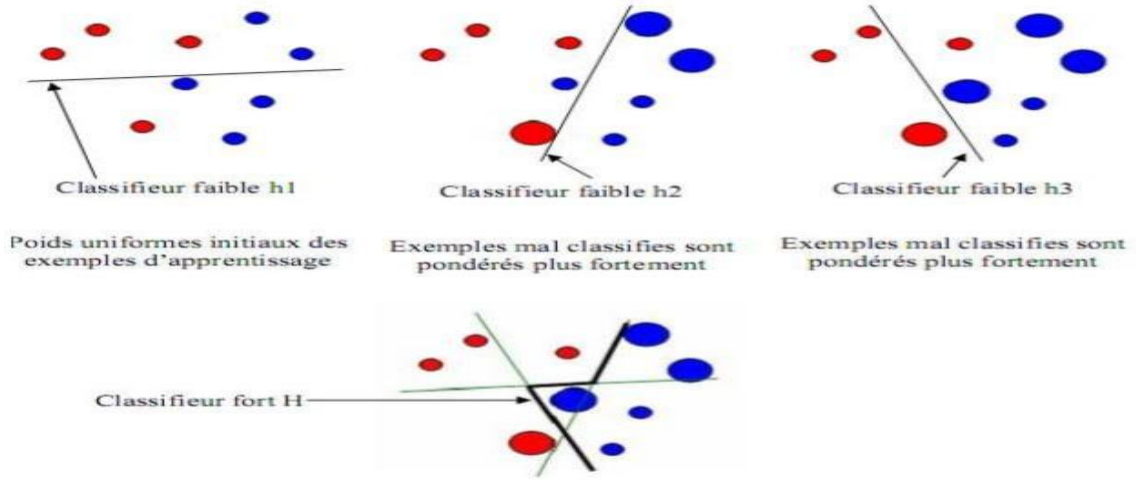


Figure 5 : Description d'AdaBoost

3.3. Classificateurs en cascade

La méthode de Viola et Jones est basée sur une approche par recherche exhaustive sur l'ensemble de l'image, qui teste la présence de l'objet dans une fenêtre à toutes les positions et à plusieurs échelles. Cette approche est cependant extrêmement coûteuse en calcul.

L'une des idées-clés de la méthode pour réduire ce coût réside dans l'organisation de l'algorithme de détection en un ensemble de classificateurs en cascade. Appliqués séquentiellement, ces classificateurs prennent une décision d'acceptation ou de rejet. Dans le premier cas, la fenêtre contient l'objet et l'exemple est alors passé au classificateur suivant, dans le cas du rejet, la fenêtre ne contient pas l'objet et dans ce cas l'exemple est définitivement écarté. L'idée est que l'immense majorité des fenêtres testées étant négatives

(c.-à-d. ne contiennent pas l'objet), il est avantageux de pouvoir les rejeter avec le moins possible de calculs. Ici, les classificateur les plus simples, donc les plus rapides, sont situés au début de la cascade, et rejettent très rapidement la grande majorité des exemples négatifs. Cette structure en cascade peut également s'interpréter comme un arbre de décision dégénéré, puisque chaque nœud ne comporte qu'une seule branche [4].

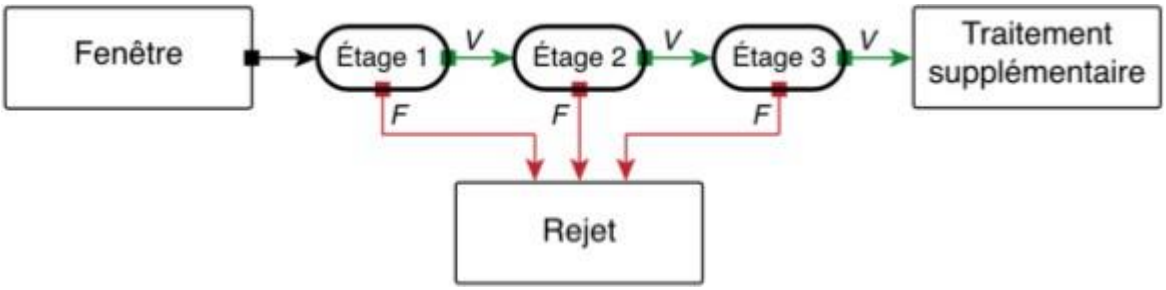


Figure 6 : Illustration de l'architecture en cascade

4. Environnement de travail

4.1. Système d'exploitation

✓ Ubuntu

Le système de base choisi pour la création de l'application est le système GNU/Linux, plus précisément Linux Ubuntu 16.04 LTS. Ce choix se base sur le fait que GNU/Linux est libre, sous la licence GPL (GNU Public License), ceci permet un accès plus facile au code et autorise toutes modifications et améliorations de l'existant. Contrairement à d'autres systèmes fermés, GNU/Linux dispose d'une communauté très dynamique. Ce système d'exploitation est assez connu pour sa stabilité, sa fiabilité et sa robustesse. Notre projet peut être amené à fonctionner pendant de longues heures sans interruption, c'est pour cela qu'elle nécessite un tel système d'exploitation.

Le système d'exploitation GNU/Linux se divise en deux versions :

- la version principale stable dit LTS. Une nouvelle version sort tous les deux ans, le support de sécurité et des mises à jour sont assurés pendant cinq ans.
- la version secondaire. Une nouvelle mise à jour sort tous les six mois et le support de sécurité et de mises à jour sont assurés pendant 9 mois.

✓ Raspbian

Raspbian est un système d'exploitation libre et gratuit basé sur Debian optimisé pour fonctionner sur un Raspberry Pi. C'est est un mot-valise formé par la fusion des mots «Raspberry Pi» et «Debian» [5]. Il s'agit d'une modification de Debian spécifiquement adaptée pour les systèmes sur une puce.

Il existe plusieurs versions de Raspbian :

- avec logiciels recommandés, comme le pack bureautique LibreOffice
- sans aucun logiciel supplémentaire
- sans aucun logiciel ni interface graphique (Raspbian Lite)

4.2. Matériels utilisés

✓ Raspberry Pi 3

Le Raspberry Pi est un nano-ordinateur mono carte à processeur ARM conçu par des professeurs du département informatique de l'université de Cambridge dans le cadre de la fondation Raspberry Pi. Nous avons utilisé le Modèle 3 B+ (figure 7)

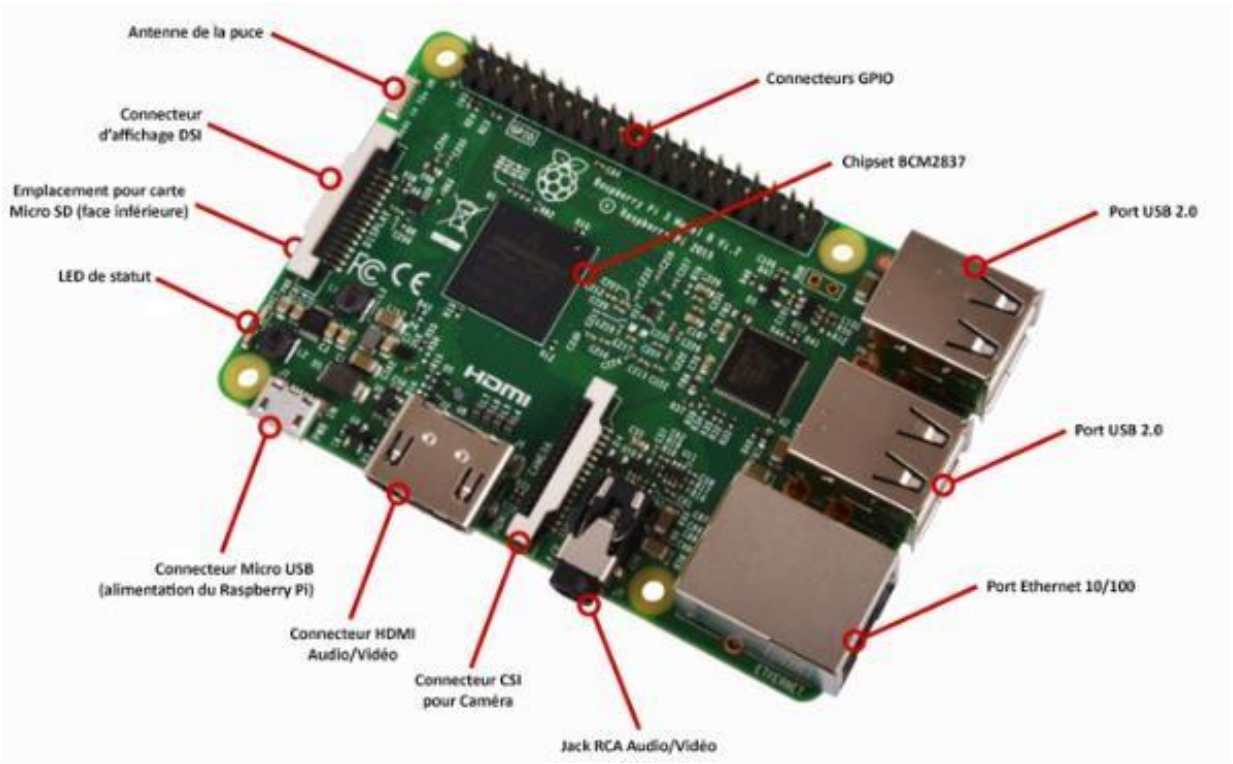


Figure 7 : les éléments de la Raspberry Pi 3

Cet ordinateur, de la taille d'une carte de crédit, est destiné à encourager l'apprentissage de la programmation informatique ; il permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux, notamment Debian, et des logiciels compatibles. Mais il fonctionne également avec l'OS Microsoft Windows.

Il est fourni nu, c'est-à-dire la carte mère seule, sans boîtier, alimentation, clavier, souris ni écran, dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération.

✓ Caméra Raspberry pi v2

Le module de caméra Raspberry Pi v2 est un capteur d'image 8 mégapixels Sony IMX219 (figure 8) conçu pour Raspberry Pi et doté d'une lentille à focale fixe. Il permet de prendre des images statiques de 3280 x 2464 pixels et supporte également les formats vidéo 1080p30, 720p60 et 640x480p60/90. Il se raccorde au Pi par le biais de l'un des petits connecteurs sur la surface supérieure de la carte et utilise l'interface CSI dédiée, spécialement conçue pour les caméras. La carte est de petite dimension, d'environ 25mm x 23mm x 9mm. Elle pèse à peine plus de 3g, pour une utilisation idéale dans des applications mobiles où la taille et le poids sont primordiaux. Connexion au Raspberry Pi via un petit câble en nappe.

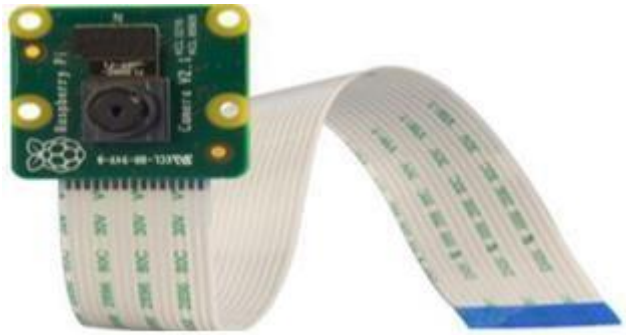


Figure 8 : Caméra Raspberry Po v2

4.3. Environnement logiciel

✓ Python

Le langage de programmation choisi pour le développement est le langage python. Ce langage est connu comme étant le meilleur outil de programmation système. Comparé au langage de programmation Java, C++. Python a l'avantage de s'adapter à tout type d'utilisation grâce à des bibliothèques spécialisées. Cependant, il est particulièrement utilisé comme langage de script pour automatiser des tâches simples, mais fastidieuses ou certains enchaînements d'actions répétitives. Ainsi, il peut être utilisé comme langage de développement de prototype lorsqu'on a besoin d'une application fonctionnelle avant de l'optimiser avec un langage de plus bas niveau. Il est particulièrement répandu dans le monde scientifique, et possède de nombreuses extensions destinées aux applications numériques. Sans oublier que c'est le système apprécié par la plupart des pédagogues qui y trouvent un langage où la syntaxe, clairement séparée des mécanismes de bas niveau, permet une initiation aisée aux concepts de base de la programmation.

✓ OpenCV

- Qu'est-ce que c'est ?

OpenCV est une bibliothèque libre de vision par ordinateur. Cette bibliothèque est écrite en C et C++ et peut être utilisée sous Linux, Windows et Mac OS X. Des interfaces ont été développées pour Python, Ruby, Matlab et autre langage. Open CV est orientée vers des applications en temps réel.

Un des buts d'OpenCV est d'aider les gens à construire rapidement des applications sophistiquées de vision à l'aide d'infrastructure simple de vision par ordinateur. La bibliothèque d'OpenCV contient près de 500 fonctions.

Il est possible grâce à la « licence de code ouvert » de réaliser un produit commercial en utilisant tout ou partie d'OpenCV. Il n'est pas obligatoire de montrer le code du produit et les améliorations réalisées au domaine public.

- Fonctionnalités

La bibliothèque OpenCV met à disposition de nombreuses fonctionnalités très diversifiées permettant de créer des programmes partant des données brutes pour aller jusqu'à la création d'interfaces graphiques basiques [6].

Traitement d'images

Elle propose la plupart des opérations classiques en traitement bas niveau des images :

· Lecture, écriture et affichage d'une image.
· Calcul de l'histogramme des niveaux de gris ou d'histogrammes couleurs.
· Lissage, filtrage.
· Binarisation, segmentation en composantes connexes.
· Morphologie mathématique.

Traitement Vidéo

Elle met également à disposition de l'utilisateur quelques fonctions d'interfaces graphiques, comme les curseurs à glissière, les contrôles associés aux événements souris, ou bien l'incrustation de texte dans une image.

Cette bibliothèque est imposée comme un standard dans le domaine de la recherche parce qu'elle propose un nombre important d'outils issus de l'état de l'art en vision des ordinateurs tels que :

· Lecture, écriture et affichage d'une vidéo (depuis un fichier ou une caméra)
· Détection de droites, de segment et de cercles par Transformée de Hough
· Détection de visages par la méthode de Viola et Jones
· Cascade de classificateurs boostés
· Détection de mouvement, historique du mouvement
· Poursuite d'objets par mean-shift ou Camshift
· Etc.…

5. Conclusion

Dans ce chapitre nous avons défini la méthode de Viola et Jones, ensuite nous avons présenté son principe et enfin étudié la partie matérielle et les outils logiciels utilisés dans notre système proposé.

Chapitre 3 :

Expérimentation

1. Introduction

Ce chapitre est sacré à la conception et la réalisation de notre algorithme. La première partie est consacrée à l'installation des systèmes d'exploitation et des logiciels d'environnement. Tandis que la seconde partie est consacrée à la constitution de la base de données. Dans ce qui suit, nous présenterons les résultats obtenus à la suite des tests effectués.

2. Installation et configuration

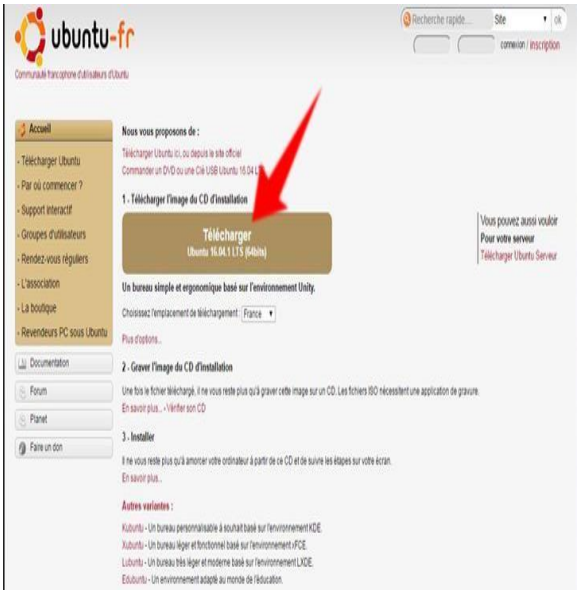
2.1. Installation d'Ubuntu 16.04 LTS

L'installation de Linux est faite en mode persistant qui permet la conservation des modifications et l'enregistrement des données sur le périphérique de stockage USB.

Ce processus nécessite 2 étapes :

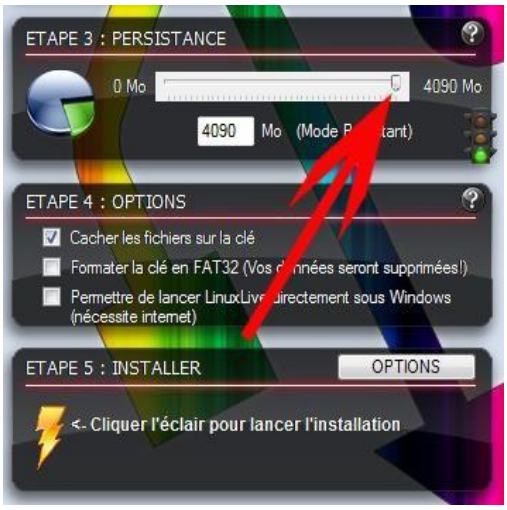
- Etape 1 : Créer une clé USB d'installation Ubuntu

Téléchargement du fichier ISO d'Ubuntu et du logiciel Live Creator USB Linux.



- Etape 2 : Préparation de l'installation d'Ubuntu en utilisant Live Cretor USB Linux

Après l'installation de logiciel «LiLi USB Creator », on insère la clé USB et on complète les paramètres nécessaires.



Une fois que l'installation est complète, on redémarre l'ordinateur à partir de la clé USB Boot.

Ensuite, on lance l'installation, on règle l'emplacement géographique, la disposition du clavier ainsi que l'identité.

2.2. Installation d'OpenCV sous Linux

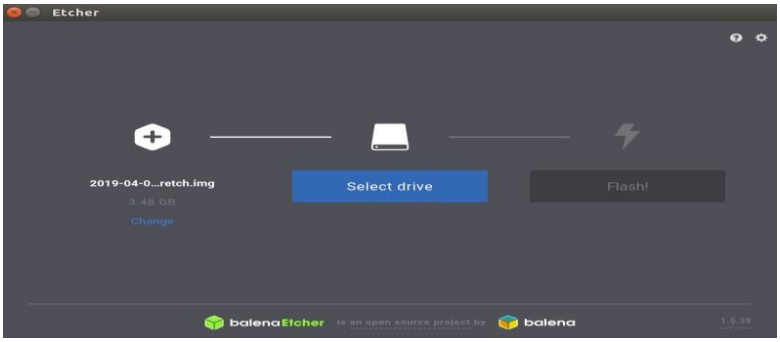
Ubuntu 16.04 LTS est livré avec Python 2.7 et Python 3.5 par défaut, cependant, ce langage ne dispose pas à la base de fonctions de traitement photo ni vidéo. Pour cela, il faut utiliser une bibliothèque qui ajoute des fonctions à notre programme. Il suffit alors d'importer Open CV et de l'installer sous Linux par les étapes suivantes :

- Sélectionnez la version d'OpenCV à installer
- Mise à jour des paquets
- Installation des bibliothèques d'OS
- Installation des bibliothèques de python
- Téléchargement d'OpenCV et OpenCV_contrib
- La compilation d'OpenCV

2.3. Installation et configuration de Raspbian

Compte tenu du fait que la Raspberry Pi ne possède pas de disque dur, cette dernière utilise une carte MicroSD comme disque dur, nous allons devoir installer un système d'exploitation dessus, nous choisirons ici Raspbian, une distribution robuste, adaptée à la grande majorité des usages et optimisée pour la Raspberry Pi.

- Etape 1: Télécharger la version de Raspbian et Etcher



- Etape 2 : Configuration sur Raspberry pi

Après avoir installé Raspbian dans notre microSD, il est temps de démarrer notre Raspberry Pi pour la première fois en suivant ces instructions:

1) Insérez la carte microSD dans la Raspberry Pi

2) Connectez un câble Ethernet de la Raspberry Pi à notre routeur pour nous assurer d'avoir une connexion Internet

3) Ouvrir un nouveau Terminal et passer en mode administrateur (root) avec «*sudo su*»

4) Créer un fichier nommé SSH sur la partition de démarrage

5) Connexion via SSH avec «*ssh* pi@adresse *IP*»
SSH est un outil bien pratique qui permet d'ouvrir un terminal de manière sécurisée à distance sur la Raspberry Pi depuis une autre machine. Cela nous permettra de poser la Raspberry dans un coin, sans clavier ni souris, et de faire des modifications dessus depuis votre PC de bureau.

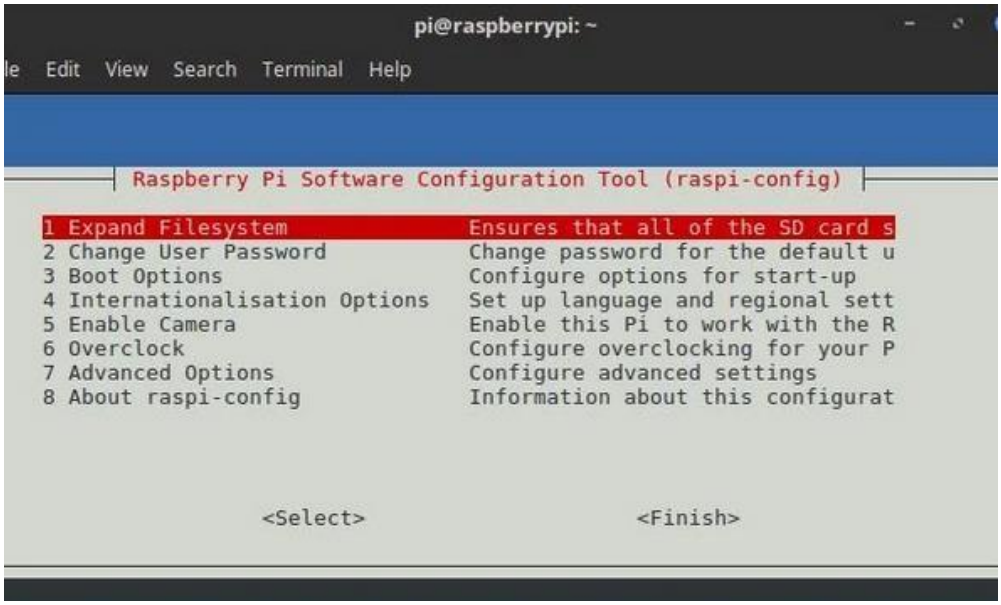
```
btissam@btissam-Latitude-E5470:~$ ssh pi@192.168.1.8
pi@192.168.1.8's password:
Linux raspberrypi 4.19.42-v7+ #1219 SMP Tue May 14 21:20:58 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat May 18 18:17:26 2019 from 192.168.1.6
pi@raspberrypi:~ $ pwd
/home/pi
```

6) Configuration avec «*raspi-config*»

raspi-config vise à fournir la fonctionnalité nécessaire pour effectuer les changements de configuration les plus courants



- Les options :

- *Expand Filesystem* : par défaut la partition système fait 2Go, cette Option va permettre d'agrandir cette partition occuper toutes la carte SD (4Go, 8Go, 16Go, 32Go, 64Go …)
- *Boot Options* : changer notre préférence de démarrage en ligne de commande ou de bureau
- *Enable Camera* : activer l'interface de la camera
- *Advanced Options* :

-- *Overscan* : ramener la bordure au cas de disparition du texte initial affiché à l'écran

– *Memory Split* : modifier la quantité de mémoire mise à disposition du GPU.

– *Audio* : Définir la résolution vidéo HDMI/DVI par défaut à utiliser lorsque le système démarre sans téléviseur ou moniteur connecté

- *About raspi-config* : Cet outil fournit une façon simple de faire la configuration initiale du Raspberry Pi. Bien qu'il puisse être exécuté à tout moment, certaines des options peuvent avoir des difficultés si l'installation est fortement personnalisée.

7) Redémarrer la Raspberry Pi avec «reboot»

Après avoir configuré des options utiles au premier démarrage de Raspbian, on redémarre notre Raspberry Pi pour valider les changements.

2.4. Installation d'OpenCV sous Raspberry Pi

Pour installer OpenCV sur Raspberry Pi, on suit les étapes suivantes

- Libérer de l'espace
- Installer des Dépendances
- Installer Python 3, setuptools, dev et Numpy
- Télécharger OpenCV 3.4 et OpenCV_contrib
- Compiler et installer OpenCV 3.4 pour python 3
- Echanger la taille de l'espace avant de compiler pour ajouter plus de mémoire virtuelle
- Compiler avec 4 cœurs
- Installer la construction sur Raspberry Pi

3. Constitution de la base de données

La bibliothèque OpenCV nous fournit une démonstration extrêmement intéressante pour une détection du mouvement de la main. De plus, elle nous fournit les programmes qu'ils utilisaient pour former des classificateurs pour un système de détection, appelé HaarTraining, afin que nous puissions créer nos propres classificateurs d'objet à l'aide de ces fonctions.

L'objectif de ce paragraphe est de fournir les procédures étape par étape pour détecter un mouvement de la main.

- Etape 1 : **Création des images positives et images négatives**

Nous devons collecter 6000 images positives ne contenant que la main, prise sous différents angles et éventuellement différents types d'éclairage, et 2000 images négatives ne contenant pas d'objets d'intérêt, pour former un classificateur de haarcascade.

.

- Etape 2 : **Organisation des images négatives**

Une liste d'images négatives est créée à l'aide d'un code python [s1] avec le contenu suivant :

image0010.jpg
image0011.jpg
image0012.jpg

.

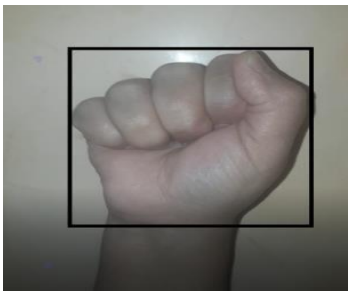
- Etape 3 : **Marquage des positifs**

Dans cette étape, nous devons créer un fichier de données vecteur contenant des images positives marquées. Notamment, nous devons préciser ou se trouve l'objet que nous souhaitons détecter, afin que l'algorithme sache quoi chercher. Ce fichier vecteur peut être créé à l'aide de l'utilitaire : ObjectMarker. Dans la figure ci-dessous nous proposons un exemple de marquage, cette procédure est réalisée pour toutes les images positives.

A la fin nous aurons un fichier texte info.txt avec le contenu suivant :

opencv_workspace/frame345.jpg 1 194 142 234 306
opencv_workspace/frame291.jpg 1 60 234 230 292
opencv_workspace/fram161.jpg 1 149 161 231 323

Le premier chiffre donne une information sur le nombre de main existant sur chaque image, ainsi que leurs positions. Pour l'image frame345.jpg, le nombre 1 veut dire qu'il y a une seule main située entre x = 194, y = 142, width = 234, height = 306.



- Etape 4 : **Création des échantillons (samples)**

Nous pouvons créer des échantillons de formation et des échantillons de test avec l'utilitaire « createsamples ».

Ceci est une liste d'options, mais il y a principalement quatre fonctions et les significations des options deviennent différentes d'une fonction à l'autre.

```
Utilisation: ./createsamples

  [-info <nom_fichier_description>]

  [-img <nom_fichier_image>]

  [-vec <nom_fichier_vec>]

  [-bg <nom_fichier_arrière-plan>]

  [-num <number_of_samples = 1000>]

  [-bgcolor <background_color = 0>]
```

```
  [-inv] [-randinv] [-bgthresh <background_color_threshold = 80>]
  [-maxidev <max_intensity_deviation = 40>]
  [-maxxangle <max_x_rotation_angle = 1.100000>]
  [-maxyangle <max_y_rotation_angle = 1.100000>]
  [-maxzangle <max_z_rotation_angle = 0.500000>]
  [-show [<scale = 4.000000>]]
  [-w <largeur_échantillon = 24>]
  [-h <sample_height = 24>]
```

Commande :

```
$ perl createsamples.pl positives.txt negatives.dat samples 6000 "opencv_createsamples -
bgcolor 0 -bgthresh 0 -maxxangle 1.1 -maxyangle 1.1 maxzangle 0.5 -maxidev 40 -w 20 -h 20"
```

Après avoir créé des échantillons de formation, nous devons générer un fichier de collection et les fusionner par le programme **mergevec.py**

Commande :

```
$ find samples/ -name '*.vec' > samples.dat
$ mergevec.py –v samples.dat –o samples.vec
```

- Etape 5 : **Entrainement**

Maintenant, nous formons notre propre classificateur à l'aide de l'utilitaire haartraining. Voici l'usage du haartraining.

```
Utilisation: ./haartraining
  -data <nom_répertoire>
  -vec <nom_fichier_vec>
  -bg <nom_fichier_arrière-plan>
  [-npos <number_of_positive_samples = 2000>]
  [-nneg <number_of_negative_samples = 2000>]
  [-nstages <number_of_stages = 14>]
  [-nsplits <number_of_splits = 1>]
```

```
[-mem <memory_in_MB = 200>]
[-sym (par défaut)] [-nonsym]
[-minhitrate <min_hit_rate = 0.995000>]
[-maxfalsealarm <max_false_alarm_rate = 0.500000>]
[-weighttrimming <weight_trimming = 0.950000>]
[-eqw]
[-mode <BASIC (par défaut) | CORE | TOUS>]
[-w <largeur_échantillon = 24>]
[-h <sample_height = 24>]
[-bt <DAB | RAB | LB | GAB (par défaut)>]
[-err <classe erronée (par défaut) | gini | entropie>]
[-maxtreesplits <nombre_max_de_splits_en_tree_cascade = 0>]
[-minpos <numéro_min_de_positive_échantillons_per_cluster = 500>]
```

Commande :

```
$opencv_haartraining -data haarcascade -vec samples.vec -bg negatives.dat -nstages 15 -nsplits
2 -minhitrate 0.99 -maxfalsealarm 0.5 -npos 5400 -nneg 2000 -w 24 -h 24 -nonsym -mem 512 -mode
ALL
```

Les données fournies dans la figure 8 sont liées pour la première étape de la formation

-parent node : Définit l'étape actuelle des procédures de formation

-N : Nombre de fonctions utilisées à cette étape

-%SMP : Pourcentage de l'échantillon utilisé pour cette caractéristique

-F : Nombre de fonctions utilisées à cette étape

-ST.THR : Seuil THRESHOLD

-HR : Taux de réussite basé sur le seuil de l'étape

-FA : Fausse alarme basée sur le seuil de la scène

-EXP. ERR : Erreur exponentielle d'un classificateur fort

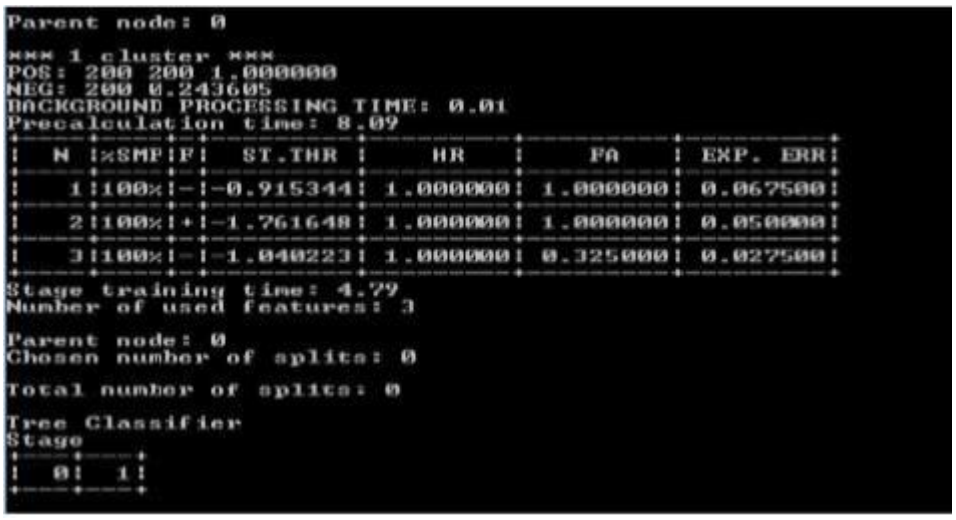


Figure 9 : Classifications des images positives

- Etape 6 : **création du XML file**

A la fin de l'apprentissage, un catalogue noté de 0 jusqu'à "N-1" est créé, avec N le nombre des étapes définies dans HaarTraining. Dans ces catalogues nous devons avoir un fichier texte AdaBoostCARTHaarClassifier.txt file. Tous les classificateurs faibles sont combiné dans un seul classificateur fort enregistré dans un fichier XML file. Le fichier A_geste.xml est le fichier de sortie représenté par 24x24 comme dimension de la main sur une image.

4. Test et résultats

4.1. Première configuration

Dans la figure suivante, nous allons présenter le résultat de la détection de la main avec une première implémentation. Nous nous servirons d'un code python [s2] pour la détection en temps réel.

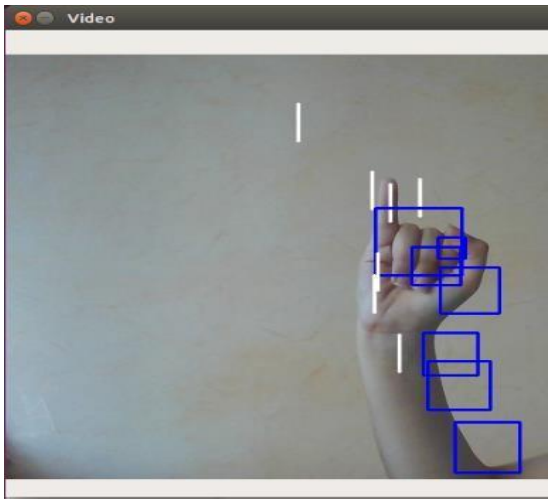


Figure 10 : Echec de détection

4.2. Deuxième configuration

Après une optimisation du résultat, les tests de la deuxième implémentation ont été effectués par le même code python [s2].

a) Test du mouvement

Le programme détecte bien le mouvement d'une seule main (à gauche) et de deux mains (à droite).

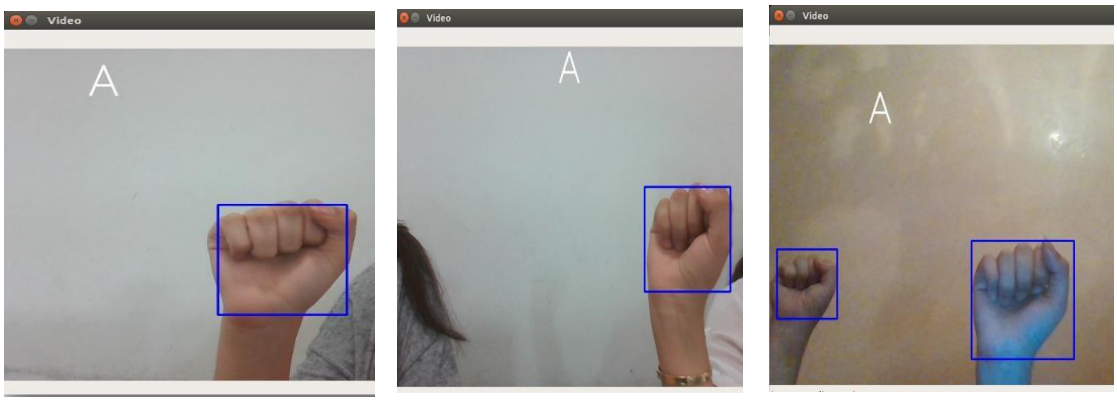


Figure 11 : détection d'un mouvement

b) Test à différentes conditions d'illumination

On remarque que l'algorithme d'un même geste de la main donne des résultats différents ; la zone détectée est plus large que celle détectée en figure 12, ce résultat est dû au changement de l'éclairage.

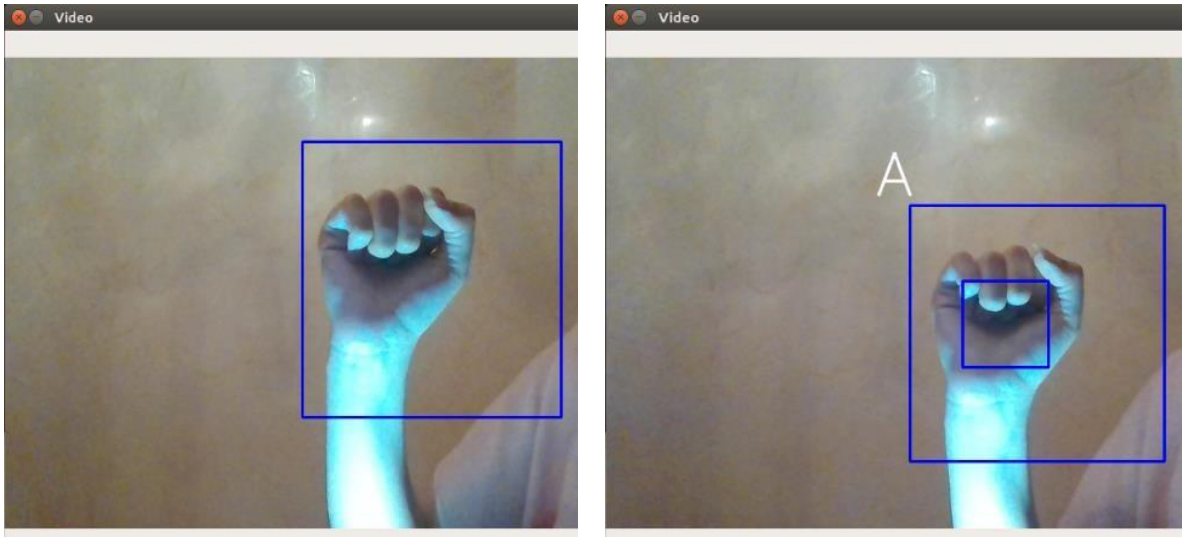


Figure 12 : Résultat de la détection a différentes conditions d'illumination

c) Test à un mouvement incliné

Le programme ne fonctionne pas quand la main présentée est inclinée latéralement ou verticalement. Ce problème est dû à ce que toutes les mains utilisées comme références se positionnent directement devant l'appareil photo.

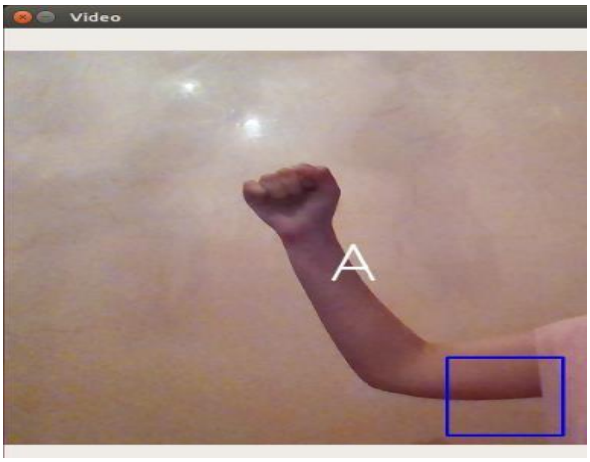


Figure 13 : Problème d'inclination de la main

- On peut remarquer d'après la figure 13 qu'il s'agit d'une détection fausse positive, c'est le résultat d'une prise de décision dans un choix à deux possibilités (positif et négatif), déclaré positif, là où il est en réalité négatif. Généralement, les faux positifs apparaissent dans des zones ou couleur est presque exacte à celle de la main.

4.3. Comparaison

Configuration	Echantillons positifs	Images négatives	MinHitRate
No 1	2000	1900	0,80
No 2	5000	2000	0,99

Tableau 1 : Résultat de deux configurations

D'après le tableau ci-dessus, on remarque que

- Augmenter le nombre de l'échantillon positif et d'image négative permet d'améliorer la précision de la détection.
- Plus le MinHiteRate est élevé, plus le taux de détection augmente.
  Le MinHiterate est le paramètre qui nous assure que nos données d'entraînement positives donnent au moins un rendement de détection décent, la valeur de 0,8 signifierait que 20 % de nos données de formation sur les objets positifs peuvent être mal classées, ce qui serait un désastre.
- Si le classificateur faible est assez grand, le classificateur fort produit aura un taux d'erreur extrêmement faible.

5. Conclusion

Dans ce chapitre nous avons présenté l'implémentation de la méthode de Viola et Jones en utilisant le langage de programmation Python sous l'environnement Linux et utilisé les bibliothèques de l'OpenCV. Nous avons testé les résultats obtenus en temps réel.

Conclusion générale

La reconnaissance gestuelle est une tâche très difficile et intéressante dans la vision informatique, non parfaitement résolu, malgré tous les travaux réalisés au cours des dernières années.

Les méthodes de détection sont nombreuses, le choix d'une méthode doit être basé sur les conditions spécifiques de chaque application. Pour y parvenir, nous avons mis en œuvre l'apprentissage par algorithme d'AdaBoost proposé par Viola et Jones, qui consiste à créer un classificateur de fonctionnalités de type Haar. .

La méthode de Viola et Jones largement reconnue comme méthode fonctionnant en temps réel et fournissant des résultats robustes et fiables

Durant notre étude, nous avons obtenu des résultats significatifs. Ces résultats ont été présentés et interprétés pour montrer l'efficacité de la méthode proposée. Néanmoins, nous suggérons une bonne position devant la caméra pour une bonne détection de la main mais aussi l'environnement bien éclairé.

Le succès que nous avons eu durant cette expérience nous donne la possibilité de suggérer aux futures étudiants le développement des applications de détection de la main et principalement celles basées sur l'algorithme d'AdaBoost et peut être les marier avec une ou plusieurs autres méthodes de détection pour obtenir une application de traitement de multimodale qui marque à nos jours une importance dans le domaine de recherche.

Bibliographie

[1] https://fracademic.com/dic.nsf/frwiki/1862108 visité le 17/04/2019

[2] Paul Viola and Michael Jones, Robust real-time face detection, International Journal of Computer Vision, **57**:137–154, 2004.

[3] https://docplayer.fr/12553079-Le-traitement-d-images-detection-d-objets-par-le-classificateur-de-haar.html visité le 30/05/2019

[4] Paul Viola et Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, IEEE CVPR, 2001

[5] https://fr.wikipedia.org/wiki/Raspbian visité le 21/05/2019

[6] Traitement de l'image et de la vidéo [archive] Editions : Ellipses - Technosup (2010)

[7] https://www.raspberrypi.org/downloads/raspbian/ visité le 21/05/2019

[8] https://www.etcher.io/ visité le 22/05/2019

ANNEXE

[s1] :

```
import urllib.request
import cv2
import numpy as np
import os
def neg():
    for file_type in ['negatives']:
        for img in os.listdir(file_type):
            line = file_type+'/'+image+'jpg'+'\n'
            with open('negatives.dat','a') as f:
                f.write(line)
create neg()
```

[s2] :

```
import cv2
handCascade = cv2.CascadeClassifier('A_geste.xml')
vs = cv2.VideoCapture(0)
while True:
        ret, frame = vs.read()
        if frame is None:
                break
        hand = handCascade.detectMultiScale(frame)
        for (x, y, w, h) in hand:
            font = cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(frame, 'A', (x-w,y-h), font, 2.0, (255,255,255), 2, cv2.LINE_AA)
            cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        cv2.imshow("Video", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
                break
cv2.destroyAllWindows()
```