الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEURET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR. YAHIA FARES DE MEDEA
FACULTE DE TECHNOLOGIE

Département du Génie Électrique



Projet de fin d'études

Présenté pour l'obtention du Diplôme De Master en Génie Electriques

Filière :Automatique

Spécialité : Automatique et systèmes

THEME
Guidage intelligent d'un robot mobile Dans un parcours balisé

Proposé et dirigé par:

Dr. Kara Reda
Mr. Benyezza Hamza

Préparé par:
M. Tassist Mohamed Amin

Année Universitaire 2020-2021

Remerciement

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la Patience d'accomplir ce modeste travail.

En second lieu, nous tenons à remercier notre encadreur **Dr. KARA Reda** et **Mr.Benyezza** hamza pour leurs précieux conseils et leurs aides durant toute la période du travail.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail Et de l'enrichir par leurs propositions.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Dédicace

Je dédie ce modeste travail à mes chers parents qui m'ontbeaucoup encouragé et soutenu pendant tout leur vie pour voir ce jour-là et pour leurs précieux conseils ainsi que leurs recommandations.

A mes très chers frères Isleme, Othamen et mes chères sœurs Malika et Khadija

A toute ma famille, grande et petite sans exception.

A tous mes amis Mohri , Hadjameur ,Benyahia , Takdenti , ben mouloud, Amel, Chaima et Aya

Mohamed amine

ملخص

يهدف المشروع ألى أجراء المراقبة المرئية للروبوت المتحرك للقيام بذلك قمنا بالنمذجة كل من كاميرا وبطاقة راسبري مع بطاقة الأردو ينو ، كما قمنا بإدخال الذكاء الاصطناعي الذي يقوم بتحكم في الربوت من خلال إشارات المرور ، كما قمنا باستعمال مفتاح ذكاء الاصطناعي لتسريع عملية علاج صور التي يقوم من خلالها معرفة نوع الإشارة.

الكلمات المفتاحية : الروبوتات المتحركة, راسبري باي, الأردو ينو, الذكاء الاصطناعي.

.

Résumé :

Le projet vise à effectuer un asservissement visuel dans un robot mobile. Pour le faire, nous avons utilisé la caméra PIXY est le capteur de trajectoire, la carte Raspberry est le cerveau du robot et la carte Arduino pour contrôler les moteurs, pour bien guider le robot mobile selon les plaques de signalisation, nous avons intégré l'intelligence artificielle pour la reconnaissance de ces plaques, et nous avons aussi utilisé un la clé d'accélération matérielle NCS2 pour accélérer la sortie du notre modèle d'intelligence artificielle pour connaître le type de plaque.

Mots-clés : Robots Mobiles, Raspberry Pi, Arduino, Intelligence Artificielle.

Summary:

The project aims to perform visual servoing in a mobile robot. To do this, we used the PIXY camera is the trajectory sensor, the Raspberry board is the brain of the robot and the Arduino board to control the motors, to guide the mobile robot well according to the signage plates, we have integrated the artificial intelligence for the recognition of these plaques, and we also used a hardware acceleration key NCS2 to speed up the release of our artificial intelligence model to know the type of plaque.

Key words: Mobile Robots, Raspberry Pi, Arduino, Artificial Intelligence.

Table des matières

Remerciement.....

Dédicace.....

Table des matières.....

Listes des figures.....

Notation.....

Introduction Générale

1 Introduction générale..... 1

Chapitre I : Robotique et asservissement Visuel

1. Introduction 4

2. Aperçu historique 4

3. Robot mobile..... 5

3.1 Définition..... 5

3.2 L'architecture globale d'un robot mobile..... 6

3.3 Structure mécanique et la motricité..... 6

3.4 Comparaison des différents types 8

3.5 Système de sécurité 10

3.6 Traitement des informations et gestion des taches 11

4. Applications..... 11

4.1 Robots domestiques ou ménagers..... 12

4.2 Robots en médecine et chirurgie 12

5. Asservissement visuel 13

5.1 Asservissement visuel 3D..... 13

5.2 Asservissement visuel 2D..... 14

6. Conclusion..... 15

Chapitre II : Les réseaux de neurones artificiels et le Deep learning.

1. Introduction 17

2. L'intelligence artificielle 17

2.1 Définition de l'Intelligence Artificielle (IA) 17

2.3. Fonctionnement de l'Intelligence Artificielle 18

3. Deep Learning 18

4. Réseaux de neurones 19

4.1. Modèles de réseaux de neurones 19
4.2. Apprentissage d'un réseau de neurones 24
5. Réseaux de neurones convolutionnels (CNN) 26
5.1. Les couches de réseaux de neurones convolutionnels 26
5.2. Choix des paramètres 30
5.3. Avantages de CNN 31
6. Conclusion 31

Chapitre III : Matériels et systèmes embarqués utilisés

1. Introduction 33
2. Neural Compute Stick 2 d'Intel 33
2.1. Distribution Intel® d'OpenVINO™ toolkit 34
2.2. Inférence du Deep Learning 34
3. Raspberry Pi 3 model B+ 36
3.1. l'architecture 37
3.2. Programmation 37
4. La carte Arduino Mega 37
4.1. Présentation 38
4.2. Spécifications techniques 38
4.3. L'extension (Shield) de commande des moteurs 39
5. La caméra Pixy 41
5.1. Aperçu 41
5.2. Simplicité de mise en œuvre du système de vision 42
5.3. Support du contrôleur 42
5.4. La détection d'objets 42
5.5. L'application PixyMon 43
5.6. Spécifications techniques 43
6. La Webcam HD C270 44
7. Environnements et bibliothèques 45
7.1. Anaconda (Python Distribution) 45
7.2. SPYDER 46
7.3. TensorFlow 46
7.4. NumPy 47

7.5. OpenCV…. 47
7.6.Keras…… .47
8. L'application Cascade 48
9. Conclusion 48

Chapitre IV : Guidage intelligent du robot

1.Introduction 50
2.Description générale 51
.3Création du jeu de plaques signalisation à respecter 60
4.Choix de l'architecture du réseau de neurones CNN 61
5. Classification par l'application Cascade 63
6. Installation de la clé Neural Compute Stick 2 d'Intel 65
7. Conversion et chargement du modèle CNN dans la clé 67
8. Résultats comparatifs 67
9. Conclusion 67

Conclusion générale

1. Conclusion général 70

Listes des figures

Figure I.1 : La tortue de Grey Walter……. 5

Figure I.2 : Robot de type uni-cycle…… 6

Figure I.3 : Robot de type tricycle……... 7

Figure I.4 : Robot de type voiture…… 7

Figure I.5 : Robot de type omnidirectionnel…..……... 8

Figure I.6 : Robot à chenille…… 9

Figure I.7 : Robot marcheur quasi-statique……. 9

Figure I.8 : Robot marcheur dynamique …… 10

Figure I.9 : Robot rampant…….. 10

Figure I.10 : Synoptique de la sécurité…… 11

Figure I.11 : Robot industriel (bras manipulateur)…….. 12

Figure I.12 : Robot domestique (aspirateur)…….……... 12

Figure I.13 : Robot médical (Da Vinci)…….. 13

Figure I.14 : Principe général de l'asservissement…….. 13

Figure I.15 : Principe général de l'asservissement 3D…… 14

Figure I.16 : Principe général de l'asservissement 2D……..... 15

Figure II.1 : Schéma d'un réseau de neurones artificiel…….. 19

Figure II.2 : différents types de réseaux de neurones…….……. 20

Figure 0.3 : Perceptron monocouche…….. 21

Figure 0.4 : Perceptron multicouche……... 22

Figure II.5 : Schéma d'un RBF…….……..... 23

Figure II.6: Architecture standard d'un réseau de neurone convolutionnel…….. 24

Figure 0.7 : Apprentissage supervisé. …… 25

Figure II.8 : Exemple de schéma de principe du CNN …… 26

Figure II.9 : Les couches de CNN ……... 27

Figure II.10 : Exemple d'architecture CNN ……..... 27

Figure 0.11 : Exemple de principe du filtre convolutionnel …..…… 28

Figure II.12 : Principe de la fonction ReLu…….**.....** 28

Figure 0.13 : Exemple de principe du Pooling …… 29

Figure II.14 : Principe de la couche entièrement connectée …….. 29

Figure II.15 : Principe de la couche Logistique ou Softmax…….. 30

Figure II.16 : Exemple montrant l'étiquette codée de la couche de sortie CNN …….. 30

Figure III.1 : Neural Compute Stick 2……..…..……..……..…… 33

Figure III.2 : Diagramme de déroulement des opérations d'OpenVINO toolki[…… ..……. 35

Figure III.3 : Raspberry pi 3 B+……..……...……..…….. 36

Figure III.4 : distribution des broches entrée sortie……..……..…… 37

Figure III.5 : la carte Arduino Mega……. 38

Figure III.6 : Description de la carte arduino Mega**……**……..……..……..……. 38

Figure III.7: Motor Shield…….. 40

Figure III.8 : fonctionnement de L293D……..... 41

Figure III.9 : Structure de la Cam Pixy CMUcam5……..……..……..…… 44

Figure III.10 : Webcam HD C270……..…….. 45

Figure III.11 : logo de la bibliothèque Anaconda……..…….**....** 45

Figure III.12 : logo de la bibliothèque Spyder ..……..…….**....** 46

Figure III.13 : logo de la bibliothèque TensorFlow…..……. 46

Figure III.14 : logo de la bibliothèque NumPy…....……..……**...** 47

Figure III.15 : logo de la bibliothèque OpenCV…..……..…… 47

Figure III.16 : logo de la bibliothèque Keras…..……..……. 47

Figure III.17 : photo de logiciel Cascade……..…….. 48

Figure IV.1: Schéma Synoptique du robot mobile…….. 51

Figure IV.2 : vue du robot mobile……... 52

Figure IV.3 : vue du robot mobile dans son environnement…….. 53

Figure IV.4 : Synoptique de l'étage asservissement visuel du robot……... 53

Figure IV.5 : Synoptique de notre schéma d'asservissement visuel …….. 54

Figure IV.6 : Organigramme de commande des moteurs du robot……. 55

Figure IV.7 : capteur LDR……... 56

Figure IV.8 : Éclairage de la trajectoire 57

Figure IV.9 : Organigramme de commande des moteurs du robot……. 58

Figure IV.10 : Sélection des plaques dans les images …… 59

Figure IV.11 : Sélection de la plaque pertinente dans une image contenant 2 plaques ……. 59

Figure IV.12: le fichier associant les plaques aux classes……... 60

Figure IV.13: classification des plaques de signalisation 60

Figure IV.14 : Les couches de CNN……... 61

Figure IV.15 : les fichiers dataset de cascade …… 63

Figure IV.16: enregistrement du ficher du jeu de données dans logiciel…… 63

Figure IV.17 : Les paramétrage de l'application……. 64

Figure IV.18: Résultat après le traitement……... 64

Figure IV.19: placer le clé sur La carte Raspberry pi 3……... 65

Figure IV.20 : résultat de confirmation d'installation de la clé……... 66

Figure IV. 21: conversion du modèle du réseau CNN…… 67

Liste des tableaux

Table I.1 : Comparaison des différents types de robot à roue……	8
Table IV.1 : les tests de l'architecteur de CNN……...	62
Table IV.2 : les tests de la clé et Raspberry pi 3……	67

Notation

3D : signifie trois dimensions. Ces dimensions permettent de définir un objet possédant trois axes x, y et z, associés respectivement à la longueur, la hauteur et la profondeur.

2D : signifie deux dimensions. Ces dimensions permettent de définir un objet possédant deux axes x, y, associés respectivement à la longueur, la hauteur

CNN : Réseaux de neurones convolutionnels

AI : L'intelligence artificielle

Introduction Générale

1. Introduction générale

Les robots mobiles jouent un rôle très important dans le domaine industriel et scientifique et notre vie moderne et ils ont utilisés dans tous les domaines et ils ont devenus indispensables dans les usines, les boulangeries et les chantiers et on ne peut pas imaginer notre future sans les robots.

La navigation d'un robot mobile peut généralement être définie comme le processus de déterminer un chemin approprié entre un point de départ et un point d'arrivé et permettre au robot de le suivre. En particulier, dans les dernières décennies, la navigation visuelle des robots mobiles est devenue une source de contributions à d'innombrables recherches. Avec vision, le robot est capable de percevoir son environnement en recevant suffisamment d'informations lui permettant de se localiser et trouver des chemins efficaces et sécuritaires de manière autonome afin d'éviter des collisions et arriver à sa destination. L'asservissement visuel constitue le processus de contrôle basé sur les informations fournies par une ou plusieurs caméras.

L'intelligence artificielle est largement utilisée dans le domaine des robots mobile, surtouts par la reconnaissance visuelle qui donne au robot une autonomie qui luis permettre de se localiser et trouver sa voie en interagissant avec son environnement, on prend l'exemple de la détection de la ligne de voie et l'interaction avec les plaques de signalisations routières ; avec des programmes informatiques complexes et des informations fournies par l'administrateur, le robot va être capable de traiter et de raisonner d'une façon automatique et autonome sans intervention humaine.

Ce document est organisé en quatre chapitres qui portent sur les aspects suivants :

Le premier chapitre : présent une introduction à la robotique mobile, Structure mécanique et la motricité et les types de robots, asservissement visuel

Le deuxième chapitre : est consacré à la présentation de l'intelligence artificielle, les réseaux de neurones en général et les réseaux de neurones convolutionnels CNN en particulier.

Le troisième chapitre : est consacré aux systèmes embarqués utilisés dans notre projet, tel que le système de traitement la clé Neural Compute Stick 2 d'Intel, la carte Raspberry Pi3, la carte Arduino Mega ainsi que le système de vision adopté.

Le quatrième chapitre : est une présentation du fonctionnement détaillé et des modifications majeures apportées à la plate-forme robot mobile pour réaliser le suivi d'une trajectoire et la détection de plaques de signalisation qui balises ce parcours à partir de données

visuelles uniquement. Pour atteindre cet objectif, nous présenterons les techniques de l'intelligences artificielles que nous avons utilisées

Chapitre I :
Robotique et asservissement Visuel

1. Introduction

La robotique a atteint son plus grand succès jusqu'à ce jour dans le monde de la fabrication industrielle.

Boulonné à son épaule à une position spécifique dans la ligne d'assemblage, le bras de robot peut se déplacer avec une grande rapidité et précision pour effectuer des tâches répétitives telles que le soudage par points et la peinture. Dans l'industrie de l'électronique, les manipulateurs place des composants montés en surface avec une précision surhumaine, ce qui rend le téléphone portable et ordinateur portable possible.

Cependant, pour tous leurs succès, ces robots commerciaux souffrent d'un inconvénient majeur : le manque de mobilité. Un manipulateur fixe dispose d'une gamme limitée de mouvement qui dépend de l'endroit où elle est boulonnée. En revanche, un robot mobile serait en mesure de voyager à travers l'usine de fabrication, en appliquant avec souplesse ses talents partout où il est le plus efficace.

De manière générale, on regroupe sous l'appellation robots mobiles l'ensemble des robots à base mobile, par opposition notamment aux robots manipulateurs. L'usage veut néanmoins que l'on désigne le plus souvent par ce terme les robots mobiles à roues. Les autres robots mobiles sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient marcheurs, sous- marins ou aériens.

Le but de ce chapitre est de situé le robot mobile objet de notre étude. A partir des descriptions des différents éléments constituants les robots mobiles ainsi que les différentes structures mécaniques qui existent.

Par ailleurs, nous introduirons les notions de l'asservissement visuel. Cette technique sera utilisée pour la commande de notre robot mobile.

2. Aperçu historique

Ce n'est qu'au tout début du XX^{e} siècle que les robots firent leur apparition, suite aux travaux d'ingénieurs qui voulaient tester des hypothèses émises par des biologistes et des psychologues. Le chien électrique conçu par Hammond et Miessner en 1915 était attiré par une lumière, selon le phototropisme animal mis en évidence par Loeb en 1918. Les machines de Russell (1913) et de Stephens (1929), les tortues cybernétiques de Grey Walter (Figure I.1)(1950), le renard électronique

de Ducrocq (1953) ou l'homéostat d'Ashby (1952) étaient, elles, dotées de capacités d'apprentissage directement issues des travaux des psychologues Thorndike (1911), Hull (1943) et du physiologiste Pavlov (1903) sur l'Homme et l'animal.

Ces réalisations sont des robots, car elles ne se comportent plus comme de simples automates dont les organes moteurs - leurs mécanismes - obéissent à un programme préétabli. À la différence des automates, ils ont des organes sensoriels - les capteurs - qui recueillent des informations de l'environnement qui vont, elles, influencer l'activité de leurs organes moteurs -les actionneurs[1].



Figure I.1 : La tortue de Grey Walter.

3. Robot mobile

3.1 Définition

Ecrivaient chèque, Karel Capek, dans son drame, introduit le mot robot au monde en1921.Il est dérivé du mot tchèque robot qui signifie "travailleur forcé"

Un "robot" est une machine pouvant manipuler des objets en réalisant des mouvements variés dictés par un programme aisément modifiable.

Programmer un robot consiste, dans un premier temps, à lui spécifier la séquence des mouvementsqu'il devra réaliser.

3.2 L'architecture globale d'un robot mobile

L'architecture des robots mobiles se structure en quatre éléments :

- La structure mécanique et la motricité
- Les organes de sécurité
- Le système de traitement des informations et gestion des tâches

3.3 Structure mécanique et la motricité

- **Robots à roues**

Il existe plusieurs classes de robots à roues déterminées, principalement, par l'apposition et le nombre de roues utilisées.

Nous citerons ici les quatre classes principales de robots à roues.

- **Robot uni-cycle**

Un robot de type uni-cycle est actionné par deux roues indépendantes, il possède éventuellement des roues folles pour assurer sa stabilité. Son centre de rotation est situé sur l'axe reliant les deux roues motrices.

C'est un robot non-holonome, en effet il est impossible de le déplace dans une direction perpendiculaire aux roues de locomotion.

Sa commande peut être très simple, il est en effet assez facile de le déplacer d'un point à un autre par une suite de rotations simples et de lignes droites.

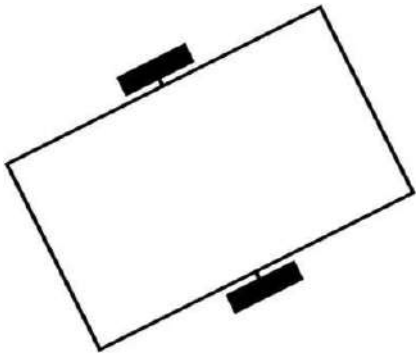


Figure I.2 : Robot de type uni-cycle

- **Robot tricycle**

Un robot de type tricycle est constitué de deux roues fixes placées sur un même axe et d'une roue

centrée orientable placée sur l'axe longitudinal.

Le mouvement du robot est donné par la vitesse des deux roues fixes et par l'orientation de la roue orientable. Son centre de rotation est situé l'intersection de l'axe contenant les roues fixes et de l'axe de la roue orientable

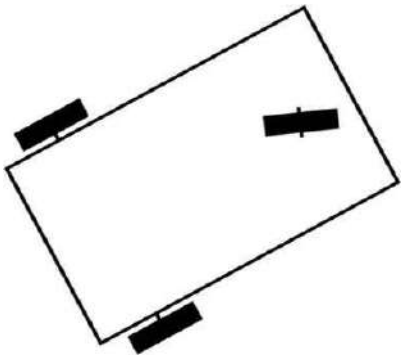


Figure 0I.3 : Robot de type tricycle

- **Robot voiture**

Un robot de type voiture est semblable au tricycle, il est constitué de deux roues fixes placées sur un même axe et de deux roues centrées orientables placées elles aussi sur un même axe. Le robot de type voiture est cependant plus stable puisqu'il possède un point d'appui supplémentaire.

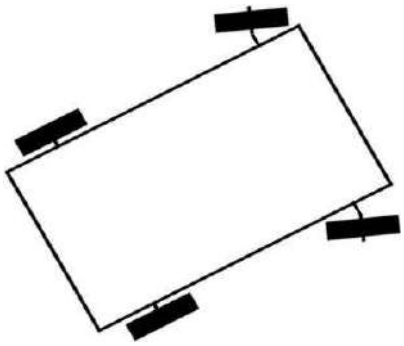


Figure I.4 : Robot de type voiture

- **Robot omnidirectionnel**

Un robot omnidirectionnel est un robot qui peut se déplacer librement dans toutes les directions. Il est en général constitué de trois roues décentrées orientables placées en triangle équilatéral.

L'énorme avantage du robot omnidirectionnel est qu'il est holonome puisqu'il peut se déplacer dans toutes les directions. Mais ceci se fait au dépend d'une complexité mécanique bien plus grande.

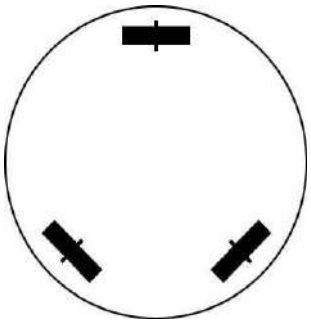


Figure I.5 : Robot de type omnidirectionnel

3.4 Comparaison des différents types

Nous pouvons observer dans le tableau ci-dessous un récapitulatif des avantages et des inconvénients des différents types de robots à roues.

Robot uni cycle	- non-holonome + stable + rotation sur soi-même + complexité mécanique faible
Robot tricycle	-non-holonome -peu stable -pas de rotation sur soi-même + complexité mécanique modérée
Robot voiture	-non-holonome + stable -Pas de rotation sur soi-même +complexité mécanique modérée
Robot omnidirectionnel	+ holonome + stable + rotation sur soi-même - complexité mécanique modérée

Table I.1 : Comparaison des différents types de robot à roue

- **Les robots à chenilles**

Les robots avec des chenilles sont pratiques pour de nombreuses fonctions comme la capacité à tondre au milieu de toutes sortes d'obstacles, comme des rochers, des fossés et des nids de poule. Compte tenu de la matière des chenilles, les robots à chenilles peuvent se déplacer sur des surfaces

glissantes comme la neige, le béton humide, ou un carrelage de cuisine propre.



Figure I.6 : Robot à chenille

- **Les robots à pattes**

Le robot à pattes se déplace la plupart du temps selon une marche qui peut être quasi- statique ou dynamique. Il devient dès lors 'robot marcheur'. Il n'en demeure pas moins que certains robots à pattes se déplacent sous la forme de 'sauts'

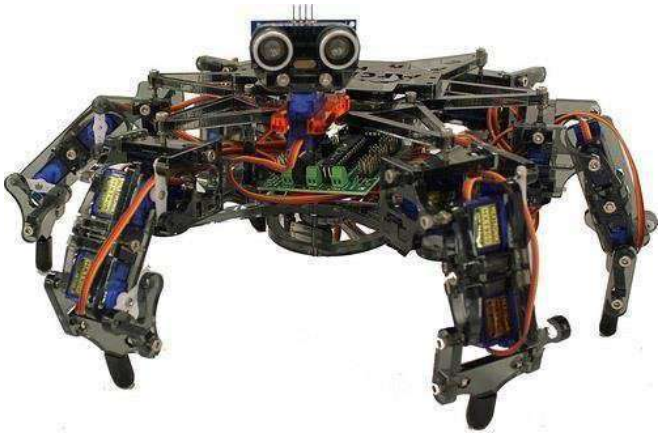


Figure I.7 : Robot marcheur quasi-statique

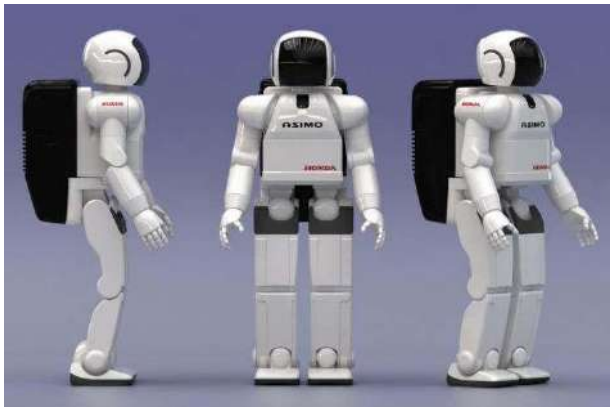


Figure I.8 : Robot marcheur dynamique

- **Les robots rampant**

On peut citer tout d'abord un robot dont le déplacement copie celui des amibes qui font circuler des fluides à l'intérieur de leur corps. L'appareil créé par l'homme est doté d'anneaux se contractant et se décontractant provocant le déplacement sans pour autant être une vrai reptation



Figure I.9 : Robot rampant

3.5 Système de sécurité

Un robot, selon la tâche qui lui est confiée, peut être amené à travailler au voisinage du personnel. A ce titre, il est obligatoire qu'il soit doté d'organes garantissant la sécurité. Des capteurs sont disponibles tout auteur du mobile afin de détecter un obstacle sur un domaine le plus étendu possible. Deux types de capteurs sont employés : les capteurs périmétriques assurant la détection avant collision (ultra-son, hyper fréquence, infrarouge…) et les capteurs de contact détectant une collision ou un choc avec l'environnement (contact électrique sur pare- chocs, résistance variable, fibre optique…). Ce sont des dispositifs redondants par rapport aux capteurs précédents.
L'organisation de la sécurité est représentée sur les schémas de la Figure I.9.

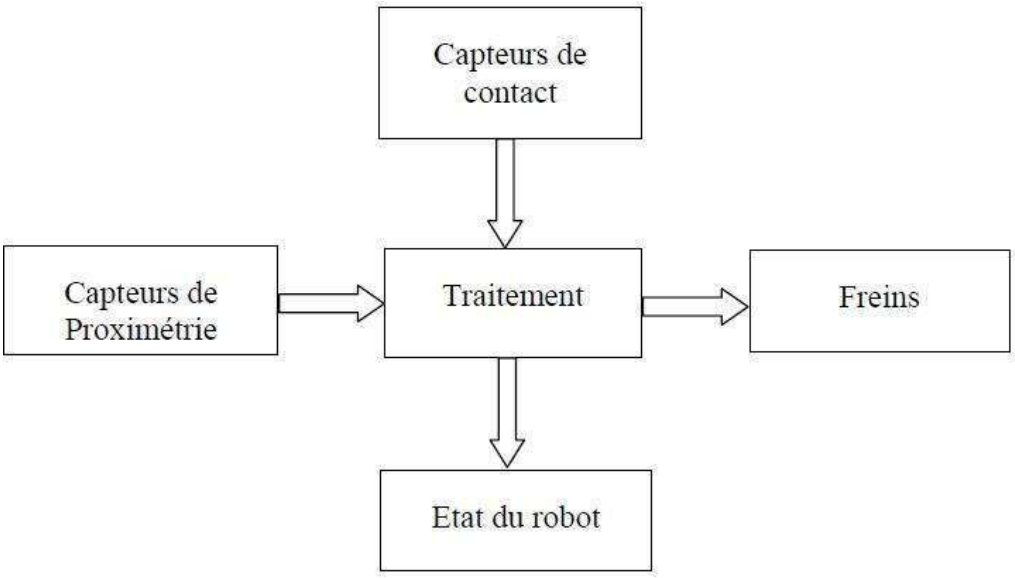


Figure I.10 : Synoptique de la sécurité

3.6 Traitement des informations et gestion des taches

L'ensemble de traitement des informations et gestion des tâches constitue le module information central qui établit les commandes permettant au mobile de réaliser un déplacement et d'activer les divers organes en accord avec l'objectif. Nous nous limiterons au problème de génération de plan qui consiste à établir la manière dont le robot se déplace par rapport à des connaissances à priori (statiques) ou obtenues en cours d'évolution (dynamiques). La génération de plus repose sur trois concepts :

- La stratégie de navigation
- La modélisation de l'espace
- La planification.

4. Applications

On cite ici quelque exemple des applications des robots mobiles ceux utilisés dans un environnement de fabrication industrielle. Ils sont utilisés dans la fabrication des automobiles, des composants et des pièces électroniques, des médicaments et de nombreux produits



Figure I.11 : Robot industriel (bras manipulateur)

4.1 Robots domestiques ou ménagers

Robots utilisés à la maison. Ce type de robots comprend de nombreux appareils très différents, tels que les aspirateurs robotiques, robots nettoyeurs de piscines, balayeuses, nettoyeurs gouttières et autres robots qui peuvent faire différentes tâches. En outre, certains robots de surveillance et de télé présence pouvaient être considérées comme des robots ménagers se il est utilisé dans cet environnement



Figure I.12 : Robot domestique (aspirateur)

4.2 Robots en médecine et chirurgie

Les robots semblent avoir de l'avenir à l'hôpital. Robodoc aide à réaliser certaines opérations de chirurgie. Le robot infirmier est encore en projet. Le cyber squelette HAL aide les personnes à se déplacer. Et le robot patient permet aux futurs chirurgiens-dentistes d'apprendre à soigner sans faire de dégâts...



Figure I.13 : Robot medical (Da Vinci)

5. Asservissement visuel

Le principe de l'asservissement visuel consiste à prendre en compte des informations visuelles issues d'une ou plusieurs caméras dans la boucle de commande d'un robot afin d'en contrôler le mouvement.

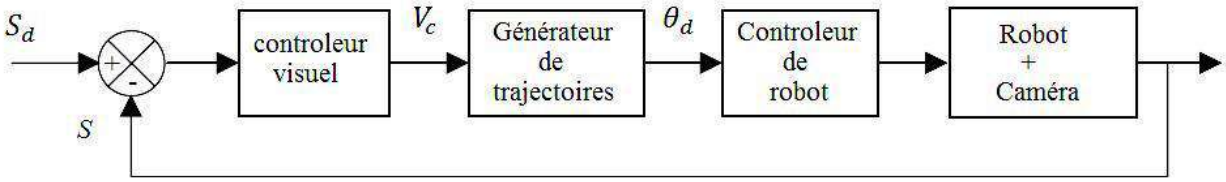


Figure I.14 : Principe général de l'asservissement

La vision industrielle est constituée de deux grands principes d'asservissement qui réponde tous deux à des besoins et critères différents.

5.1 Asservissement visuel 3D

Pour ce type d'asservissement, la référence est exprimée sous la forme d'une attitude. Dans notre cas, il s'agit de l'attitude « **r** » d'un repère lié à l'objet vu par la caméra par rapport à un repère lié à l'organe terminal du robot (voir ci-dessous).

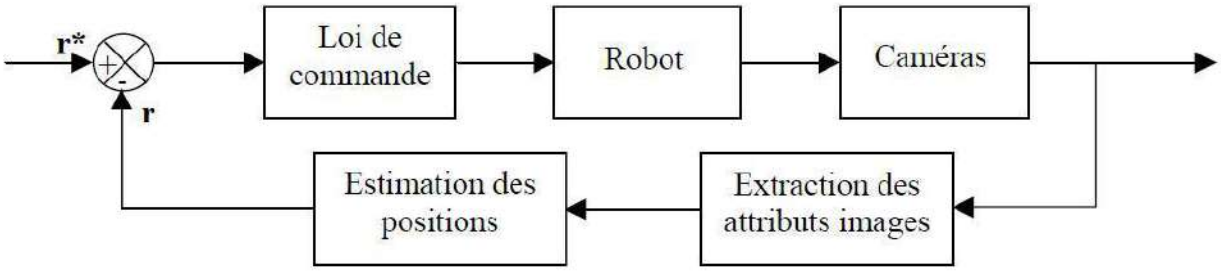


Figure I.15 : Principe général de l'asservissement 3D

Cette mesure s'obtient grâce aux primitives extraites de l'image et à un modèle géométrique de la cible, mais elle est très sensible aux incertitudes sur ce modèleainsi qu'aux erreurs de calibrage de la caméra. Néanmoins, ce problème peut être contourné par la boucle fermée, ce qui nous évite les problèmes de convergence du système.

De nombreux articles traitent des techniques d'estimation de l'attitude entre une ou plusieurs caméras et un objet dont la géométrie est connue. La complexité des calculs nécessaires à cette estimation croît avec la complexité de l'objet. C'est pourquoi les objets sont souvent simplifiés au maximum dans les asservissements visuels.

Il a été montré qu'il est nécessaire de connaître la projection d'au moins 4 primitives de type "point" pour pouvoir obtenir cette attitude de manière unique. Si seulement trois projections sont disponibles, l'attitude ne peut être déterminée de manière unique. Le problème de la reconstruction de l'attitude à partir de la projectionde 3 ou 4 points est abordé par de nombreux auteurs.

5.2 Asservissement visuel 2D

Alors que l'asservissement visuel 3D, implique de connaître un modèle tridimensionnel de la cible et de pouvoir la situer dans un repère commun, et de définir une consigne d'asservissement en termes de position et orientation,l'asservissement visuel 2D travaille au niveau de l'image, avec des primitives visuelles (indices) définis au préalable. C'est dans l'image que la reconnaissance de l'objet par ses indices a lieu, et c'est toujours dans l'image que la consigne que doivent atteindre les indices est définie, comme il est indiqué sur la figure suivante.

L'algorithme n'a donc pas besoin de connaître la structure 3D de la scène, les calculs exploitent seulement l'image. Une primitive est une forme géométrique

Élémentaire (point, segment de droite, portion d'ellipse,...). Elle sert à modéliser la projection d'un objet dans le plan image. La plupart des travaux qui traitent de l'asservissement 2D utilisent des primitives constituées de points.

Ces points peuvent par exemple être situés à l'intersection de segments dans l'image ou encore

être extraits du centre de gravité de la projection de disques. En fait, les cibles les plus couramment rencontrées sont simplement constituées de plusieurs disques coplanaires. De telles cibles ont l'avantage de nécessiter un tempsde traitement d'image faible.

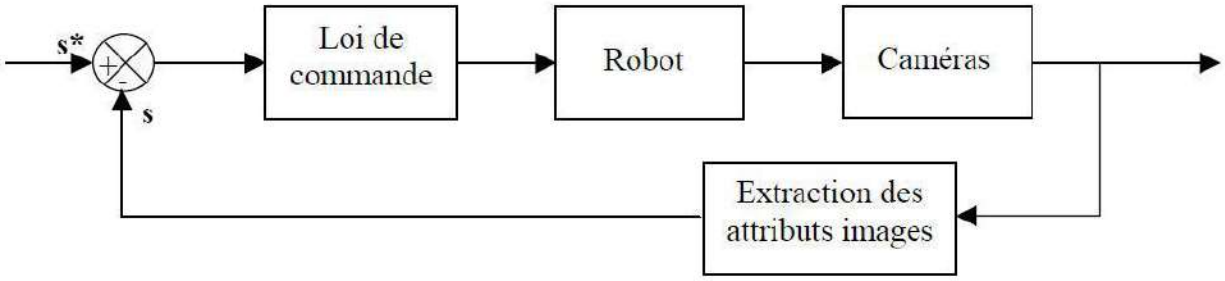


Figure 1.16 : Principe général de l'asservissement 2D

6. Conclusion

La présentation faite dans cette première partie nous a permis de décrire les éléments constituants d'un robot mobile. A partir de ce descriptif, nous avons constaté que pour réaliser la commande de ce type de robot, nous devions répondre à trois impératifs à savoir la perception, la décision et l'action.

L'exploitation d'informations issues de capteurs extéroceptifs permet d'envisager la réalisation de tâches robotiques complexes. Dans le cas de la robotique mobile, le traitement de toutes ces informations nécessite une puissance de calcul importante. Par ailleurs, l'autonomie de ce type de robot impose l'utilisation d'algorithmes de contrôle robustes et adaptatifs. C'est pour cela que nous consacrerons, le prochain chapitre, à la description des réseaux de neurones artificiels.

Chapitre II :
Les réseaux de neurones artificiels et le Deep learning.

1. Introduction

Les réseaux neuronaux artificiels (RNA) sont parmi les techniques récentess utilisées dans la reconnaissance et la classification des objets. Ils peuvent modéliser et approximer des systèmes complexes et difficiles, en utilisant une modélisation conventionnelle mathématique.

La classification automatique des images consiste à attribuer automatiquement une classe à une image à l'aide d'un système de classification. On retrouve ainsi la classification d'objets, de scènes, de textures, la reconnaissance de visages, d'empreintes digitale et de caractères.

Dans ce chapitre, une description générale des réseaux neuronaux artificiels et des aspects liés à leur fonctionnement sera entreprise. Nous présenterons des généralités sur le réseau neuronal, telles que la définition de ses structures, les fonctions d'activation et les différentes architectures de ces réseaux.

2. L'intelligence artificielle

2.1 Définition de l'Intelligence Artificielle (IA)

Selon le dictionnaire Larousse, l'intelligence Artificielle se définit comme étant «l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence.»

Ce serait, de ce fait, des ordinateurs ou des machines dotées de programmes capables de performances similaires à l'intelligence humaine, ou même, amplifiées par la technologie. Ces machines sont en mesure de :

- Raisonner.
- Traiter de grandes quantités de données.
- Discerner des modèles indétectables par l'œil d'un humain.
- Comprendre et analyser ces modèles.
- Interagir avec l'Homme.
- Apprendre progressivement.
- Améliorer continuellement ses performances.

Depuis 1950, année à laquelle l'IA a été développée, cette dernière est en constante mutation.

2.2.Fonctionnement de l'Intelligence Artificielle

Selon Harry Shum, Président Executif de Microsoft, l'IA fonctionne seulement s'il y a présence « d'une vaste quantité de data; d'une puissance informatique extraordinaire, notamment grâce au cloud; et des algorithmes révolutionnaires, basés sur le deep learning ».

L'IA s'applique aujourd'hui dans des domaines variés tels que:

- Les jeux de réflexion
- La recherche mathématique
- La finance
- La médecine
- Les assistants personnels et la domotique
- La reconnaissance faciale et la compréhension des langues
- La robotique

Le monde de demain se voit promettre des perspectives d'avenir positives telles que le développement durable et les progrès de la médecine.

3. Deep Learning

Le deep learning (ou apprentissage profond) est un ensemble de méthodes d'apprentissage automatique conçues sur la base de réseaux de neurones profonds, visant à mimer la « profondeur » des couches d'un cerveau :

le cerveau humain est « profond », dans le sens où chaque action est le résultat d'une longue chaîne de communications synaptiques avec de nombreuses couches de traitement. Le deep learning réunit une classe d'algorithmes d'apprentissage correspondants à ces architectures profondes. Il est souvent utilisé pour un apprentissage « de bout en bout », c'est à dire l'apprentissage simultané des caractéristiques utiles des données, et de la meilleure façon de les utiliser.

Par exemple, pour distinguer une voiture d'une moto, l'algorithme peut apprendre à reconnaître les roues (caractéristique utile), puis à utiliser le nombre de roues pour les distinguer (utilisation de cette caractéristique). Cela fait appel à la fois à des connaissances en neurosciences, en mathématiques et aux progrès technologiques. Le système est constitué d'une série de modules (des

couches de neurones), chacun représentant une étape de traitement. Chaque module est entraînable, comportant des paramètres ajustables (similaires aux poids des classifieurs linéaires). A chaque exemple, tous les paramètres de tous les modules sont ajustés de manière à rapprocher la sortie produite par le système de la sortie désirée. Le qualificatif « profond » vient de l'arrangement de ces modules en couches successives. Les réseaux convolutifs et les réseaux récurrents sont deux des architectures les plus en vogue actuellement pour le deep leaning. Les applications sont multiples.

4. Réseaux de neurones

Les réseaux de neurones (Neural Networks) est l'un des algorithmes d'apprentissage automatique les plus populaires à l'heure actuelle. Au fil du temps, il a été prouvé de manière décisive que les réseaux de neurones surpassent les autres algorithmes en termes de précision et de rapidité. Avec diverses variantes telles que CNN (Réseaux de Neurones Convolutionnels (abrégé en CNN)), RNN (Réseaux de Neurones Récurrents), Auto-Encodeurs, etc…, les réseaux de neurones deviennent peu à peu pour les scientifiques ou les praticiens de l'apprentissage automatique, ce que la régression linéaire était pour les statisticiens.

Les réseaux de neurones profonds (CNN) ont connu un succès considérable dans la reconnaissance et la localisation d'objets dans des images. L'approche fondamentale qui a conduit aux CNN consiste à construire des systèmes artificiels basés sur le cerveau et la vision humaine. Pourtant, sous de nombreux aspects importants, les capacités des CNN sont inférieures à celles de la vision humaine. Un axe de rechercheprometteur consiste à étudier les similitudes et les différences en comblant les lacunes,pour améliorer les CNN. Il existe plusieurs algorithmes de Deep Learning.

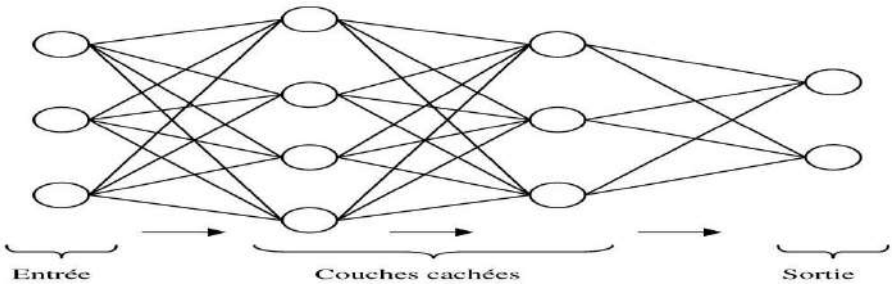


Figure II.1 : Schéma d'un réseau de neurones artificiel

4.1. Modèles de réseaux de neurones

Selon la topologie de connexion des neurones, on peut les classer en deux grandes catégories : réseaux non bouclés (statique ou feedforward) et réseaux bouclés (dynamique, feedback ou

récurrent). (Voir figure.II.1)

Un réseau de neurones non bouclé réalise une fonction (ou plusieurs) algébrique sur ses entrées par composition des fonctions réalisées par chacun de ses neurones. Dans un tel réseau le flux d'information circule des entrées vers les sorties sans retour en arrière. Tout neurone dont la sortie est une sortie du réseau est appelé « neurone de sortie » et les autres qui effectuent des calculs intermédiaires sont des « neurones cachés » [2].

Il existe deux types de réseaux de neurones : les réseaux complètement connectés et les réseaux à couche.

L'architecture la plus générale pour un réseau de neurones est le réseau bouclé (Feedback), dont le graphe des connexions est cyclique : lorsqu'on se déplace dans le réseau en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ (un tel chemin est appelé cycle). La sortie d'un neurone du réseau peut donc être fonction d'elle-même [2].

Les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche.

Les réseaux de neurones les plus utilisés dans la littérature sont représentés sur la figure.II.2.

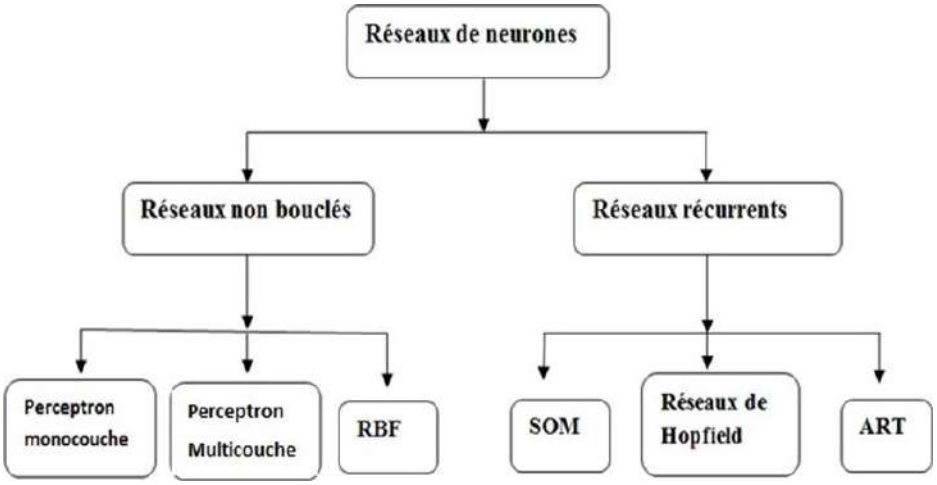


Figure II.2 : différents types de réseaux de neurones

- **Réseau de neurone monocouche (Perceptron)**

Un réseau de neurones est une structure de réseau constituée d'un nombre de nœuds interconnectés par des liaisons directionnelles. Chaque nœud représente une unité de traitement et les liaisons représentent les relations causales entre les nœuds [3].

Le réseau le plus simple est celui monocouche appelé le perceptron et caractérisé de la manière suivante (figure II.3) :

- Il possède R informations en entrée ;
- Il est composé de K neurones, que l'on représente généralement alignés verticalement. Chacun peut en théorie avoir une fonction d'activation différente. En pratique, ce n'est généralement pas le cas .
- Chacun des K neurones est connecté aux R informations d'entrée.

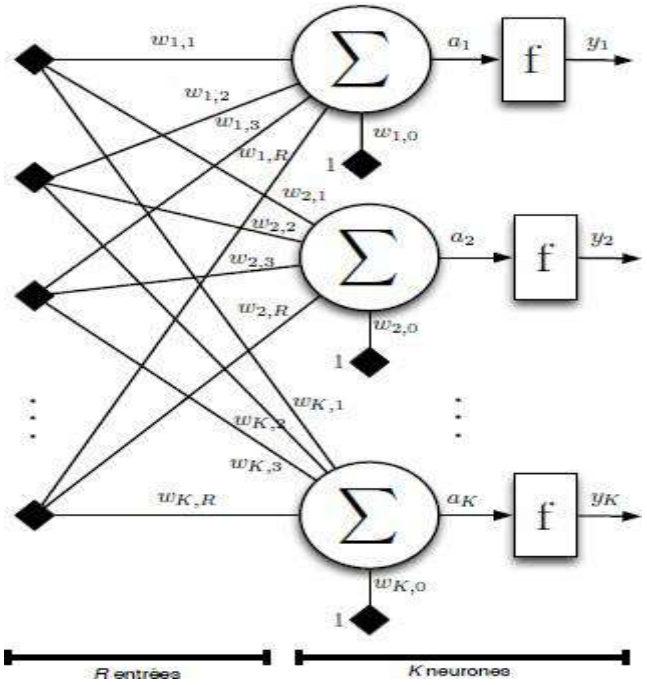


Figure 0.3 : Perceptron monocouche.

Le réseau de neurones possède ainsi R informations en entrée et K sorties, chaque neurone renvoyant sa sortie. Une utilisation courante est que chaque neurone de la couche représente une classe. Pour un exemple X donné, on obtient la classe de cet exemple en prenant la plus grande des K sorties.

- **Perceptron multicouche**

Le perceptron multicouche (Multi Layer Perceptron MLP) est un réseau comportant plusieurs couches, chaque neurone d'une couche étant totalement connecté aux neurones de la couche suivante. Chaque neurone est un automate linéaire généralisé dont la fonction de transfert est supposée sigmoïdale (Figure II.4). Une information circule de la couche d'entrée vers la couche de sortie uniquement ; il s'agit donc d'un réseau de type feedforward. Chaque couche est constituée

d'un nombre variable de neurones, les neurones de la couche de sortie correspondant toujours aux sorties du système [4].

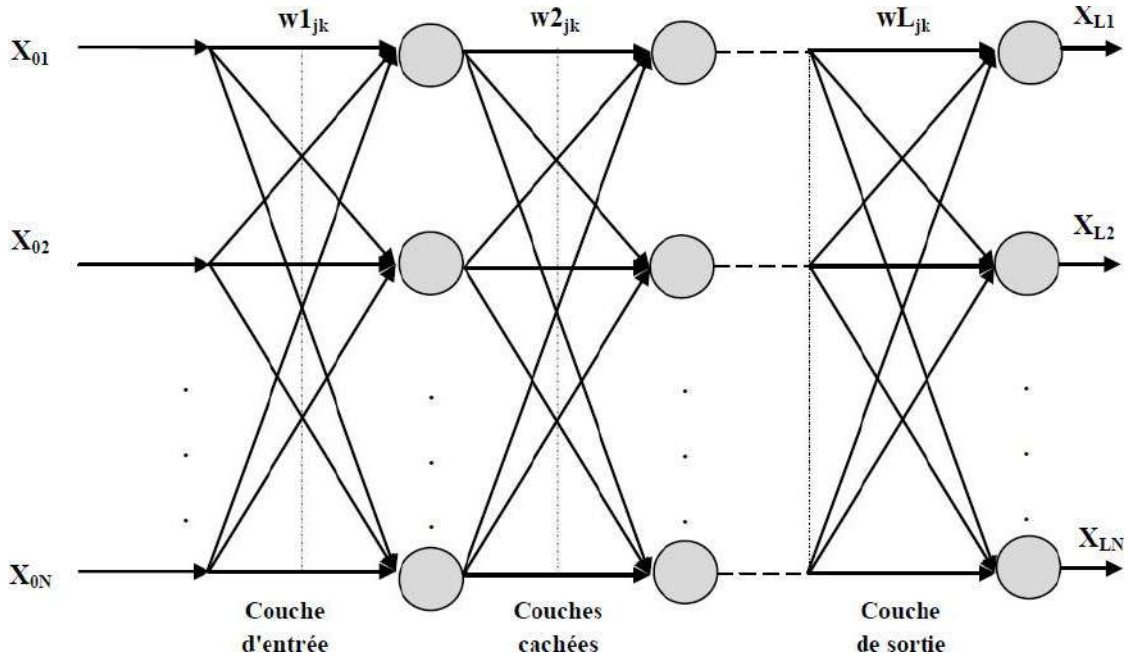


Figure 0.4 : Perceptron multicouche.

- **Les réseaux RBF**

Le réseau RBF est un réseau de neurones supervisé. Il s'agit d'une spécialisation d'un PMC. Un RBF est constitué uniquement de 3 couches (voir figure II.5).

- La couche d'entrée : elle retransmet les entrées sans distorsion.
- La couche RBF : couche cachée qui contient les neurones RBF.
- La couche de sortie : simple couche qui contient une fonction linéaire.

Chaque neurone RBF contient une gaussienne qui est centrée sur un point de l'espace d'entrée. Pour une entrée donnée, la sortie du neurone RBF est la hauteur de la gaussienne en ce point. La fonction gaussienne permet aux neurones de ne répondre qu'à une petite région de l'espace d'entrée, région sur laquelle la gaussienne est centrée. Donc il y a quatre paramètres principaux à régler dans un réseau RBF :

- Le nombre de neurones RBF (nombre de neurones dans l'unique couche cachée).
- La position des centres des gaussiennes de chacun des neurones.
- La largeur de ces gaussiennes.

Le poids des connexions entre les neurones RBF et le(s) neurone(s) de sortie. Toute modification d'un de ces paramètres entraîne directement un changement du comportement du réseau [5].

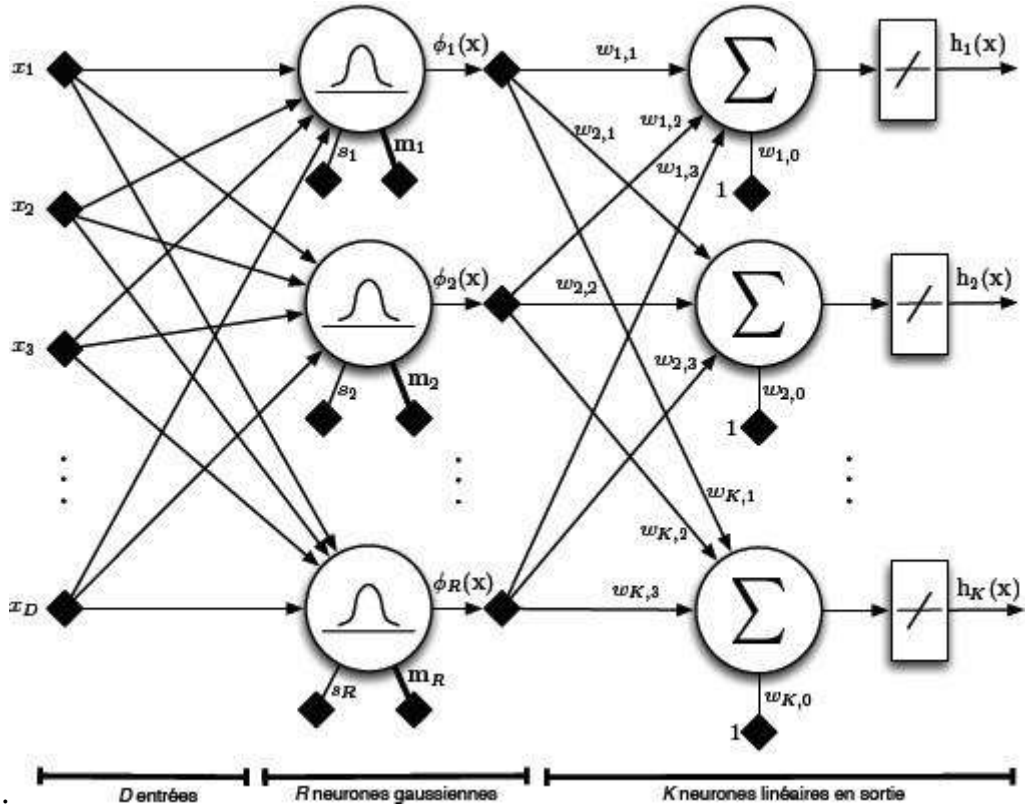


Figure II.5 :Schéma d'un RBF.

- **Réseaux de Hopfield**

Il s'agit d'un réseau récurrent constitué de neurones à deux états (-1 et 1, ou 0 et 1), dont la loi d'apprentissage est la règle de Hebb (1949), qui veut qu'une synapse améliore son activité si et seulement si l'activité de ses deux neurones est corrélée (c'est-à-dire que le poids d'une connexion entre deux neurones augmente quand les deux neurones sont activés au même temps) [6].

- **Réseaux de Kohonen (Self Organized Map)**

Contrairement aux réseaux de Hopfield où les neurones sont modélisés de la façon la plus simple possible, on recherche ici un modèle de neurone plus proche de la réalité. Ces réseaux sont inspirés des observations biologiques du fonctionnement des systèmes nerveux de perception des mammifères. Une loi de Hebb modifiée (tenant compte de l'oubli) est utilisée pour l'apprentissage. La connexion est renforcée dans le cas où les neurones reliés ont une activité simultanée et diminuée dans le cas contraire (alors qu'il nese passait précédemment rien dans ce cas). Tous ces réseaux ont

des applications dans la classification, le traitement d'image, l'aide à la décision et l'optimisation [7].

- **Réseaux Neurones Convolutionnels**

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants pour classer des images. Désignés par l'acronyme CNN, de l'anglais **C**onvolutional **N**eural **N**etwork, ils comportent deux parties bien distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a 2 dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert, Bleu].

La première partie d'un CNN est la partie convolutive à proprement parler. Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de filtres, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image par une opération de maximum local. Au final, les cartes de convolutions sont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN [7] (voir figure II.6)

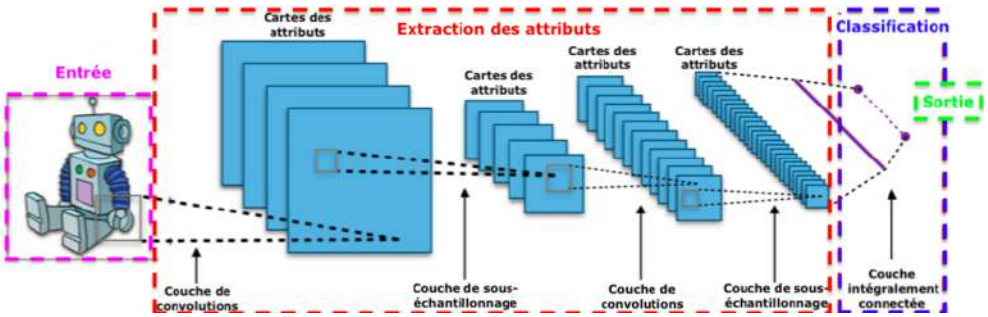


Figure II.6 : Architecture standard d'un réseau de neurone convolutionnel.

Ce code CNN en sortie de la partie convolutive est ensuite branché en entrée d'une deuxième partie, constituée de couches entièrement connectées (perceptron multicouche). Le rôle de cette partie est de combiner les caractéristiques du code CNN pour classer l'image.

La sortie est une dernière couche comportant un neurone par catégorie. Les valeurs numériques obtenues sont généralement normalisées entre 0 et 1, de somme 1, pour produire une distribution de probabilité sur les catégories.

4.2.Apprentissage d'un réseau de neurones

L'apprentissage est la propriété la plus intéressante des réseaux neuronaux. Une caractéristique

des réseaux de neurones est leur capacité à apprendre « Par exemple à reconnaître une lettre, un son... ». Mais cette connaissance n'est pas acquise dès le départ.

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement.

Durant cette phase de fonctionnement, le réseau adapte sa structure « Le plus souvent, les poids des connexions » afin de fournir sur ses neurones de sortie les valeurs désirées. Cet apprentissage nécessite des exemples désignés aussi sous l'appellation d'échantillon d'apprentissage ainsi qu'un algorithme d'apprentissage.

Après initialisation des poids du réseau « En général des valeurs aléatoires », il y a présentation des exemples au réseau et calcul des sorties correspondantes. Une valeur d'erreur ou de correction est calculée et une correction des poids est appliquée.

Le type d'apprentissage est déterminé par la dont les manière changements deparamètres auront lieu. Il existe deux grandes stratégies d'apprentissage.

- **Apprentissage supervisé**

Dans ce cas, la connaissance à priori de la sortie désirée est nécessaire. On présente au réseau le vecteur d'entrée puis sa sortie est calculée et comparée à la sortie désirée, ensuite les poids sont ajustés de façon à réduire l'écart entre elles. Cette procédure est répétée jusqu'à ce qu'un critère de performance soit satisfait. La figure.II.7illustre le processus d'apprentissage supervisé [9].

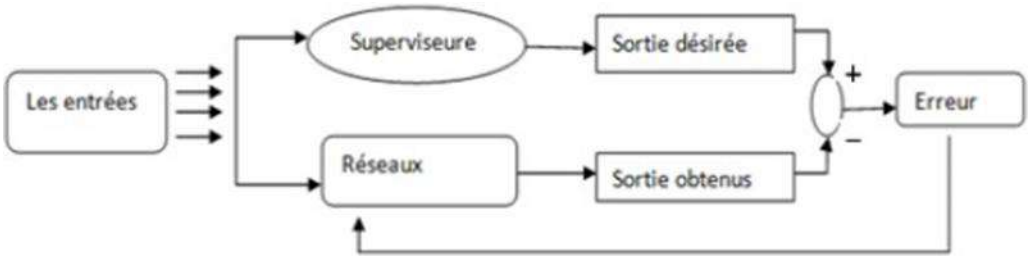


Figure 0.7 : Apprentissage supervisé.

- **Apprentissage non supervisé**

L'apprentissage non supervisé implique la fourniture d'un réseau autonome avec un nombre suffisant d'exemples contenant des répétitions (c'est-à-dire la redondance) de sorte qu'il efface automatiquement les modèles. Ces réseaux sont souvent appelés auto- organisateurs ou apprentissage compétitif.

Dans l'apprentissage non supervisé, les données ne contiennent pas d'informations sur un résultat souhaité, il n'y a pas de superviseur ou d'expert humain. Il s'agit de déterminer les paramètres du réseau neuronal selon un critère défini.

Dans ce cas, les exemples présentés à l'entrée entraînent une auto-adaptation du réseau pour produire des valeurs de sortie proches en réponse à des valeurs d'entrée similaires. Ce type d'apprentissage a souvent moins de complexité dans le calcul par rapport à l'apprentissage supervisé bien sûr, l'architecture réseau, préalablement définie par l'utilisateur, est une forme de non supervision.

5. Réseaux de neurones convolutionnels (CNN)

La vision par ordinateur évolue rapidement de jour en jour. Une des raisons est le développement de l'apprentissage en profondeur. Lorsque nous parlons de vision par ordinateur, un terme réseau de neurones convolutionnel nous vient à l'esprit parce que CNN est fortement utilisé ici. Les exemples de CNN en vision par ordinateur sont la reconnaissance de visage, la classification d'images, etc. Il est similaire au réseau de neurones de base. CNN possède également des paramètres pouvant être appris, tels que le réseau de neurones, à savoir les pondérations, les biais…Un exemple de schéma de principe est illustré par la figure II.8

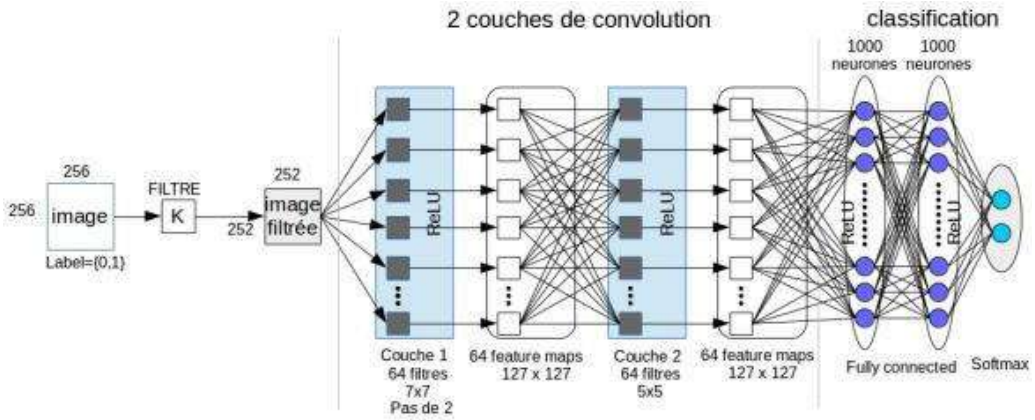


Figure II.8 : Exemple de schéma de principe du CNN[10]

5.1. Les couches de réseaux de neurones convolutionnels

Il y a plusieurs couches différentes dans CNN comme le montre la figure II.9 :

- Couche d'entrée (Input layer).
- Couche de convolution (Convo layer : Convolution + ReLU).
- Couche de Pooling.
- Couche entièrement connectée (Couche Fully connected).

- Couche Softmax/logistique.
- Couche de sortie (Output layer)

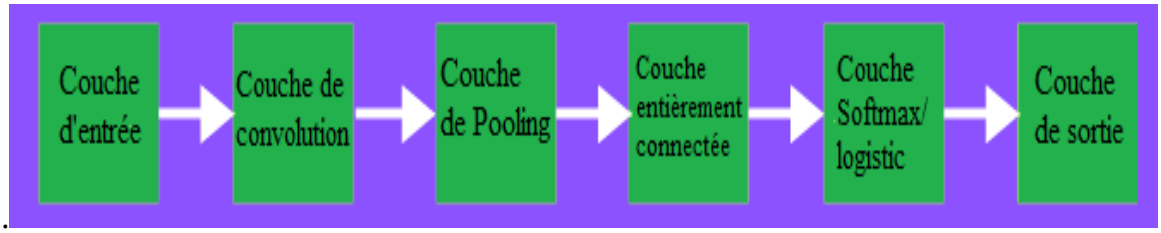


Figure II.9 : Les couches de CNN[11]

Un exemple de l'architecture du CNN est montré par la Figure II.10

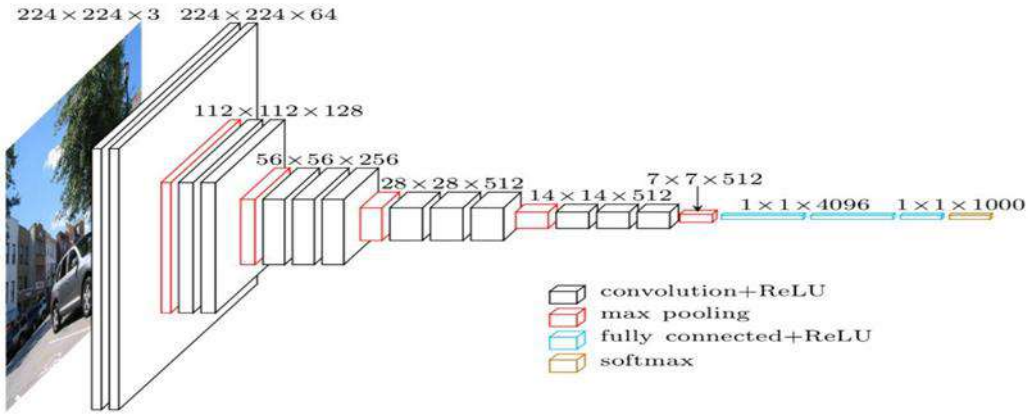


Figure II.10 : Exemple d'architecture CNN[11]

5.1.1. Couche d'entrée du CNN

La couche d'entrée dans CNN doit contenir des données décrivant l'image. Les données d'image sont représentées par une matrice tridimensionnelle qui en général doit être remodelée en une seule colonne (représentation vectorielle).

5.1.2. Couche de convolution

La couche de convolution est parfois appelée couche d'extraction de caractéristiques, car les caractéristiques de l'image sont extraites dans cette couche. Tout d'abord, une partie de l'image est connectée à la couche Convolution pour effectuer une opération de convolution et calculer le produit scalaire entre le champ récepteur (c'est une région locale de l'image d'entrée ayant la même taille que celle du filtre) et le filtre comme le montre la figure II.11. Le résultat de l'opération est un entier unique du volume de sortie. Ensuite, nous faisons glisser le filtre sur le champ récepteur suivant de la même image d'entrée par une foulée et refaisons la même opération. Cette opération est répétée par le même processus encore et encore jusqu'à ce que toute l'image soit parcourue.

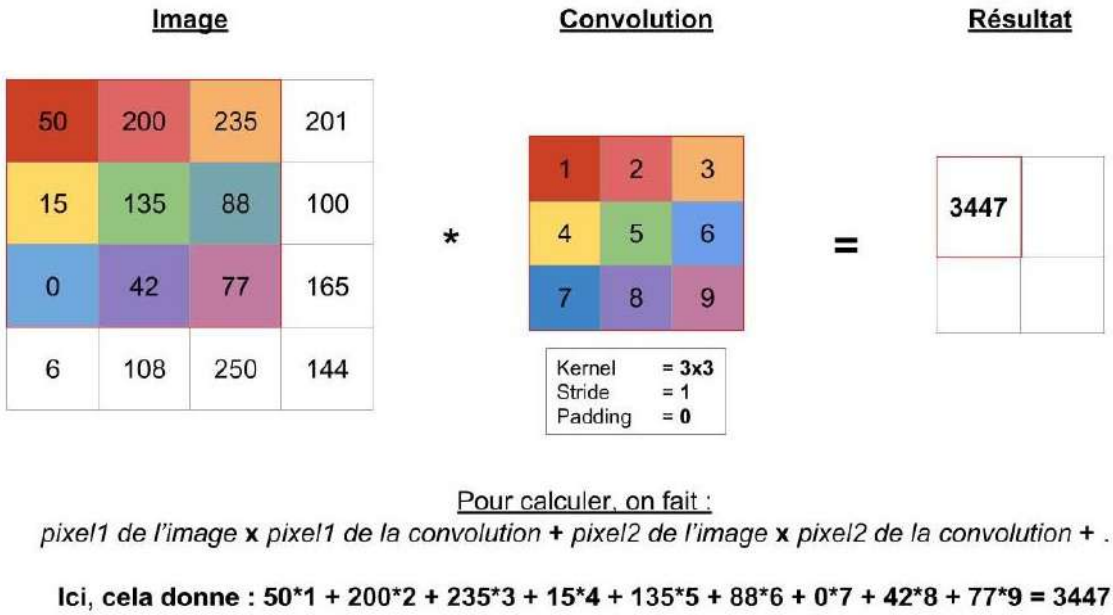


Figure 0.11 : Exemple de principe du filtre convolutionnel[11]

L'activation ReLU voir figure II.12 pour que toutes les valeurs négatives soient mises à zéro.

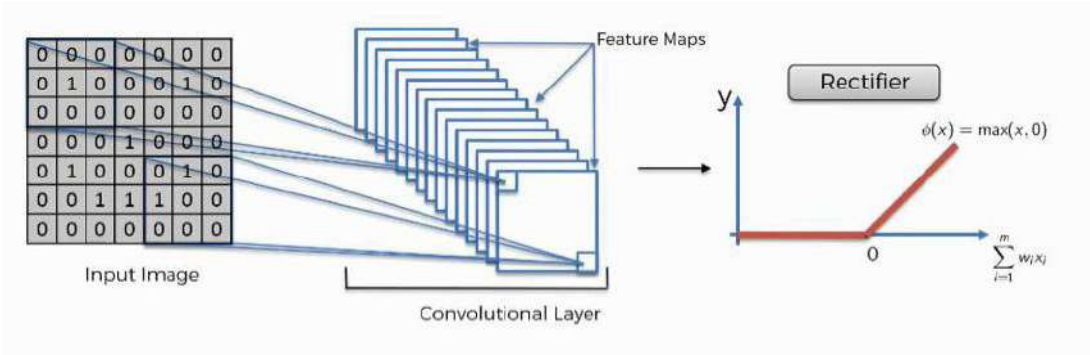


Figure II.12 : Principe de la fonction ReLu[12]

5.1.3. Couche de mise en commun (Pooling)

La couche de Pooling est utilisée pour réduire le volume spatial de l'image d'entrée après la convolution. Elle est utilisée entre deux couches de convolution. Si nous appliquons ***fc*** (Fully Connected) après la couche Convolution sans appliquer le pooling ou le pooling maximum, le calcul sera coûteuse. Ainsi, la mise en commun maximale est le seul moyen de réduire le volume spatial de l'image d'entrée en codant l'information.

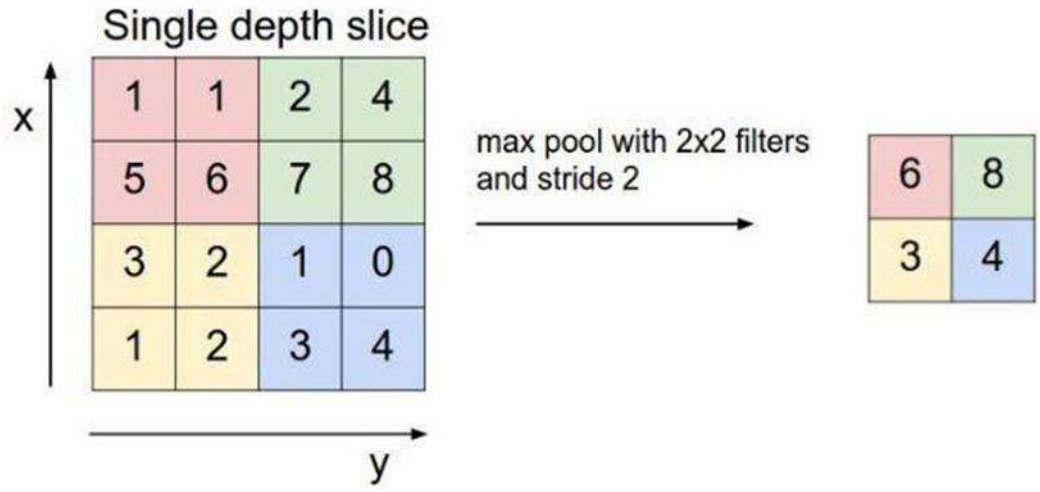


Figure 0.13 : Exemple de principe du Pooling[11]

5.1.4. Couche entièrement connecté (Fully Connected)

Une couche entièrement connectée implique des poids, des biais et des neurones. Il connecte les neurones d'une couche aux neurones d'une autre couche. Il est utilisé pour classer les images entre différentes catégories par formation.

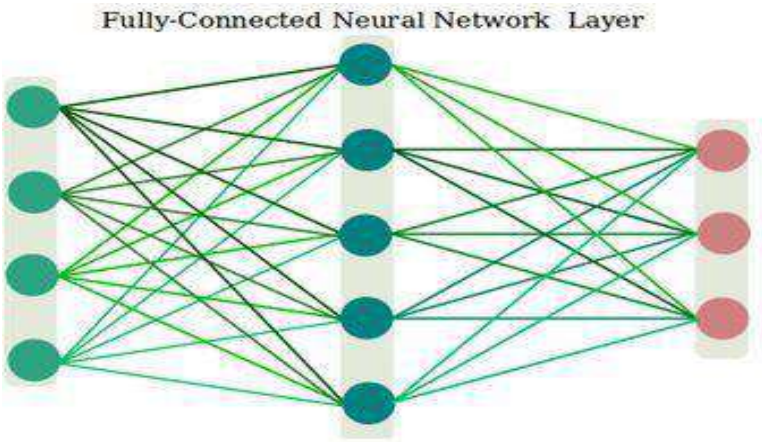


Figure II.14 : Principe de la couche entièrement connectée

5.1.5. Couche Logistique ou Softmax

Softmax ou couche logistique est la dernière couche de CNN. Elle réside à la fin de la couche FC. La logistique est utilisée pour la classification binaire et Softmax est pour la multi-classification comme lemontre la figure II.15.

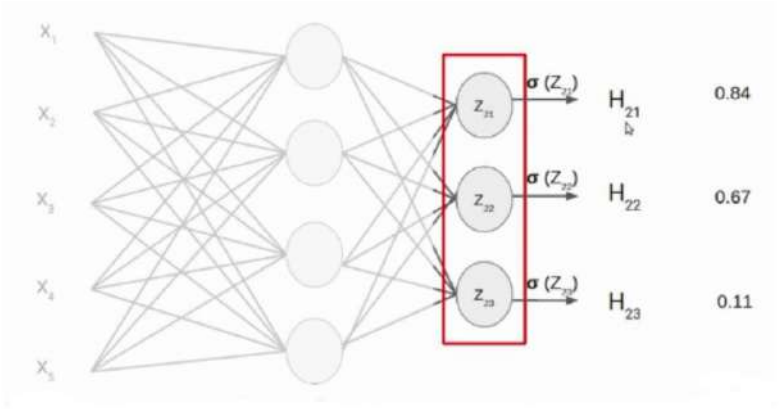


Figure II.15 : Principe de la couche Logistique ou Softmax.

5.1.6. Couche de sortie (output layer)

La couche de sortie contient l'étiquette qui est sous forme codée comme le montre la figure II.16

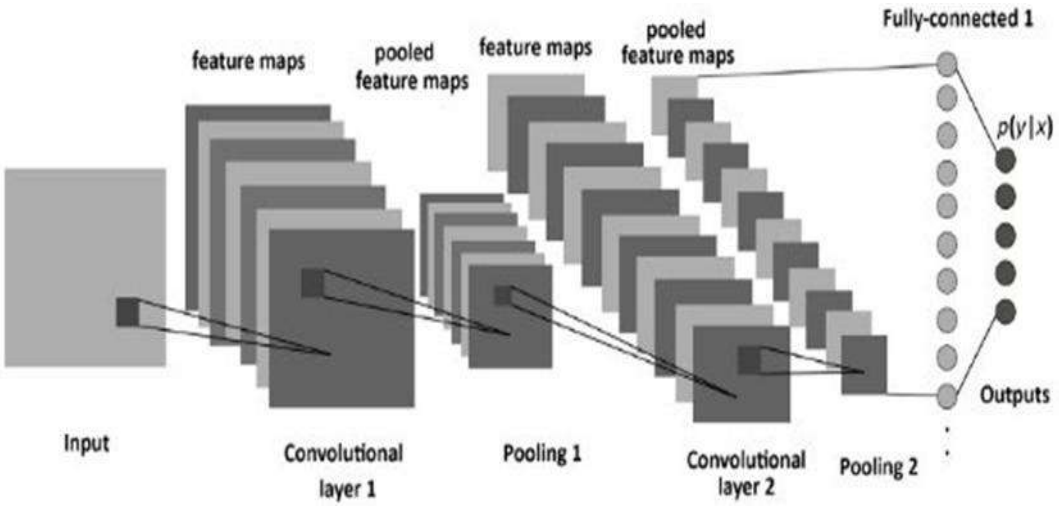


Figure II.16 : Exemple montrant l'étiquette codée de la couche de sortie CNN

5.2. Choix des paramètres

Les CNNs utilisent plus de paramètres qu'un MLP standard. Même si les règles habituelles pour les taux d'apprentissage et des constantes de régularisation s'appliquent toujours, il faut prendre en considération les notions de nombre de filtres, leur forme et la forme du max pooling.

- **Nombre de filtres**

Comme la taille des images intermédiaires diminue avec la profondeur du traitement, les couches proches de l'entrée ont tendance à avoir moins de filtres tandis que les couches plus proches de la

sortie peuvent en avoir davantage. Pour égaliser le calcul à chaque couche, le produit du nombre de caractéristiques et le nombre de pixels traités est généralement choisi pour être à peu près constant à travers les couches

- **Forme du filtre**

Ils sont généralement choisis en fonction de l'ensemble de données. Les meilleurs résultats sur les images de MNIST (28x28) sont habituellement dans la gamme de 5x5 sur la première couche, tandis que les ensemblesde données d'images naturelles (souvent avec des centaines de pixels dans chaque dimension) ont tendance à utiliser de plus grands filtres de première couche de 12x12, voire 15x15. Le défi est donc de trouver le bon niveau de granularité de manière à créer des abstractions à l'échelle appropriée et adaptée à chaque cas.

- **Forme du Max Pooling**

Les valeurs typiques sont 2x2. De très grands volumes d'entrée peuvent justifier un pooling 4x4 dans les premières couches Cependant, le choix de formes plus grandes va considérablement réduire la dimension dusignal, et peut entraîner la perte de trop d'information.

5.2.Avantages de CNN

Un avantage majeur des réseaux convolutifs est l'utilisation d'un poids unique associé aux signaux entrant dans tous les neurones d'un même noyau de convolution. Cette méthode réduit l'empreinte mémoire, améliore les performances et permet une invariance du traitement par translation. C'est le principal avantage du CNN par rapport au MLP, qui lui considère chaque neurone indépendant et donc affecte un poids différent à chaque signal entrant.

6. Conclusion

Ce chapitre a été pour nous l'occasion d'aborder les différents types de réseaux de neurones. Cette étude nous a permis de différencier ces réseaux de par leurs structures et de leurs algorithmes d'apprentissage.

Par ailleurs, nous avons étudié et décrit dans le détail le réseau de neurones CNN. Ce prélude a son implémentation sur notre plate-forme robotique nous a permis de comprendre l'influences des paramètres de ce réseau de neurones sur la qualité de l'apprentissage. Cette maitrise nous sera très utile lors de son utilisation dans la commande de notre robot mobile.

Chapitre III :

Matériels et systèmes embarqués utilisés

1. Introduction

Dans ce chapitre nous allons introduire et présenter les deux cartes électroniques embarquées ainsi que du système de vision que nous utiliserons dans l'élaboration de notre système d'acquisition de données, leur traitement et le calcul des lois de commandes nécessaires à l'asservissement de notre robot mobile. L'architecture matérielle que nous avons développée s'articule essentiellement autour de la carte Arduino de la carte Raspberry Pi et du système de vision Pixy / Caméra HD. La camera embarquée Pixy est considérée comme l'œil humain dans le sens ou grâce à son électronique elle nous fournira directement les caractéristiques visuelles nécessaires au suivi de la trajectoire cible. De plus, nous utiliserons une deuxième caméra HD pour la lecture des plaques de signalisation. Le Raspberry Pi sera quant à lui assimilé au cerveau de notre robot tandis que le Arduino aura pour rôle la gestion de l'ensemble Capteurs / Actionneurs du robot.

Enfin, nous décrirons le fonctionnement détaillé d'une nouvelle carte électronique « Neural Compute Stick 2 d'Intel ». Celle-ci aura pour rôle de décharger le Raspberry Pi d'un certain nombre de tâches.

2. Neural Compute Stick 2 d'Intel

La figure III.1 représente NTSC 2 qui est un kit de développement "Plug and Play" pourl'inférence AI.



Figure III.1 : Neural Compute Stick 2[13].

Il est maintenant plus facile que jamais de mettre la vision par ordinateur et l'intelligenceartificielle au service d'Internet des Objets (Internet of Things : IoT) et des prototypes de périphériques grâce à ses capacités améliorées. Elle offre des prototypes plus intelligents pour le développement des appareils intelligents telles que :

- Une caméra intelligente.
- Un drone doté de capacités de reconnaissance de gestes .
- Un robot industriel.
- Un dispositif intelligent à domicile.

Ce qui ressemble à une clé USB standard cache beaucoup plus à l'intérieur. Elle est basée sur le tout dernier VPU Intel® Movidius™ Myriad™ X qui comprend le moteur de calcul neuronal, un accélérateur matériel dédié pour les inférences de réseaux neuronaux profonds. Avec plus de noyaux calculs que la version originale (NCS) et l'accès à la Distribution d'OpenVINO™ toolkit d'Intel®. NCS2 offre des performances 8 fois supérieures à celles de la génération précédente. Elle regroupe plusieurs caractéristiques [14].

2.1.Distribution Intel® d'OpenVINO™ toolkit

Une demande fréquente pour le NCS 2 d'Intel® est qu'il peut se connecter à n'importe quel ordinateur mono-carte (Single Board Computer : SBC) disponibles sur le marché. La distribution Intel® d'OpenVINO™ toolkit fournit une installation binaire qui prend en charge un grand nombre de développement et de plates-formes en fournissant des outils et une API pour l'Intel® NCS 2. Mais de nombreux utilisateurs qui réalisent des prototypes avec le NCS 2 d'Intel® veulent choisir leur plateforme SBC préférée afin de se rapprocher le plus possible de leur configuration finale du système. Pour répondre aux besoins de ces développeurs, Intel a ouvert l'API du moteur d'inférence OpenVINO™ toolkit et le plug-in VPU Intel® Movidius™Myriad™[13].

2.2.Inférence du Deep Learning

OpenVINO™ toolkit permet aux développeurs de déployer des modèles d'apprentissage profond performants grâce à une API de haut niveau du moteur d'inférence C++ ou Python* intégrée à la logique applicative. L'inférence du DL prend en charge plusieurs plateformes Intel® et est inclus dans la distribution Intel® d'OpenVINO™. Les opérations dans OpenVINO™ toolkit se déroulent comme le montre la figure III.2

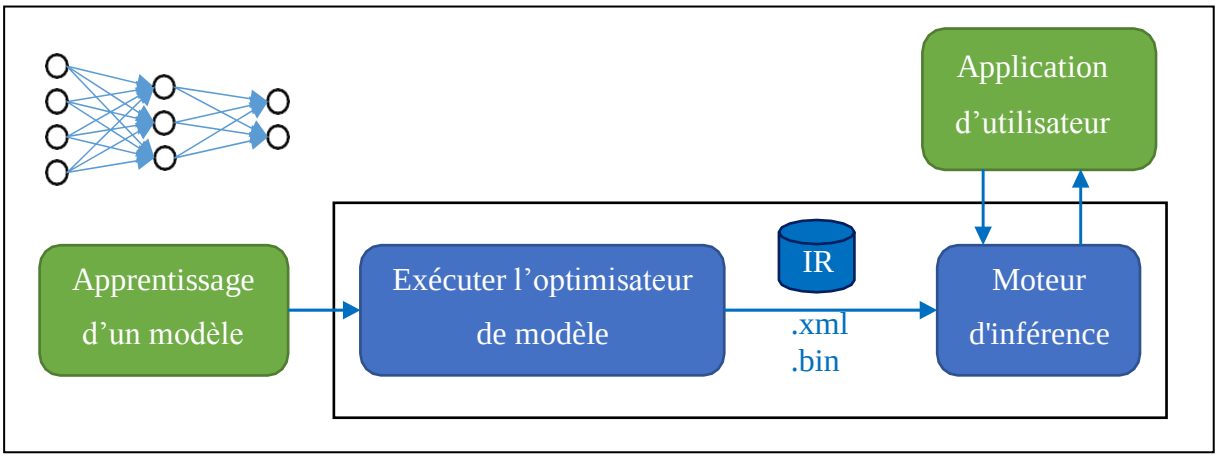


Figure III.2 : Diagramme de déroulement des opérations d'OpenVINO toolki[15].

La distribution Intel® d'OpenVINO™ comprend les deux éléments suivants :

a) L'optimisateur de modèle

C'est un outil de ligne de commande basé sur Python. Il importe des modèles formés à partir de frameworks d'apprentissage profond populaires tels que Caffe*, TensorFlow*, Apache MXNet*, et Open Neural Network Exchange (ONNX*).

Les tâches de l'optimisateur de modèle sont:

- Fonctionne sur plusieurs systèmes d'exploitation : Windows*, Linux*, et macOS* ;
- Effectue des analyses et des ajustements pour une exécution optimale sur les dispositifscibles en utilisant des modèles statiques et entraînés ;
- Sérialise et ajuster le modèle dans un format de représentation intermédiaire (IR)d'Intel
- Prend en charge de plus de 100 modèles publics pour Caffe, TensorFlow, MXNet et ONNX.

Les frameworks standards ne sont pas nécessaires pour générer des fichiers IR pour les modèles constitués de couches standards. Lors du traitement de couches personnalisées dans des modèles originaux, l'Optimisateur de modèle fournit un mécanisme flexible d'extensions [15].

b) Le moteur d'inférence

Ce moteur utilise une API commune pour fournir des solutions d'inférence sur la plateforme de votre choix : CPU, GPU, VPU ou FPGA. Les tâches du moteur d'inférence sont :

- Exécuter différentes couches sur différentes cibles (par exemple, un GPU et des couches sélectionnées sur un CPU) ;

- Implémenter des couches personnalisées sur un CPU tout en exécutant la topologie restante sur un GPU - sans avoir à réécrire les couches personnalisées ;
- Optimiser les charges de travail (analyse des graphiques de calcul, ordonnancement et compression des modèles) pour le matériel cible grâce à une solution de notation intégrée et conviviale ;
- Exploiter la nouvelle exécution asynchrone pour améliorer les performances de la fréquence d'images tout en limitant les cycles gaspillés ;
- Utiliser une API pratique en C, C++ ou Python pour travailler sur les fichiers IR et optimiser l'inférence [15].

3. Raspberry Pi 3 model B+

Le Raspberry Pi est un micro-ordinateur possédant un processeur embarqué ARM, Les avantages sont un prix attractif (entre 25 et 45 €), et une taille minimaliste

Nous présentons dans la suite, le Raspberry pi 3 Model B que l'on a utilisé dans notre projet.

Ce dernier est considéré comme le cerveau de notre système parce qu'il va gérer les informations issues de la caméra, les traiter et calculer les lois de commandes nécessaires aux mouvements du robot mobile.

Le Raspberry Pi 3 Model B est un petit ordinateur (Mini-PC) qui fonctionne sous le système d'exploitation Linux sur carte SD destiné à des applications d'informatique embarquée.

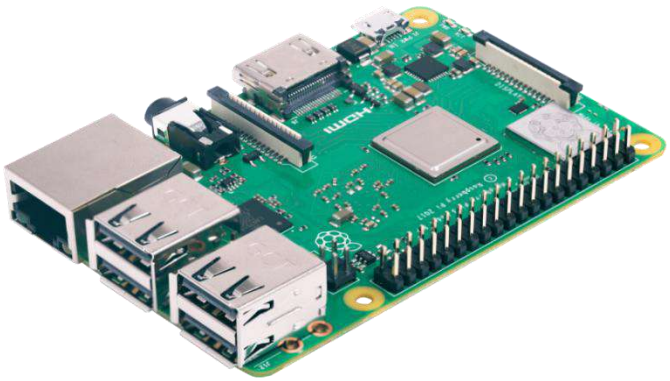


Figure III.3 : Raspberry pi 3 B+

3.1.1'architecture

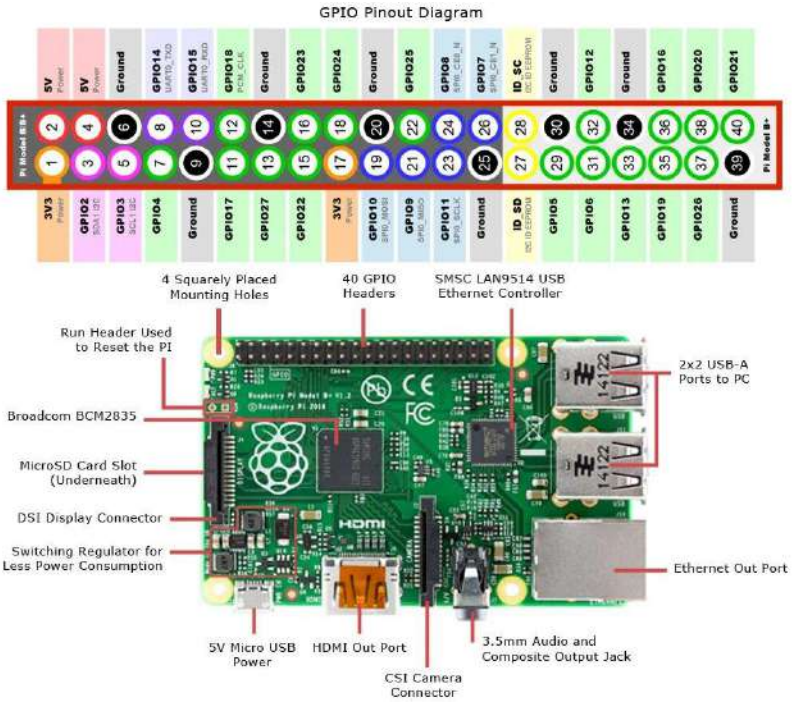


Figure III.4 : distribution des broches entrée sortie

3.2.Programmation

Il existe plusieurs langages de programmation qui peut être compilé sur le Raspberry Pi, telle que le langage C, C++, HTML5, le Java, le JavaScript...etc. Le langage utilisant dans notre projet et le Python.

Le Python est un langage de programmation inventé par Guido van Rossum, la première version de python est sortie en 1991.

C'est un langage de programmation interprété, c'est à dire qu'il n'est pas nécessaire de le compiler avant de l'exécuter.

Étant un langage très riche, surtout si l'on prend en compte les nombreuses Bibliothèques complémentaires disponibles, Ce dernier est à la fois simple et puissant, c'estun langage facile à apprendre et son code est plus lisible, il est donc plus facile à maintenir. Il est parfois jusqu'à 5 fois plus concis que le langage Java par exemple, ce qui augmente la productivité du développeur et réduit mécaniquement le nombre de bugs [16].

4. La carte Arduino Mega

Arduino Mega est une plate-forme électronique open-source basée sur un matériel et un logiciel facile à utiliser. Arduino Mega est né à l'Ivrea Interaction Design Institute comme un outil facile pour le prototypage rapide, destiné aux étudiants sans formation en électronique et en

programmation. Au fil des années, Arduino Mega a été le cerveau de milliers de projets, des objets de tous les jours aux instruments scientifiques complexes.

Une communauté mondiale de décideurs - étudiants, amateurs, artistes, programmeurs et professionnels - s'est réunie autour de cette plate-forme open source, leurs contributions ont contribué à une connaissance incroyable de connaissances accessibles qui peuvent être d'une grande aide pour les novices et les experts.



Figure III.5 : la carte Arduino Mega

4.1.Présentation

Il s'agit d'une carte électronique basée autour d'un microcontrôleur Atmega du fabricant Atmel, dont le prix est relativement bas pour l'étendue possible des applications. Voilà à quoi ressemble la carte que nous allons utiliser dans la figure III.6 :

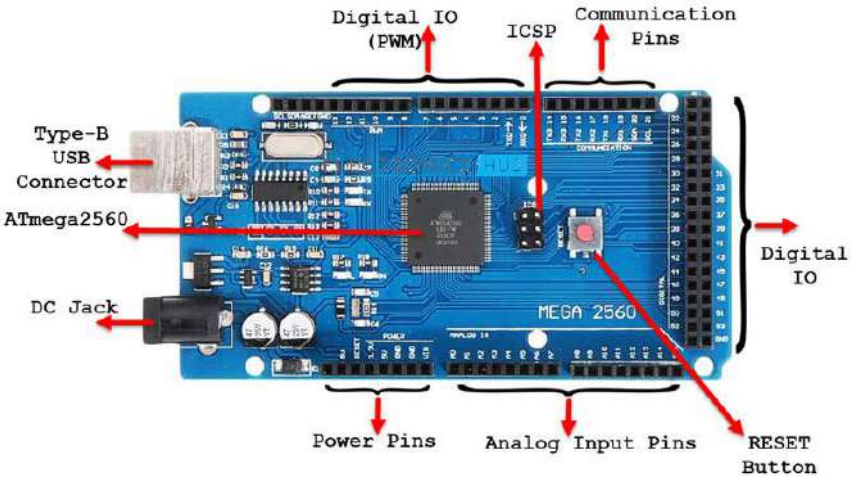


Figure III.6 : Description de la carte arduino Mega

4.2.Spécifications techniques

- Microcontrôleur : ATmega2560
- Tension de fonctionnement : 5 V
- Gamme de tension d’entrée (recommandée) : 7-12 V
- Gamme de tensions d’entrée (limite) : 6-20 V

- Pins digitaux I/O : 54
- Pins digitaux I/O PWM : 14
- Pins d'entrée analogique : 16
- Courant direct par pin I/O : 40 mA
- Courant direct pour les pins 3,3 V : 50 mA
- Mémoire flash : 256 KB
- Mémoire Flash du Bootloader : 8 KB
- SRAM : 8 KB
- EEPROM : 4 KB
- Fréquence d'horloge de l'oscillateur à quartz : 16 MHz
- Dimensions : 101,52 x 53,3 mm
- Poids : 37 g

4.3. L'extension (Shield) de commande des moteurs

Cette extension (Adafruit Motor Shield) est basée sur le circuit CI L293D, qui est un double pilote à pont complet conçu pour gérer des charges inductives telles que les relais, les solénoïdes, les moteurs DC et les servomoteurs. Il vous permet de commander 4 moteurs DC avec votre carte Arduino, en contrôlant la vitesse et la direction de chacun d'eux indépendamment Adafruit Motor Shield ne doit être alimenté que par une alimentation externe, Parce que le L293D monté sur le blindage comporte deux connexions d'alimentation séparées, l'une pour la logique et l'autre pour l'alimentation des charges inductives. Lecourant requis du moteur dépasse souvent la note maximale du courant USB.

L'alimentation externe (non-USB) peut provenir d'un adaptateur AC-DC ou d'une batterie.

L'adaptateur peut être branché en branchant une prise centrale positive de 2,1 mm dans la prise d'alimentation de la carte Arduino sur laquelle le blindage du moteur est monté ou en connectant les fils qui alimentent l'alimentation des bornes à vis (la pin Vin et GND). [17]

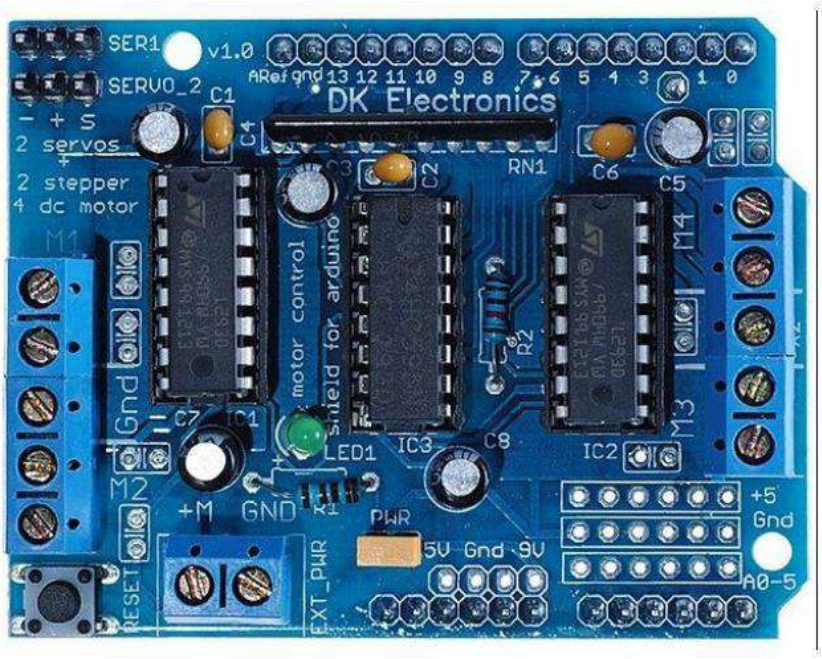


Figure III.7: Motor Shield.

Principales caractéristiques :

- Tension d'alimentation du circuit logique : 5V
- Tension d'alimentation du circuit puissance : 0V→ 8V
- Tension minimale de sensibilité MLI : 2.3V
- Courant maximal du circuit puissance : 3A

Les moteurs et le registre à décalage :

Le blindage du moteur peut contrôler jusqu'à quatre moteurs à courant continu à l'aide de deux circuits intégrés de moteur à moteur à double pont L293D. Vous pouvez lire la fiche technique pour plus de détails, Comme le montre la Figure III.12, chaque CI possède quatre entrées numériques qui commandent bi directionnellement deux moteurs à courant continu. Les entrées sont par paires, et chaque paire contrôle un moteur. Par exemple, si la broche 2 est réglée en haut et que la broche 7 est bas, le moteur 1 est allumé, Si les entrées sont inversées, le moteur est à nouveau activé, mais son sens de rotation est inversé, et si les deux sont bas, le moteur est éteint. Il existe d'autres configurations possibles, mais celle- ci est supportée par le bouclier du moteur. [17]

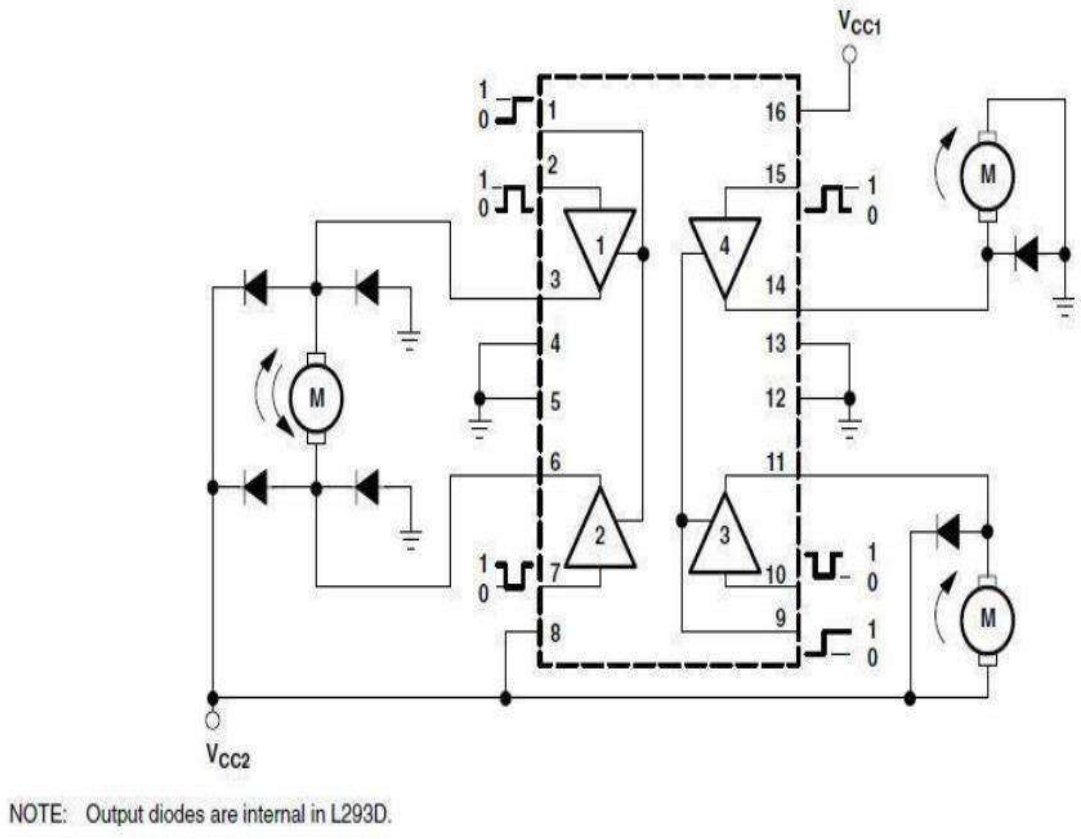


Figure III.8 : fonctionnement de L293D.

5. La caméra Pixy

5.1.Aperçu

Un Système de vision petit, rapide, facile à utiliser, peu coûteux et facilementaccessible. Il apprends à détecter des objets que vous l'enseignez et il indique ce qu'il détecte 50 fois par seconde.

La caméra Pixy se connecte à Arduino avec un câble inclus et fonctionne également avec Raspberry Pi, BeagleBoneet des contrôleurs similaires.

Toutes les bibliothèques pour Arduino, Raspberry Pi, etc. sont fournies et les langage de programmation C / C ++ et Python sont pris en charge.

Cette caméra communique via l'une des nombreuses interfaces : SPI, I2C, UART, USB ou sortie analogique / numérique.

L'utilitaire de configuration s'exécute sous Windows, MacOs et Linux.

Tout logiciel / firmware est open-source GNU-licence. [18]

Toute la documentation sur le matériel, y compris les schémas, la liste dematériaux, la mise en page des PCB, etc.

5.2.Simplicité de mise en œuvre du système de vision

Si vous souhaitez que votre robot effectue une tâche, comme la collecte d'un objet,la poursuite d'une balle, la localisation d'une station de chargement etc, et que vous souhaitez qu'un seul capteur aide à accomplir toutes ces tâches, la vision est votre capteur. Les capteurs Vision (image) sont utiles car ils sont si flexibles Avec le bon algorithme, un capteur d'image peut détecter ou détecter pratiquement n'importe quoi. Mais il existe deux inconvénients avec les capteurs d'image :

Ils sortent beaucoup de données, des dizaines de mégaoctets par seconde. Le traitement de cette quantité de données peut submerger de nombreux processeurs, Et si le processeur peut suivre les données, une grande partie de sa puissance de Traitement ne sera pas disponible pour d'autres tâches.

Pixy résout ces problèmes en associant un puissant processeur dédié au capteur d'image.

Elle traite les images du capteur d'image et envoie uniquement les informations utiles (par exemple, le dinosaure violet détecté à x = 54, y = 103) à votre microcontrôleur, Et cela se fait à cadence (50 Hz). L'information est disponible via l'une des nombreuses interfaces : UART série, SPI, I2C, USB ou sortie numérique / analogique. Ainsi, votre Arduino ou un autre microcontrôleur peut facilement parler avec Pixy et dispose de nombreuses CPU disponibles Pour d'autres tâches.

Il est possible de brancher plusieurs Pixys sur votre microcontrôleur - par exemple, un robot avec 4 Pixys et 360 degrés de détection, Ou utiliser Pixy sans microcontrôleur et utiliser les sorties numériques ou analogiques pour déclencher des événements, des commutateurs, des servons, etc .[18]

5.3.Support du contrôleur

La caméra Pixy peut facilement se connecter à de nombreux contrôleurs différents car il prend en charge plusieurs options d'interface (UART série, SPI, I2C, USB ou sortie numérique / analogique), mais Pixy a commencé sa vie à parler à Arduino. Au cours des derniers mois, nous avons ajouté un support pour Arduino Due, Raspberry Pi et BeagleBone Black. Des bibliothèques de logiciels sont fournies pour toutes ces plates-formes afin que vous puissiez vous lancer rapidement. En outre, nous avons ajouté une API Python si vous utilisez un contrôleur basé sur Linux (par exemple, Raspberry Pi, BeagleBone) [18].

5.4.La détection d'objets

La caméra Pixy utilise un algorithme de filtrage basé sur les couleurs pour détecter les objets, Les méthodes de filtrage basées sur les couleurs sont populaires car elles sont rapides, efficaces et

relativement robustes. La plupart d'entre nous connaissent le RGB (rouge, vert et bleu) pour représenter les couleurs. Pixy calcule la couleur (teinte) et la saturation de chaque pixel RGB du capteur d'image et les utilise comme paramètres de filtrage primaires. La teinte d'un objet reste en grande partie inchangée avec les changements d'éclairage et d'exposition. Les changements d'éclairage et d'exposition peuvent avoir un effet frustrant sur les algorithmes de filtrage des couleurs, ce qui leur permet de se casser. L'algorithme de filtrage de Pixy est robuste en matière d'éclairage et de changements d'exposition.

La caméra peut trouver littéralement des centaines d'objets à la fois. Il utilise un algorithme de composants connectés pour déterminer où un objet commence et un autre se termine. Pixy compile les tailles et les emplacements de chaque objet et les signale à travers l'une de ses interfaces (par exemple SPI).

Dans la documentation technique de ce système de vision, nous retrouvons souvent le terme "50 images par seconde" ou "50 frames par seconde". En bref, cela signifie que Pixy est rapide. Pixy traite toute une image 640x400 toutes les 1 / 50ème de seconde (20 millisecondes). Cela signifie que vous obtenez une mise à jour complète de toutes les positions des objets détectés toutes les 20 ms. À ce rythme, il est possible de suivre le parcours de la balle tombante / rebondissant. (Une balle qui se déplace à 30 mph se déplacemoins d'un pied en 20 ms.) .[18]

5.5.L’application PixyMon

PixyMon est une application qui fonctionne sous Windows, MacOs et Linux. Il vous permet de voir ce que Pixy voit, en tant que vidéo brute ou transformée. Il vous permet également de

configurer votre Pixy, de définir le port de sortie et de gérer les signatures des couleurs. PixyMon communique avec Pixy sur un câble mini USB standard.

PixyMon est idéal pour le débogage de votre application, Vous pouvez brancher un câble USB à l'arrière de Pixy et exécuter PixyMon, puis voir ce que Pixy voit lorsqu'il est accroché à votre Arduino ou à un autre microcontrôleur - pas besoin de débrancher quoi que ce soit. PixyMon est open source, comme tout le reste.[18]

5.6.Spécifications techniques

La puissance du système de vision Pixy est l’intégration sur une seule carte à la fois du système optique de vision est des composants électroniques (processeur, RAM, …) nécessaires à l’acquisition et aux traitements des images.[18]

- Processeur*:* NXP LPC4330, 204 MHz, dual core
- Capteur d'image: Omnivision OV9715, 1/4 ", 1280x800

- Lens field-of-view: 75 degrés horizontal, 47 degrés vertical
- Type d'objectif: standard M12 (plusieurs types différents disponibles)
- Consommation électrique: 140 mA typique
- Entrée d'alimentation: entrée USB (5V) ou entrée non régulée (6V à10V)
- RAM: 264K octets
- Flash: 1M octets
- Sorties de données disponibles: UART série, SPI, I2C, USB, numérique,analogique
- Dimensions: 2.1 "x 2.0" x 1.4
- Poids: 27 grammes

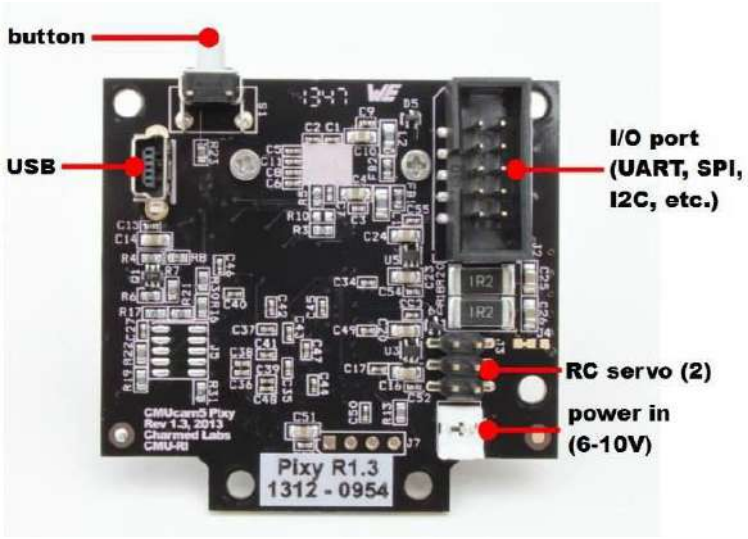


Figure III.9 : Structure de la Cam Pixy CMUcam5

6. La Webcam HD C270

D'un point de vue fonctionnel, cette Webcam HD C270 reste plutôt classique puisqu'elle se relie à l'ordinateur hôte via un port USB 2.0. Ordinateur sur lequel la Webcam s'accroche par une pince servant aussi de socle/pied.

Ce qui est moins classique, c'est la possibilité de la Webcam d'enregistrer des vidéos en HD 720p (1280 x 720 pixels).



Figure III.10 : Webcam HD C270

7. Environnements et bibliothèques

7.1.Anaconda (Python Distribution)

Figure III.11 : logo de la bibliothèque Anaconda.

Anaconda est une distribution libre et open source des langages de programmation Python et R. Elle est appliquée au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions de paquetages sont gérées par le système de gestion de paquets Conda. La distribution Anaconda comprend plus de 250 paquets populaires en science des données adaptés pour Windows, Linux et MacOS.

Ces paquets sont automatiquement installés lors de l'installation d'Anaconda, et de nombreux autres peuvent être installés en tapant simplement « conda install [nom du logiciel] » dans l'invite de commande d'Anaconda.

Conda forge : est un effort communautaire qui fournit des paquets Conda pour une large gamme de logiciels. Ces paquets peuvent être trouvés en ligne.

7.2.SPYDER

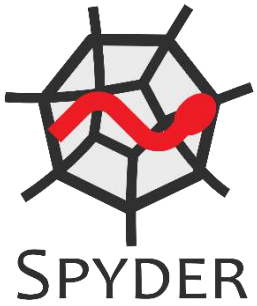


Figure III.12 : logo de la bibliothèque Spyder.

SPYDER (The Scientific Python Development Environment) est un environnement scientifique puissant écrit en Python, pour Python, et conçu par et pour les scientifiques, les ingénieurs et les analystes de données. Il offre une combinaison unique des fonctionnalités avancées d'édition, d'analyse, de débogage et de profilage d'un outil de développement complet avec les capacités d'exploration des données, d'exécution interactive, d'inspection approfondie et de visualisation d'un progiciel scientifique. De plus, Spyder offre une intégration intégrée avec de nombreux progiciels scientifiques populaires, dont NumPy, SciPy, Pandas, IPython, QtConsole, Matplotlib, SymPy, et bien d'autres encore. Au-delà de ses nombreuses fonctionnalités intégrées, ses capacités peuvent être étendues encore plus loin grâce à son système de plugins et à son API.

7.3.TensorFlow



Figure III.13 : logo de la bibliothèque TensorFlow.

TensorFlow (TF) est une plate-forme multiplateforme à code source libre pour le ML. Elle contient des bibliothèques et des ressources communautaires qui permettent d'expérimenter rapidement de nouvelles idées, de transformer des concepts en code de modèles de référence et en publication. TF permet aux développeurs de créer et de déployer facilement des applications de ML. À partir de la version 2.0, TensorFlow est devenu capable de construire facilement des modèles d'apprentissage automatique à plusieurs niveaux d'abstraction avec une grande vitesse et de bonnes performances. Pour commencer, TensorFlow fournit les API Keras séquentielles de haut niveau qui présentent les avantages suivants : convivialité, modularité, compossibilité et extensibilité.

7.4.NumPy

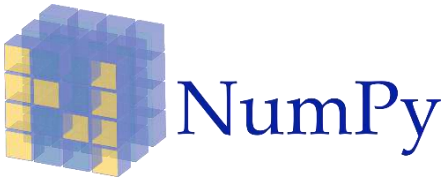


Figure III.14 : logo de la bibliothèque NumPy.

NumPy est la bibliothèque mathématique de calcul scientifique pour le langage de programmation Python. Elle prend en charge les objets de tableaux multidimensionnels (matrices) avec une large collection de fonctions d'opérations sophistiquées (diffusion). NumPypeut être utilisé de manière pratique comme un gestionnaire multidimensionnel efficace de données génériques et définir de nouveaux types de données arbitraires. Cela permet une intégration rapide et transparente avec une grande variété de frameworks et de bases de donnéespopulaires.

7.5.OpenCV



Figure III.15 : logo de la bibliothèque OpenCV.

OpenCV (Open Source Computer Vision Library) est une bibliothèque de logiciels de vision par ordinateur et de ML à code source ouvert. OpenCV a été construite pour fournir une infrastructure commune pour les applications de vision par ordinateur et pour accélérer l'utilisation de la perception des machines dans les produits commerciaux. Étant un produit sous licence BSD[7], OpenCV permet aux entreprises d'utiliser et de modifier facilement le code. La bibliothèque contient plus de 2500 algorithmes optimisés parmi lesquels se trouve un ensemble complet d'algorithmes classiques de vision par ordinateur et de ML. Ces algorithmes peuvent être utilisés dans plusieurs applications.

7.6.Keras



Figure III.16 : logo de la bibliothèque Keras.

Keras est une API de réseaux de neurones de haut niveau, écrite en Python et interfaçable avec TensorFlow, CNTK et Theano. Elle a été développée avec pour objectif de permettre des expérimentations rapides. Être capable d'aller de l'idée au résultat avec le plus faible délai possible étant la clef d'une recherche efficace.

Nous vous recommandons Keras si vous êtes à la recherche d'une librairie qui :

- Permet le prototypage rapide et facile (de par sa convivialité, sa modularité et son extensibilité).
- Supporte à la fois les réseaux convolutifs et les réseaux récurrents ainsi que la combinaison des deux.
- Fonctionne de façon transparente sur CPU et GPU.

8. L'application Cascade

Cascade Trainer GUI est un programme qui peut être utilisé pour former, tester et améliorer les modèles de classificateurs en cascade. Il utilise une interface graphique pour définir les paramètres et faciliter l'utilisation des outils OpenCV pour la formation et le test des classificateurs.

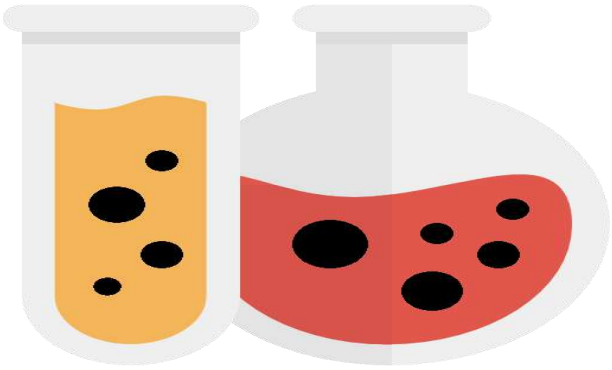


Figure III.17 : photo de logiciel Cascade

9. Conclusion

Ce chapitre a été pour nous l'occasion de présenter les cartes Raspberry PI 3 B, Arduino et le Neural Computer Stick 2 d'Intel ainsi que le système de vision Pixy / Caméra HD que nous avons utilisé pour concevoir l'architecture matérielle nécessaire à la commande du robot mobile.

L'asservissement de ce robot basé sur l'utilisation des données visuelles, nécessite d'une part une puissance de calcul importante et d'autre part des calculateurs peu volumineux et léger pouvant être embarqués sur la structure mécanique du robot mobile.

La solution que nous avons adoptée et qui sera présentée dans le dernier chapitre a considérablement amélioré les performances et le traitement d'image et d'asservissement visuel. De

plus, cette configuration nous permettra d'implémenter notre algorithme neuronal pour la détection et le classement des plaques de signalisation. Ce dernier basé sur le réseau de neurones CNN nécessite l'utilisation de ressources informatiques importantes (mémoire, puissance de calcul, ...) que notre architecture devra être capable de fournir.

Chapitre IV :
Guidage intelligent du robot

1. Introduction

Pour mettre en œuvre une commande neuronale de notre robot mobile, nous avons besoin d'une stratégie de commande. Pour cela, nous allons utiliser un asservissement visuel afin de contrôler la position d'un robot mobile et détecter les plaques de signalisation qui balises le parcours de ce robot. Cet objectif doit être atteint en utilisant uniquement les données visuelles issues des caméras. De plus, la contrainte d'autonomie du robot mobile doit être satisfaite, c'est pour cela que nous utiliserons des algorithmes intelligents.

2. Description générale

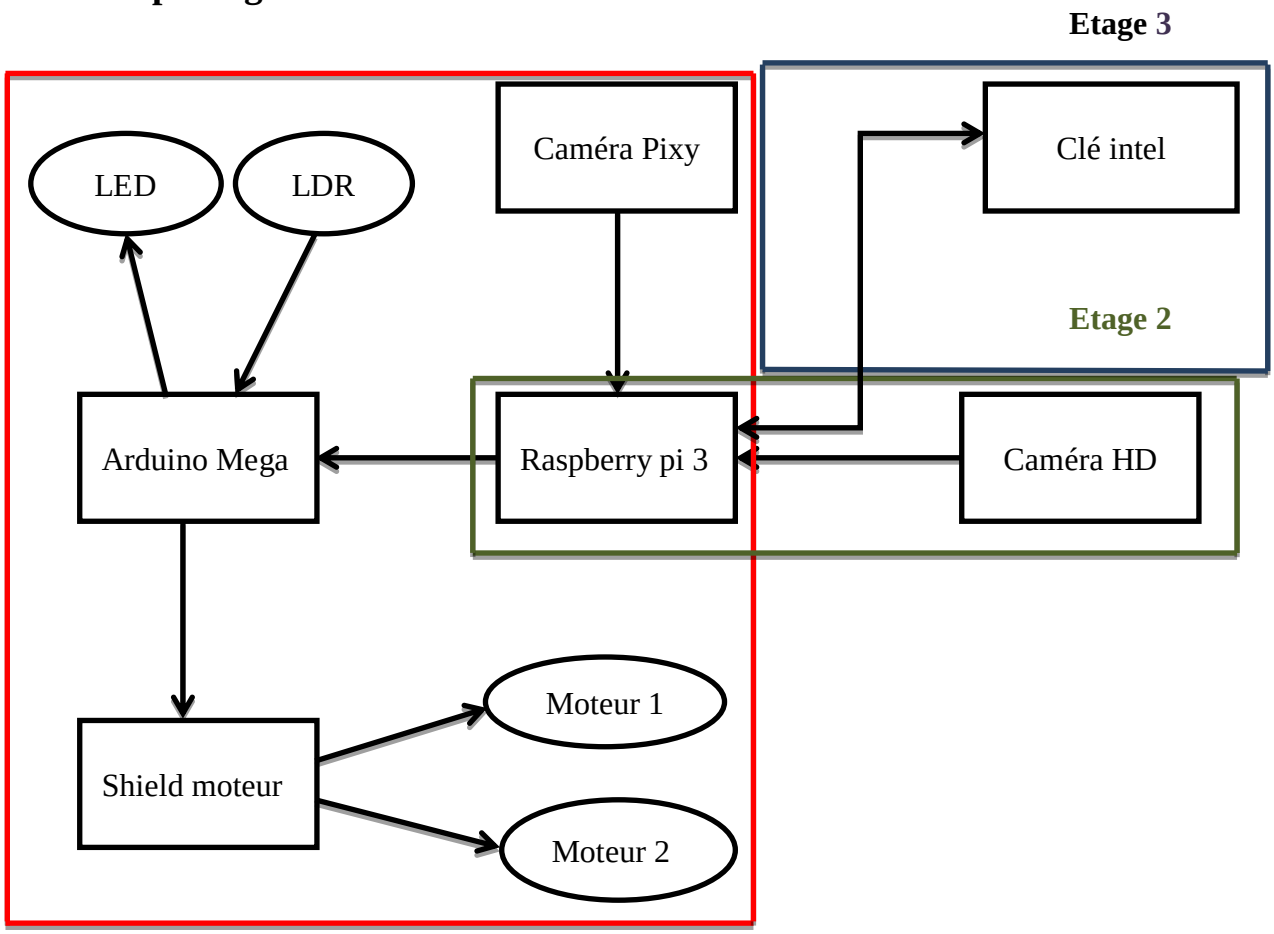


Figure IV.1: Schéma Synoptique du robot mobile

Le schéma bloc ci-dessus decrit le fonctionnement global du robot mobile que nous avons conçu. Celui-ci est composé de trois étages :

Le premier est dédié à l'opération de suivi de la trajectoire décrite par une ligne de couleur. Cette trajectoire est détectée grâce à la camera pixy. Les données issues du processus de traitement

d'image sont transmises au Raspberry qui calcul les consignes nécessaires au suivi de la trajectoire. Ces consignes sont transmises, via la liaison usb, à la carte Arduino / Shield qui se charge d'asservir les roues du robot mobile.

Le deuxième étage est destiné à la détection des plaques de signalisation. Cette opération nécessite l'utilisation d'une deuxième camera. Les images qu'elle fournira au Raspberry serviront au réseau de neurones CNN à décider de l'action du robot en fonction de la plaque de signalisation rencontrer lors du parcours.

Enfin le dernier étage permet l'implémentation de la clé Neural Compute Stick 2 d'Intel dont le rôle est de décharger le Raspberry en gérant le processus détection/décision des plaques de signalisation.

L'ensemble des 3 étages qui constituent notre plate-forme robotique est présenté par les photos suivantes :

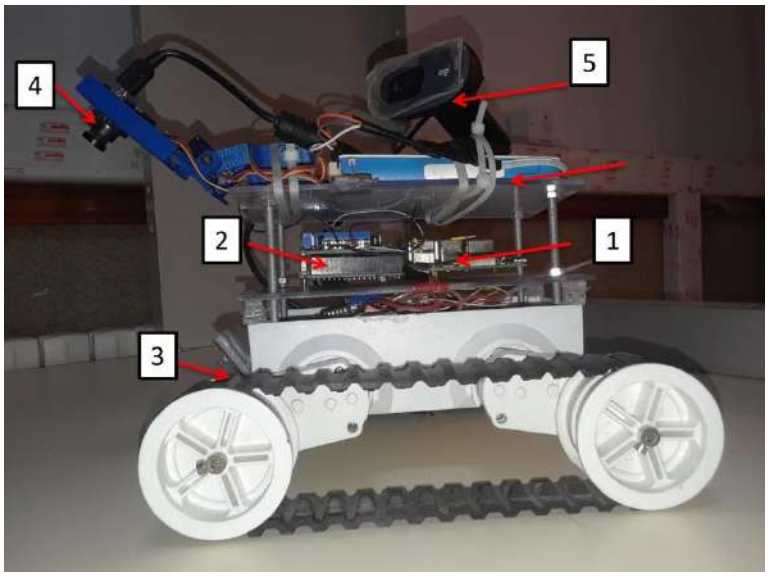


Figure IV.2 : vue du robot mobile

1 : La carte Raspberry pi 3

2 : L'ensemble Arduino mega/Shield

3 : Le châssis du robot mobile à chenilles

4 : La caméra Pixy

5 : La caméra HD

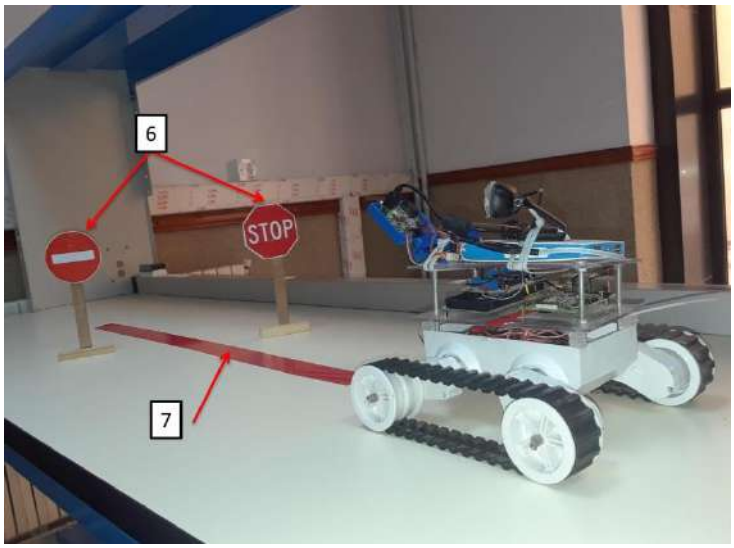


Figure IV.3 : vue du robot mobile dans son environnement

6 : les plaques de signalisation

5 : la trajectoire

Étage I : L'asservissement visuel du robot mobile

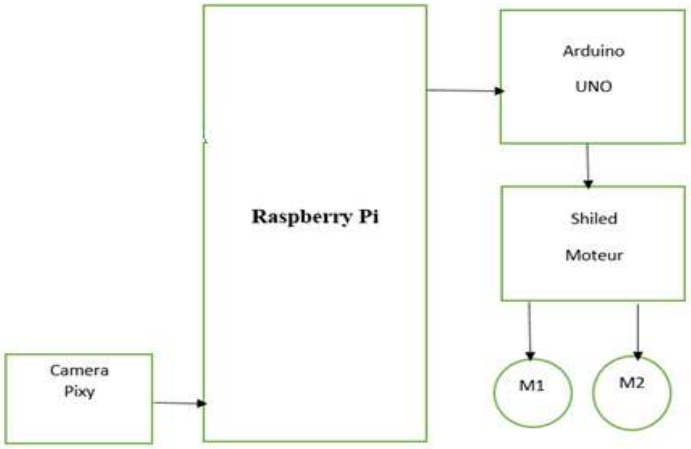


Figure IV.4 : Synoptique de l'étage asservissement visuel du robot

La figure IV.4 représente le schéma synoptique de l'ensemble robot mobile et le système de commande embarquée que nous avons réalisé. Ce dernier s'articule autour de 4 blocs principaux. Les deux premiers blocs représentent l'association du robot mobile (avec ces moteurs Îlectriques à courant continu) et de la carte Arduino qui sert à gérer la commande bas niveau des deux moteurs du robot. Tandis que les deux derniers blocs regroupés au sein de la carte Raspberry et qui ont pour rôle

le calcul des lois de commandes nécessaire au suivi de la trajectoire à partir des données fournies par le système de vision Pixy.

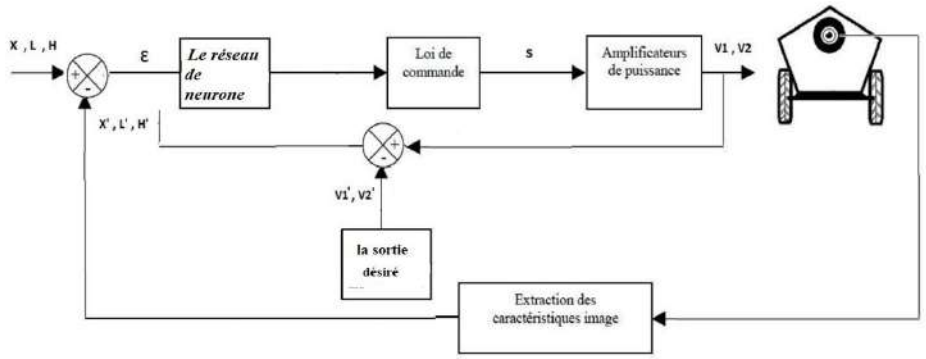


Figure IV.5 : Synoptique de notre schéma d'asservissement visuel

Notre schéma de contrôle consiste à calculer directement les valeurs des commandes à partir des caractéristiques de la cible dans les images. L'erreur de positionnement est calculée directement dans l'espace images et c'est cette variable que le contrôleur devra réduire. Ces caractéristiques sont extraites à partir des informations issues de la caméra Pixy. Lorsque la trajectoire cible désignée apparait dans le champ de la caméra, le traitement d'images va la repérer et va nous fournir un certain nombre de données.

L'utilisation de la vision monoculaire, ne permet pas d'estimer la profondeur de la trajectoire. Pour contourner cette limitation (en évitant d'introduire une deuxième caméra), nous avons calculé la surface de la trajectoire cible. Si la valeur de surface grandie, la cible est proche de la caméra et donc du robot et inversement si, la valeur de la surface diminue, la cible s'éloigne du robot. Par conséquent, cette approche nous permettra d'asservir et de suivre la trajectoire cible quand le robot se déplace en perpendiculaire du plan image.

Quant aux déplacements sur le plan parallèle au plan image, nous utiliserons les coordonnées (estimées en pixels) X, Y du centre de gravité de la trajectoire cible.

En réalité, nous utilisons uniquement la coordonnée X pour maintenir le centre de gravité de la trajectoire au centre de l'image.

Grâce à l'estimation de la position de la trajectoire dans l'espace cartésien (faite à partir des données visuelles L, H et X), nous avons contraint le robot à maintenir la cible à une distance donnée et surtout au centre de l'image. C'est le principe même de l'asservissement visuel avec une approche basée sur l'image.

L'organigramme suivant décrit la commande bas niveau du robot mobile :

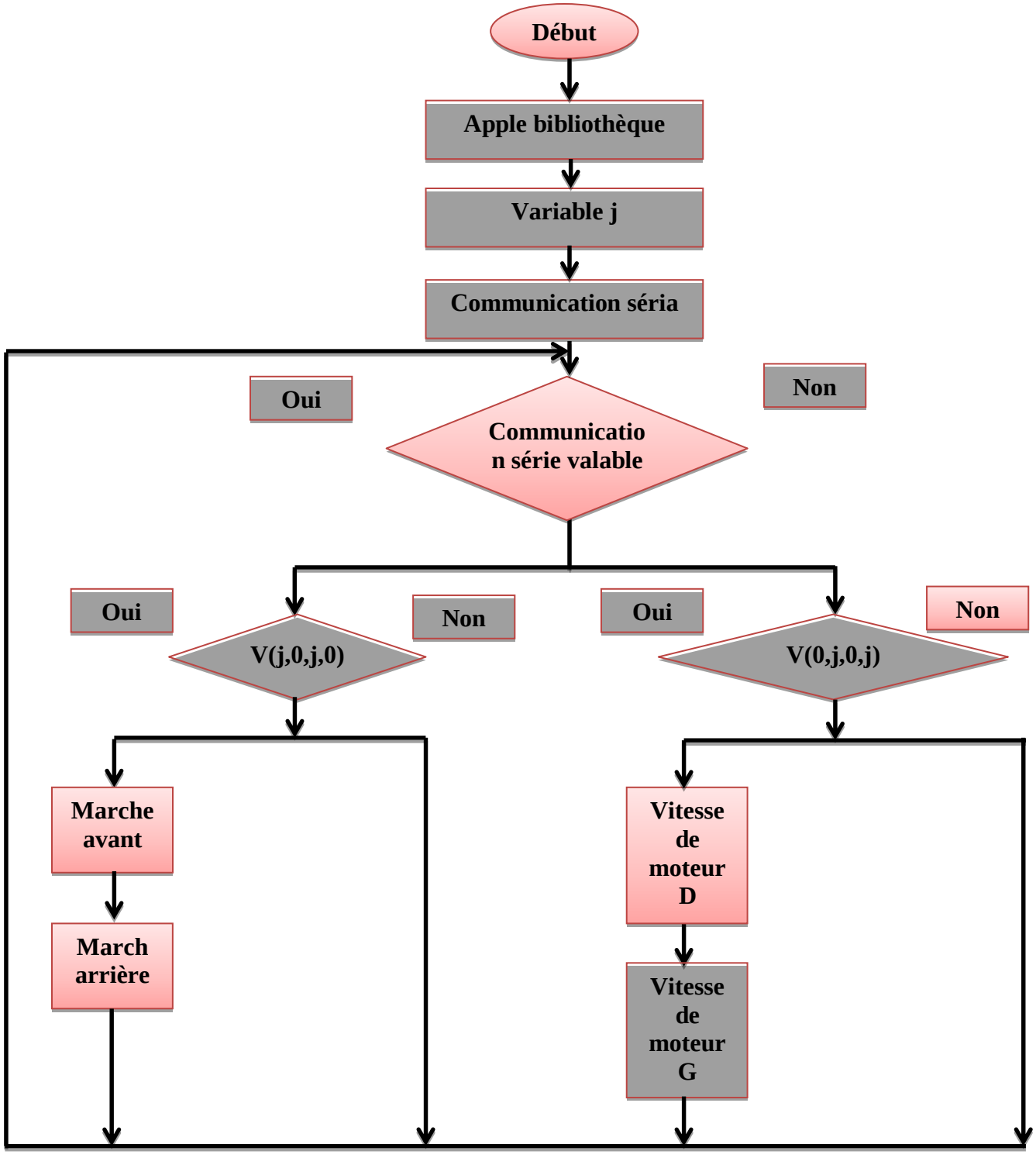


Figure IV.6 : Organigramme de commande des moteurs du robot

Ce programme est implémenté dans la carte Arduino et tourne dans une boucle infinie. Lorsque le Rapspberry calcul les consignes pour corriger le mouvement du robot, il envoi à la carte Arduino un vecteur de 4 chiffres qui représente respectivement le sens, la vitesse du moteur 1, le sens, la vitesse de moteur 2.

Lors de nos nombreux tests, nous avons rencontré le problème de perte de données visuels dans les zones sombre de la trajectoire. La caméra Pixy avait du mal à distinguer notre trajectoire et fournir les informations visuelles de celle-ci. Pour résoudre ce problème, nous avons doté notre plate-forme d'un capteur de luminosité qui, à partir d'un certain seuil de lumière, allume des LED pour éclairer la trajectoire.

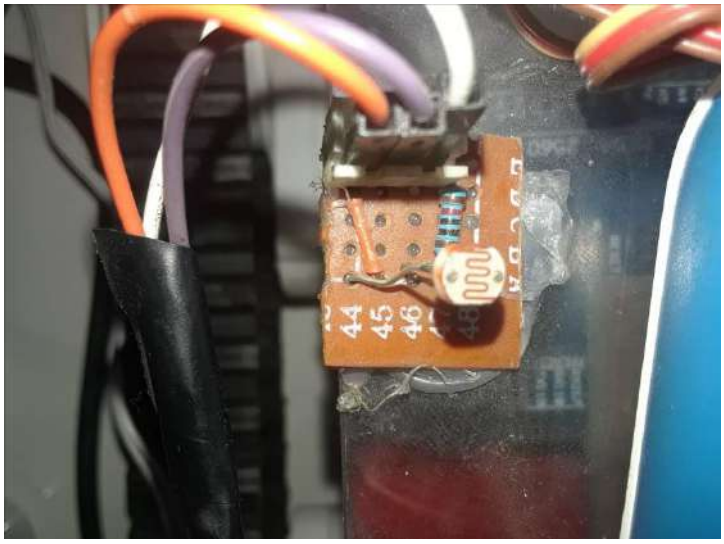


Figure IV.7 : capteur LDR



Figure IV.8 : Éclairage de la trajectoire

Étage 2 : Détection intelligente des plaques de signalisation

Cet étage est dédié à la détection des plaques de signalisation qui balises le parcours du robot. Il utilise une deuxième camera HD relié au Raspberry sur lequel est exécuté un algorithme intelligent qui, à partir des images acquises, détecte les plaques de signalisation qui se trouve sur le parcours et décide de l'action du robot en fonction de celle-ci. Nous avons dans un premier temps testé une application (Cascade) dédiée à cet effet et évaluée ses performances puis nous avons implémenté et tester un réseau de neurones Convolutif CNN.

Les résultats obtenus sont détaillés dans cette partie. Mais avant des réaliser ces évaluations, nous avons dû surmonter un problème d'envergure. En effet, la richesse d'informations du capteur visuel nous a poussé à introduire une technique de sélection de la plaque la plus proche pour travailler avec cette donnée. Cette technique utilise la bibliothèque OpenCv décrite précédemment. Le programme que nous avons conçu est décrit par l'organigramme ci-dessous :

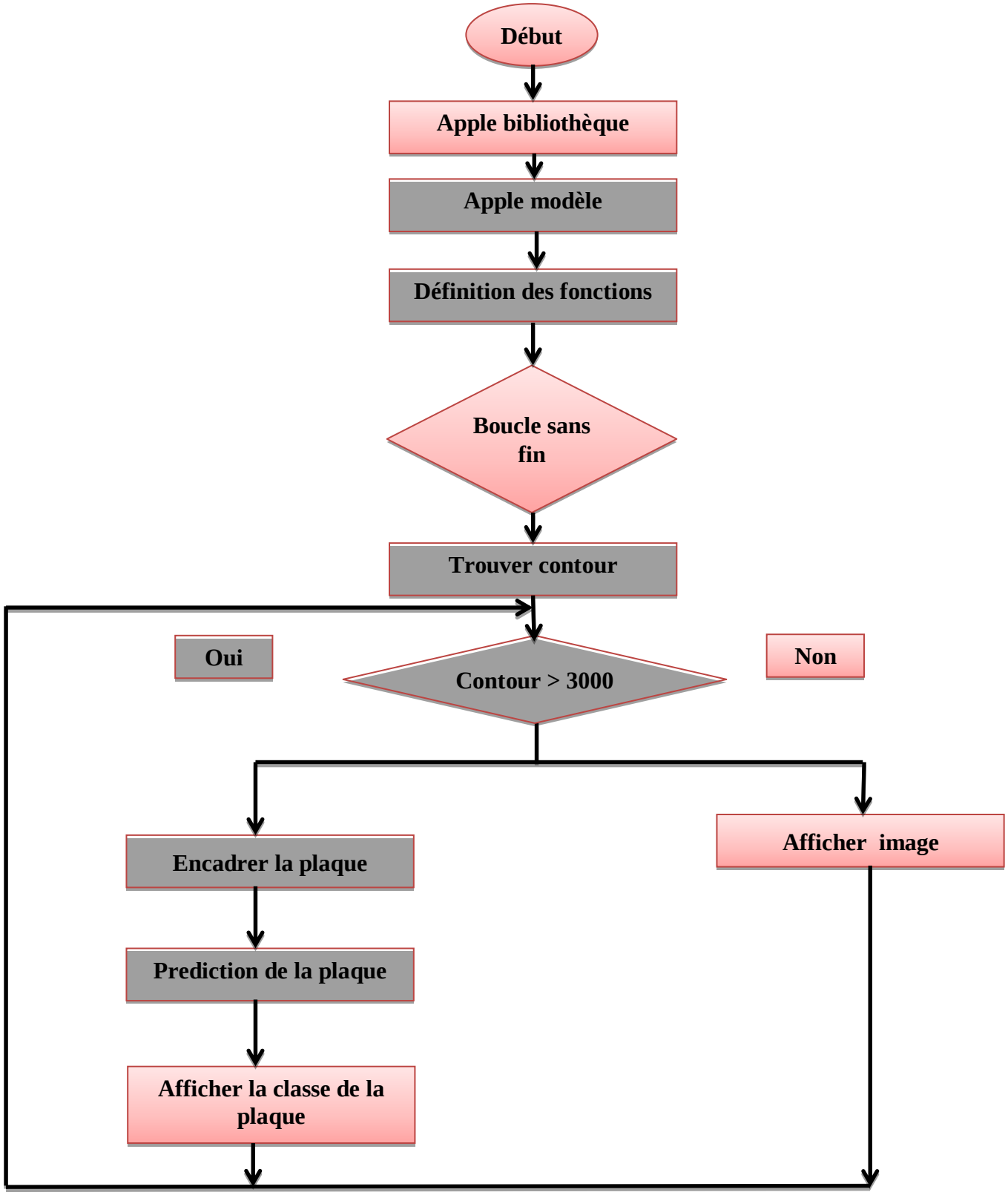


Figure IV.9 : Organigramme de encadrer les plaque

La figure suivante montre les résultats obtenus lors de la présentation des plaques de signalisation en face de la caméra HD.

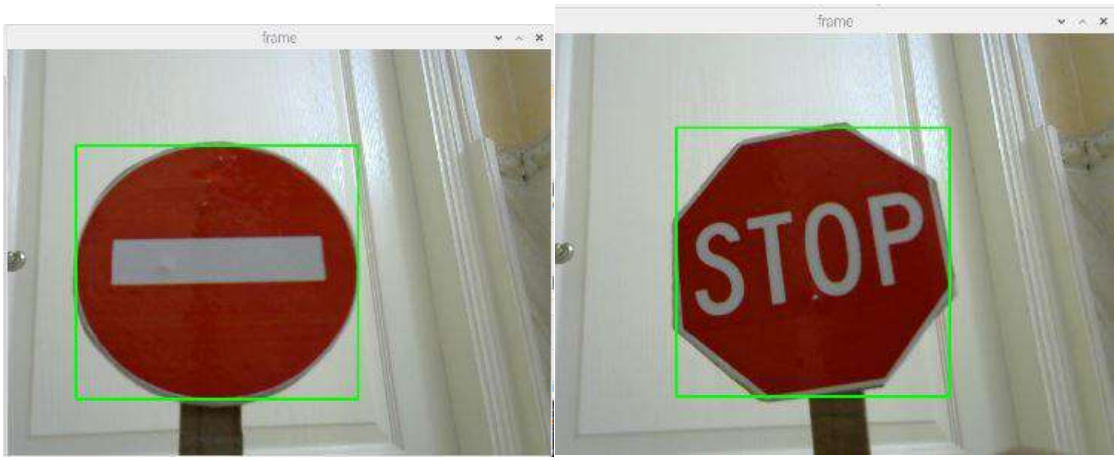


Figure IV.10 : Sélection des plaques dans les images

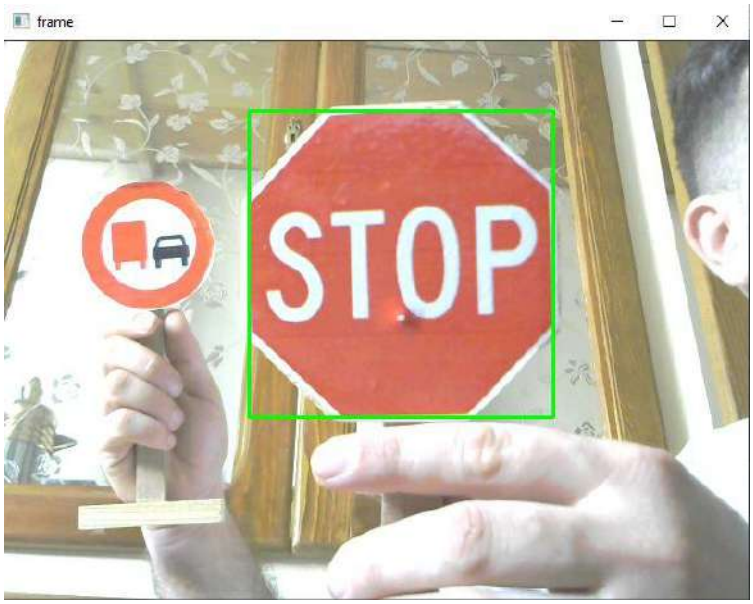


Figure IV.11 : Sélection de la plaque pertinente dans une image contenant 2 plaques

Au regard de la dernière figure, nous avons grâce à notre algorithme résolu notre problème de détection de plaque pertinente dans des images contenant d'autres informations.

Cette étape franchie, nous pouvons implémenter nos algorithmes de détection et de décision pour guider notre robot mobile dans le respect de la signalisation.

3. Création du jeu de plaques signalisation à respecter

Le jeu de plaques de signalisation est composé de plusieurs photos de chacune d'entre elles. Plusieurs prises de vue d'une même plaque ont été réalisés et regroupées pour que notre système puisse les détecter dans n'importe quelle position. L'ensemble de chaque plaque est stocké dans un dossier précis. Ensuite, un fichier Excel est généré pour associer une plaque (un dossier donc) à une classe. Ceci est illustré par la figure ci-dessous.

	A	B
1	ClassId	Name
2	0	Speed limit (20km/h)
3	1	Speed limit (60km/h)
4	2	No passing for vechiles over 3.5 metric tons
5	3	Priority road
6	4	Yield
7	5	Stop
8	6	Vechiles over 3.5 metric tons prohibited
9	7	No entry
10	8	General caution
11	9	Dangerous curve to the left
12	10	Turn right ahead
13	11	Turn left ahead
14	12	Ahead only
15		

Figure IV.12: le fichier associant les plaques aux classes.

Notre jeu est constitué des treize de plaques de signalisation.

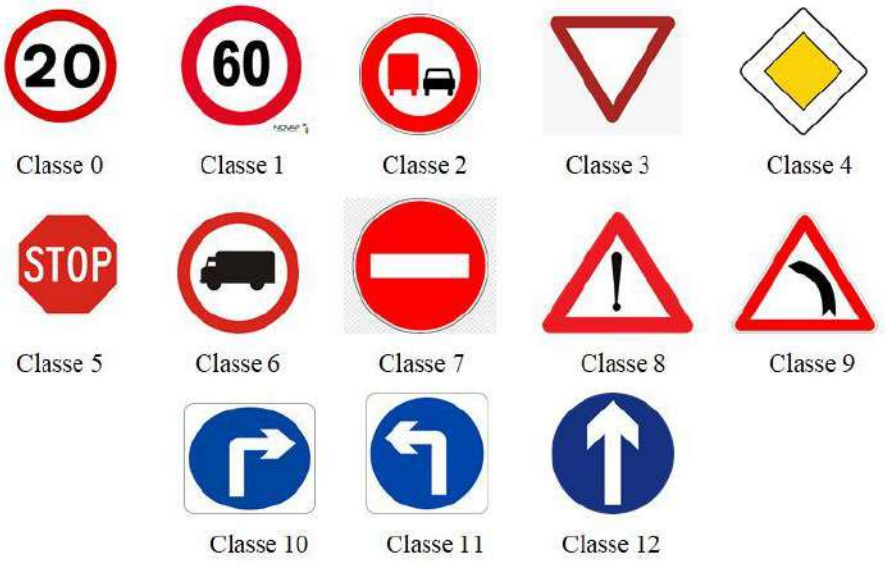


Figure IV.13: classification des plaques de signalisation

C'est ce jeu de plaques de signalisation que nous allons utiliser pour faire apprendre nos algorithmes intelligents (de détection/décision) nécessaires au guidage de notre robot mobile.

4. Choix de l'architecture du réseau de neurones CNN

Comme nous l'avons présenté dans le chapitre 2 de ce document, les réseaux CNN sont composés de plusieurs couches différentes (figure IV.14) :

- Couche d'entrée (Input layer).
- Couche de convolution (Convo layer : Convolution + ReLU).
- Couche de Pooling.
- Couche entièrement connectée (Couche Fully connected).
- Couche Softmax/logistique.
- Couche de sortie (Output layer)

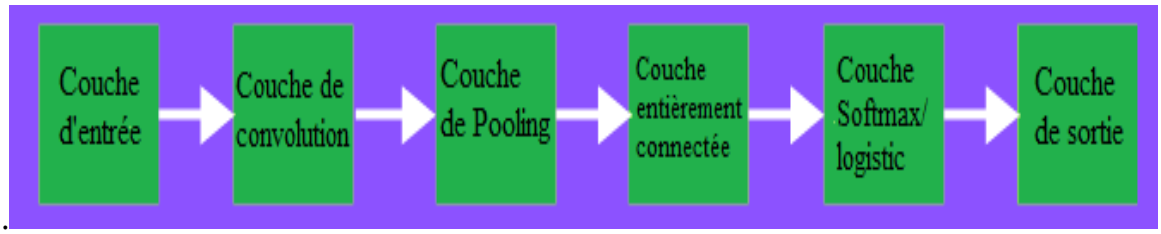


Figure IV.14 : Les couches de CNN

Ces couches et ces paramètres définissent, d'une part, l'architecture du réseau et d'autre part, influent considérablement sur les performances du CNN. Leur sélection judicieuse est donc importante pour obtenir de bonnes précisions. Pour déterminer l'architecture optimale du réseau de neurones CNN, nous avons effectué un grand nombre de tests. Un résumé des principales architectures est décrit par le tableau ci-dessous.

	Architectures	**Les paramètres**	**Résultats**
1	Convolution Convolution Max pooling Convolution Convolution Max pooling	Matrice de convolution : (4,4) Matrice de max pooling : (2,2) Image : (64,64)	**Précision : 0.749**
	Convolution	Matrice de convolution :	

	Architectures	Les paramètres	Résultats
2	Max pooling Convolution Max pooling	(3,3) Matrice de max pooling : (2,2) Image : (32,32)	**Précision : 0.877**
3	Convolution Convolution Convolution Max pooling Convolution Convolution Convolution Max pooling	Matrice de convolution : (4,4) Matrice de max pooling : (2,2) Image : (32,32)	**Précision : 0.926**
4	Convolution Convolution Max pooling Convolution Convolution Max pooling	Matrice de convolution : (3,3) Matrice de max pooling : (2,2) Image : (32,32)	**Précision : 0.998**

Table IV.1 : les tests de l'architecteur de CNN

Dans un 1er temps, nous avons opté pour une architecture simple qui nous a donné une précision moyenne de 74%. Cette performance ne correspondait pas à nos objectifs car le robot confondait des plaques dans certaines condition (éclairage, orientation de la plaques, …). C'est pour cela que nous avons augmenté la complexité de l'architecture du réseau en rajoutant des couches de convolutions et joué sur la taille du filtre jusqu'à obtenir le dernière architecture qui nous fournit une précision de 99.8%. C'est la performance maximale que nous avons pu obtenir. Augmenter la complexité du réseau ne rajoutera qu'une surcharge de calcul.

Cette précision élevée et du essentiellement à la simplicité des plaques de signalisation. On obtient des performances moindres avec ce type de réseau quand les caractéristiques image à détecter sont complexes (les visages humains par exemple). Ceci étant, dans notre cas cette précision nous convient parfaitement, car en implémentant ce modèle sur notre robot mobile, les

résultats du guidage étaient excellents. A chaque fois qu'il rencontre une plaque il réagissait en fonction de celle-ci (par exemple il s'arrête pour 5 secondes quand il rencontre la plaque STOP).

5. Classification par l'application Cascade

Comme notre problème de guidage s'apparente à un problème de classification, nous avons décidé de comparer les résultats obtenus avec le CNN avec l'application Cascade.

En effet, Cascade est une application qui permet de tester très facilement des classificateurs pour la détection et le suivi d'objets à l'aide d'OpenCv en fournissant une interface utilisateur graphique pour définir les paramètres et effectuer les étapes nécessaires.

Après son installation sur le Raspberry, nous avons inséré les photos des panneaux de signalisation dans un dossier spécial nommé Positive. Nous avons également pris des photos de de l'environnement de la trajectoire du robot que nous avons inséré dans le dossier nommé négatif.

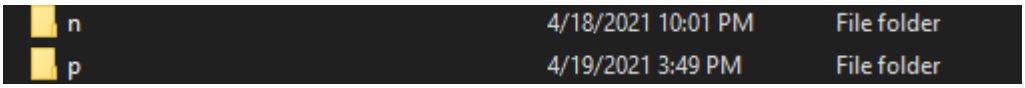


Figure IV.15 : les fichiers dataset de cascade

Ensuite, il faudra indiquer le répertoire du jeu de données et le nombre d'images traité comme l'illustre la figure suivante :

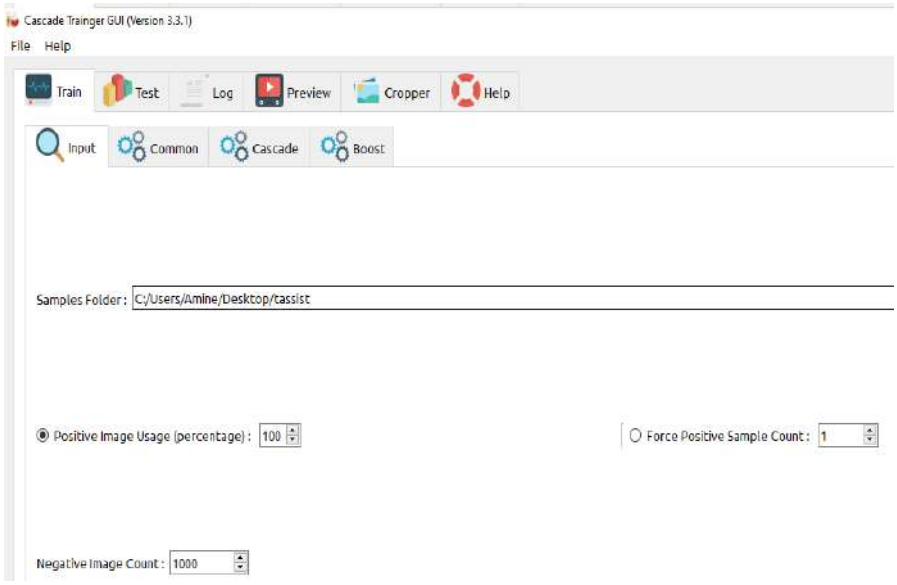


Figure IV.16: enregistrement du ficher du jeu de données dans logiciel

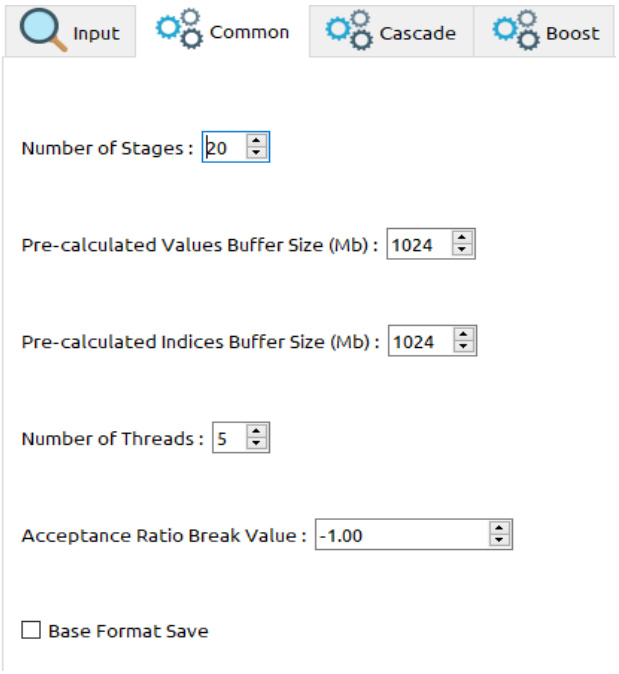


Figure IV.17 : Les paramétrage de l'application

Une fois le paramétrage terminé, l'application débute la phase d'apprentissage pour générer le modèle du classificateur.

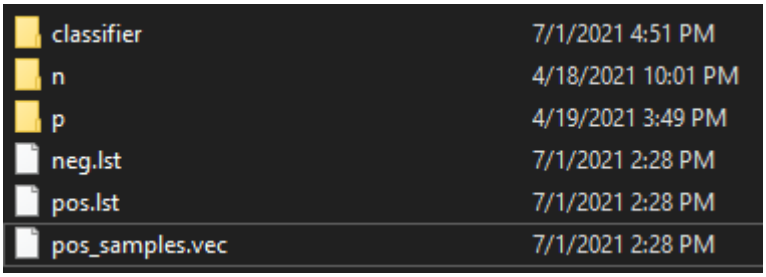


Figure IV.18: Résultat après le traitement

L'utilisation de cette application dans notre cas apporte une souplesse d'implémentation et une facilité dans le développement du classificateur. Sa mise en œuvre sur notre robot expérimental nous a permis d'obtenir un bon guidage. Cependant, cette approche montre des limites lorsqu'on change d'environnement du robot et lors de la modification de l'éclairage des plaques. En effet, les performances du CNN sont meilleures dans le sens ou les performances de cet algorithme neuronale est indépendantes de l'environnement extérieur dans lequel évolue le robot et de ce fait, il lui procure une autonomie indéniable.

Étage 3 : Mise en œuvre de la clé Neural Compute Stick 2 d'Intel

Le cahier des charges initial de notre projet n'incluait pas cette partie. Mais comme Elle était disponible au laboratoire, nous avons tenue à la mettre en œuvre sur la plate-forme robotique pour pouvoir la tester et connaitre ses performances. Nous voulions également vérifier si elle pouvait réduire la charge sur le processeur du Raspberry et réduire les temps d'exécution.

Mais tout d'abord, nous vous présentons les étapes nécessaires à la mise en œuvre de cette clé et surtout des précautions à prendre pour garantir son bon fonctionnement.

6. Installation de la clé Neural Compute Stick 2 d'Intel

Pour l'installation nous suivons les étapes suivantes :

Étape 1 : Rassemblez votre équipement



Figure IV.19: placer le clé sur La carte Raspberry pi 3

Étape 2 : Installez la boîte à outils OpenVINO™

```
1. cd ~/Downloads

2. tar xvf l_openvino_toolkit_<VERSION>.tgz

3. cd l_openvino_toolkit_<VERSION>

4. sudo -E ./install_openvino_dependencies.sh

5. ./install_GUI.sh
```

Étape 3 : Configurer le pilote USB Neural Compute Stick

1. source ~/intel/openvino/bin/setupvars.sh
2. cd ~/intel/openvino/install_dependencies
3. ./install_NCS_udev_rules.sh

Étape 4: Tester l'installation

1. cd ~/intel/openvino/deployment_tools/model_optimizer/install_prerequisites/
2. ./install_prerequisites.sh
3. cd ~/intel/openvino/deployment_tools/demo
4. ./demo_squeezenet_download_convert_run.sh -d MYRIAD

Recherchez des résultats comme celui-ci qui indiquent une installation réussie

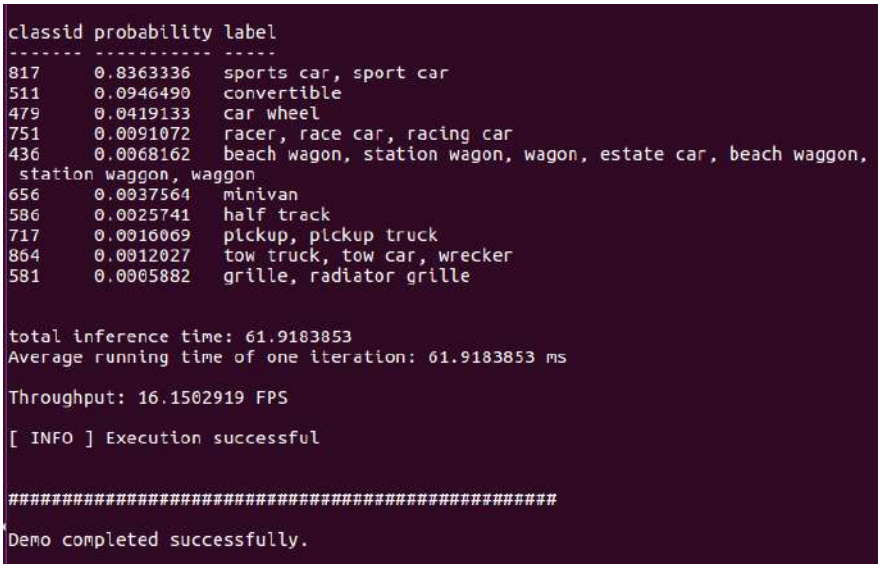


Figure IV.20 : résultat de confirmation d'installation de la clé.

PRECAUTIONS

La clé ne peut pas être brancher direct dans porte USB de Raspberry pi3 parce qu'elle a besoin d'une puissance que le Raspberry ne peut fournir et au risque de le mettre HS. C'est pour cela qu'il faut utiliser un Hub usb alimenté avec un chargeur 5V.

7. Conversion et chargement du modèle CNN dans la clé

Le modèle doit être converti, grâce à une application, à partir du fichier *.h5 vers les fichiers *.Xml et *.Bin comme le montre la figure suivant :

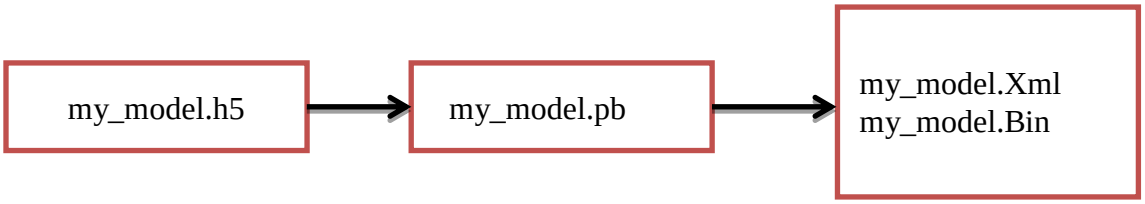


Figure IV. 21: conversion du modèle du réseau CNN.

8. Résultats comparatifs

Après l'installations de clé et l'exécution de notre algorithme neuronal de guidage nous avons comparé la charge du processeur et le temps d'exécution du Raspberry seul et avec la clé d'Intel. Les résultats obtenus sont résumés dans le tableau suivant.

	Raspberry pi3 seul	**Raspberry pi3 + Neural Compute Stick 2 d'Intel**
Charge du processeur	62%	48%
Temps d'exécution	15ms	5.98 ms

Table IV.2 : les tests de la clé et Raspberry pi 3

Au regard des résultats obtenus, nous pouvons dire que l'introduction de cette clé améliore le fonctionnement du système robotique. En effet, elle permet de **diminuer de 30%** environ la charge d'utilisation du processeur du Raspberry et de **diviser par trois (3)** le temps d'exécution. Cet apport nous permet d'envisager d'autre taches que le Raspberry pourra effectuer sans baisser les performances du l'asservissement du robot mobile.

9. Conclusion

Dans ce chapitre, nous avons présenté le fonctionnement détaillé et les modifications majeures apportées à la plate-forme robot mobile pour réaliser le suivi d'une trajectoire et le respect des plaques de signalisation qui jalonne ce parcours. Ces objectifs ont été atteint grâce à l'exploitation de données visuelles par des algorithmes basés sur l'intelligence artificiel.

Dans un premier temps, nous avons étudié et décrit l'agencement et l'organisation de notre système embraqué composé du système de vision Pixy de la carte Raspberry Pi3 et de la carte Arduino / Shield. Celles-ci grâce à leur interconnexion, via le port USB, communiquent et s'échangent les données nécessaires à la commande du robot mobile.

Dans un second temps, nous avons étudié et évalué le réseau de neurones CNN utilisé pour la détection/décision de plaques de signalisation. Ce contrôleur reconnaissait avec une précision importante ces plaques (quel que soit leurs orientations ou l'environnement dans lequel elles se trouvaient). Ces performances nous ont permis de guider le robot mobile dans cet environnement balisé. De plus, nous avons pu démontrer l'autonomie de ce robot mobile.

Enfin, nous avons intégré à notre plate-forme robotique une clé de nouvelle génération conçue par Intel et dédié à l'exécution de CNN. Les expérimentations réalisées nous ont permis de réduire à la fois la charge du processeur et le temps d'exécution. Ce gain permettra d'envisager d'autre tâches pour le Raspberry en toute sérénité.

Bien entendu, d'autres tests sont à prévoir pour étudier de plus près les performances de notre contrôleur. Mais en tout état de cause, et au vu des résultats obtenus, nous avons rempli les conditions qui nous ont été imposées, à savoir la conception et la réalisation d'une plateforme évolutive, intelligente et embraquée pour asservir un robot mobile à partir de données visuelles.

Conclusion générale

Conclusion général

Au terme de ce travail, nous avons apporté des modifications majeures à la plate-forme robot mobile pour réaliser le suivi d'une trajectoire et la détection de plaques de signalisation qui balises ce parcours à partir des données visuelles uniquement. Cet apport se matérialise par l'implémentation d'un contrôleur basé sur réseau de neurones convolutionnel (CNN).

La solution que nous avons adoptée dans notre projet nous a permis d'obtenir des performances importantes en termes de précision de détection des plaques de signalisation. Ces performances ont engendré un guidage optimal du robot mobile dans cet environnement balisé. De plus, son autonomie a été accrue.

Enfin, l'intégration à notre plate-forme robotique de la clé Neural Compute Stick 2 d'Intel a permis de réduire à la fois la charge du processeur et le temps d'exécution. Ce gain permettra d'envisager d'autre tâches pour le Raspberry en toute sérénité.

Par ailleurs, les différents tests expérimentaux que nous avons entrepris, nous ont permis de valider le fonctionnement de cette nouvelle version de la plate-forme mais aussi de mettre en évidence les performances de l'algorithme de commande que nous avons élaboré. En effet au vu des résultats obtenus lors des différentes expérimentions, nous pouvons conclure que notre contrôleur neuronal a rempli son rôle avec des performances importantes.

Enfin, ce travail nous a permis d'aborder les différentes problématiques inhérentes à la robotique mobile, notre travail a permis de mettre en exergue qu'il est possible d'implémenter d'autres réseaux de neurones voir d'autres techniques intelligentes sur notre plate-forme robot-vision. Cela est dû qu'elle dispose de ressources matérielles suffisamment puissantes pour accueillir ces contrôleurs complexes.

Quant aux perspectives futures que nous envisageons pour compléter ce travail :

- Remplacer la carte Raspberry par une autre carte plus performante comme FPGA
- Ajouter des capteurs comme le GPS, capteurs de gaz et d'humidité pour une meilleur interaction avec son environnement.

Bibliographie

[1] ZAIBAK Serhane, ,BOUAISSA Mouadh " Asservissement Visuel d'un robot mobile avec évitement d'obstacle", Projet de fin d'étude, Université Yahia Fares de Médéa, année 2016

[2] A.K. Jain, J. Mao, and K.M. Mohiuddin, " Artificial neural networks : a tutorial ", Computer 29(3), pp. 31-44, 1996.

[3] M. Minsky and S. Papert, "Perceotrons : An Introduction to Computational Geomerty", MIT Press, Cambridge, Mass., 1969.

[4] S. Haykin, " Neural Networks : A Comprehensive Foundation " , Macmillan College Publishng Company, New York, 1994.

[5] J. Park and I.W. Sandberg, " Universal Approximation Using Radial-Basis-Function Networks", Neural Computation, Vol. 5, pp. 305-316, 1993.

[6] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," in Proc.Nat'l Academy of Sciences, USA 79, pp. 2554-2558, 1982.

[7] T. Kohonen, Self Organization and Associative Memory Third Edition, Springer-Verlag, New York, 1989.

[8] M.Z. Mokri, " Classification des images avec les réseaux neurones convolutionnels " , Mémoire de PFE, Université de Tlemcen, 2017.

[9] N. Zaarour, " Modélisation d'un canal minier ultra large bande (UWB) en utilisant les Réseaux de neurones artificiels RBF ", Québec, 2013.

[10] Lionel Pibre, Marc Chaumont, Dino Ienco, and Jérôme Pasquet. Etude des réseaux de neurones sur la stéganalyse. InCORESA: Compression et Représentation des Signaux Audiovisuels, Nancy, France, May 2016.

[16] BENAISSA el mahdi, BENYAIHIA Dalal, " Asservissement Visuel d'un robot mobile", Projet de fin d'étude, Université Yahia Fares de Médéa, année 2016

Webographie

[11] https://towardsdatascience.com/covolutional-neural-networkcb0883dd6529?Bled=IwAR0UWoPkFYTEAqFitcR4fuuQUcNBvxV8ig1oJGZ3EhbRypu8Qf9pk9CXdy4

[12]https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-1b-relu-layer/

[13] N. Smith, "Intel® Neural Compute Stick 2 and Open Source OpenVINO™ Toolkit," 2019. Disponible: https://software.intel.com/en-us/articles/intel-neural-compute-stick-2-and-open-source-openvino-toolkit

[**14]** A. Stahl, "New Neural Compute Stick 2 by INTEL," 2019. Disponible: https://www.codico.com/en/News/neural-compute-stick-intel.htm

[15] Intel. (2016). Deep Learning Inference. Disponible: https://software.intel.com/enus/openvino-toolkit/deep-learning-inference

[17] ATMEL, "8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System ProgrammableFlash"

[18] Anthony Rowe, Rich LeGrand, Scott Robinson, "*Aperçu sur CMUcam5 Pixy*", http://cmucam.org/projects/cmucam5