الجـــمـــهـــوريـــــــة الجـــــــزائـــريـــة الديمـــــقـــراطـيـــة الـشـــــــعـبـيـــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التـــــــعـــــليـــــم العـــــــــالي و البـــــــــحث العـــــــلـــــمـــــي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جـــــــــــامعة أبي بـكـــــــر بـــلـــقـــايـد – تـــلمســـــــــــــان –
Université Aboubakr Belkaïd – Tlemcen –
Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : (Automatique)

Spécialité : (Automatique et Informatique Industrielle)

Par :

Mr. BOUHARAOUA Mouadh
Mr. KEBIRI Mohammed

Sujet

Outils de vision par ordinateur et d'apprentissage profond pour la détection de prise en robotique.

Soutenu à distance, le 30 /09 / 2020, devant le jury composé de :

Mme. CHOUKCHOU-BRAHAM Amel	Professeur	Présidente
Mr. HADJ Abdelkader	Professeur	Directeur de mémoire
Mlle. HANDOUZI Wahida	Maitre de conférences	Co- Directeur de mémoire
Mr. MECHERNENE Abdelkader	Maitre de conférences	Examinateur 1

Année Académique: 2019/2020

Outils de vision par ordinateur et d'apprentissage profond pour la détection de prise en robotique

6 octobre 2020

Remerciements

Tout d'abord, nous tenons à remercier ***Allah*** le tout puissant et miséricordieux, qui nous a donné la sagesse, la volonté, la force, la patience d'accomplir ce modeste travail et l'audace pour dépasser toutes les difficultés.

Ensuite, nous tenons à remercier ***Mme CHOUKCHOU-BRAHAM Amel***, Professeur à l'université Abou Bekr Belkaid – Tlemcen, d'avoir accepté d'être la présidente du jury. Un grand merci à notre encadreur ***Mr HADJ Abdelkader Amine***, Professeur à l'université Abou Bekr Belkaid – Tlemcen, département de génie électrique et électronique, Vous avez bien voulu nous confier ce travail riche d'intérêt et nous guider à chaque étape de sa réalisation. Vous nous avez toujours réservé le meilleur accueil, Vos encouragements inlassables, votre amabilité et votre gentillesse méritent toute admiration. Nous saisissons cette occasion pour vous exprimer notre profonde gratitude tout en vous témoignant notre respect.

Nous tenons également remercier ***Mr MECHERNENE Abdelkader***, Maître de conférences à l'université de Abou Bekr Belkaid – Tlemcen, pour avoir contribué par sa participation à l'examination de ce travail.

Nous adressons aussi nos remerciements à ***Mlle HANDOUZI Wahida***, Maître de conférences à l'université Abou Bekr Belkaid – Tlemcen, Pour ses nombreux conseils et soyez patiente pour répondre à nos innombrables questions.

Enfin, nous diéserons remercier très sincèrement toutes nos familles et nos amis pour leur encouragements et leur soutien inestimable.

Mohamed & Mouad

Dédicace

A mes chers parents ***Moussa & Fatma*** qui sont toujours sacrifiés pour me voir réussir, vous avez été à mes côtés pour me soutenir et m'encourager. Que ce travail traduit ma gratitude et que Dieu vous procure une longue vie et surtout une bonne santé.

A mon cher frère ***Abderrahim*** pour son soutien et son encouragement.

A mes chères sœurs ***Asma***, ***Oumaima*** et ***Chifaa*** Que Dieu les protège et leurs offre le bonheur.

A mon encadreur ***Mr. HADJ Abdelkader Amine***, merci beaucoup pour avoir accepté la charge de m'encadrer et pour ses précieux conseils.

A mon co-encadreur ***Mlle. HANDOUZI Wahida***, merci pour votre aide et d'avoir accepté de répondre à nos questions, vous avez toujours été pour nous lorsque nous avons besoin d'aide, votre contribution essentielle à réaliser ce mémoire n'est pas négligeable.

A mon ami et binôme ***Mohammed***, qu'Allah te garde le meilleur pour l'avenir.

A ***toute ma famille*** pour leur soutien tout au long de mon parcours universitaire, Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fruit de votre soutien infaillible, Merci d'être toujours là pour moi.

A tous mes amis que j'ai connus.

A tous ceux que j'aime.

Mouadh

Dédicace

Je dédie ce modeste travail, fruit de mes études...

A mon très cher père ***Azzedine*** qui s'est toujours sacrifié pour me voir réussir, tu as été à mes côtés pour me soutenir et m'encourager. Que ce travail traduit ma gratitude et que Dieu te procure une longue vie et surtout une bonne santé.

A mon cœur, le miroir de ma vie, la femme qui a souffert sans me laisser souffrir, qui n'a jamais dit non à mes exigences. Ton affection me couvre me et guide, ta présence toujours à mes côtés était la source de ma force. Sans toi je serais perdu et ma vie n'aura aucun sens, mon adorable mère ***Fouzia***, je t'adore.

A mes chers frères ***Marwane, Sofiane*** et ***Rayane***, à ma chère sœur ***Selsabil***, Que Dieu les protège et leurs offre le bonheur.

A mon encadreur ***Mr HADJ Abdelkader Amine***, merci beaucoup pour avoir accepté la charge de m'encadrer et pour ses précieux conseils.

A mon co-encadreur ***Mlle HANDOUZI Wahida***, merci pour votre aide, votre contribution essentielle à réaliser ce mémoire n'est pas négligeable.

A mon ami et binôme ***Mouadh***, qu'Allah te garde le meilleur pour l'avenir

A ma famille, mes oncles et mes tantes, cousins et cousines, qui m'ont toujours encouragé.

A tous mes amis, que j'ai connu jusqu'à maintenant.

A tous ceux que j'aime. .

Mohamed

Résumé

La robotique et la vision appliquent les techniques de l'intelligence artificielle au problème de la fabrication d'appareils capables d'interagir avec le monde physique. Cela inclut se déplacer dans le monde (robotique mobile), déplacer des objets dans le monde (robotique de manipulation), acquérir des informations par détection directe du monde (par exemple, vision industrielle) et, surtout, fermer la boucle en utilisant la détection pour contrôler le mouvement.

A travers notre projet, nous avons privilégié une nouvelle méthode de détection de prise en robotique basée sur le deep learning, l'idée est d'entraîner un modèle de détection d'objets sur l'une des bases de données disponibles, ce modèle est le yolov3 avec les paramètres par défaut. Afin que nous puissions le convertir pour la détection de prises, contrairement aux methodes de détection d'objet classique, notre modèle développé sera capable de détecter toute la région de d'intérêt d'un objet, et c'est une chose très importante dans le domaine de la robotique.

Enfin, pour fermer la boucle, une méthode de contrôle du robot est nécessaire pour prendre l'objet en toute précision et sécurité, la méthode a été réalisée en simulation pour certaines raisons mais elle est valable pour un robot manipulateur réel.

Abstract

Robotics and vision apply AI techniques to the problem of making devices capable of interacting with the physical world. This includes moving around the world (mobile robotics), moving objects around the world (manipulative robotics), acquiring information by direct sensing of the world (e.g. machine vision), and most importantly, closing the loop using sensing to control the movement.

Through our project, we proposed a new robotic grip detection method based on deep learning, the idea is to train an object detection model on one of the available databases, this model is the yolov3 with default settings. In order to convert it for grasp detection, unlike the classical object detection methods, our developed model will be able to detect the entire region of interest of an object, and this is a very important thing in the field of robotics.

Finally, to close the loop, a robot control method is necessary to take the object in all precision and safety, the method was carried out in simulation for certain reasons but it is valid for a real robot anyway.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Sans remonter aux premiers concepts de machines qui ont aidé l'homme il y a bien longtemps, la naissance de la robotique se situe quelque part au milieu du XXe siècle, du croisement des besoins et des disponibilités de nouvelles technologies développées durant la seconde guerre mondiale, L'informatique, l'électronique à semi-conducteurs et la robotique sont nées presque ensemble. Aujourd'hui, nous sommes entourés d'ordinateurs et d'appareils électroniques, qui sont entrés dans une relation presque symbiotique avec l'humanité.

Pendant ce temps, la vision par ordinateur a trouvé son chemin dans des applications répandues dans l'industrie, la médecine, le service public, etc. Le défi consiste à extraire les informations pertinentes des données visuelles par un système automatisé. Il peut même être gênant pour un être humain de fournir au système une description utile de ce que sont les informations pertinentes. En général, les humains ont tendance à raisonner à un niveau perceptuel plus élevé. Une instruction telle que "saisir un objet" est généralement claire pour un opérateur humain, alors que pour un système robotisé, la tâche n'est pas du tout facile.

Avec le développement incroyable des méthodes d'apprentissage automatique et d'apprentissage prfond, qui sont largement utilisés dans les tâches de vision par ordinateur telles que la classification d'objets, la détection d'objets et la localisation, et en raison des récents résultats positifs sur le domaine, de nombreux chercheurs en robotique ont commencé à explorer l'application des méthodes d'apprentissage en profondeur dans leurs recherches.

Inspirés par ces innovations, nous avons choisi d'aller plus loin dans le domaine, en fusionnant robotique et vision par ordinateur, nous allons essayer de faire un projet basé en principe sur deux systèmes indépendants, le premier est le système de vision, avec un détecteur d'objets développé basé sur l'apprentissage profond pour extraire des informations de l'environnement externe, ces informations pourraient être des positions des objets sur une table, l'exécution sera sur une carte électronique connectée au robot, qui représente le deuxième système. Ce dernier reçoit des informations du système de vision et applique la cinématique inverse pour prendre ces objets, et peut être pour les classer.

Dans les deux premiers chapitres nous allons donner une introduction à la théorie de ce projet, dans le 1er, nous allons parler de la robotique en général, et nous allons discuter sur des bras manipulateurs, nous allons montrer les lois de la modélisation directe et inverse concernant la géométrie et la cinématique. Enfin nous allons terminer par une section descriptive concernant notre robot choisi et ses caractéristiques.

Le deuxième chapitre concerne la vision et l'apprentissage automatique, ce sont des domaines très vastes, nous allons tenter de donner l'essentiel, commençant par la vision par ordinateur, nous parlerons des bases de cette branche, traitement de l'image, etc ..., en terminant avec l'apprentissage automatique, nous allons donner une vue générale sur les méthodes utilisées par les chercheurs dans le domaine, et nous allons finir par les travaux connexes ayant déjà traité ce sujet.

Dans le troisième chapitre, nous allons décrir la méthode de réalisation de ce projet, nous le diviserons en deux parties, chaque partie représente un sous-système, à la fin nous allons citer le matériel utilisé pour ce projet.

Enfin, le quatrième chapitre contiendra les résultats des expériences précédentes, et leur discussion. Une conclusion générale permettra de résumer l'essentiel du travail accompli dans le cadre de ce mémoire.

Chapitre 1

Généralités sur les bras manipulateurs articulés

1.1 Introduction

La robotique est un domaine relativement récent de la technologie moderne qui dépasse les frontières de l'ingénierie traditionnelle. La compréhension de la complexité des robots et de leurs applications nécessite des connaissances en génie électrique, en génie mécanique, en génie des systèmes industriels, en informatique, en économie et en mathématique. De nouvelles disciplines de l'ingénierie, telles que l'ingénierie de fabrication, l'ingénierie des applications et l'ingénierie des connaissances sont apparues pour faire face à la complexité du domaine de la robotique et de l'automatisation industrielle.

Dans ce chapitre, nous allons fournir une introduction générale à la robotique, en particulier aux bras manipulateurs que nous allons considérer dans ce projet. Après ça, nous allons expliquer les principales méthodes de modélisation s et pourquoi elles sont très importantes pour contrôler les robots.

Enfin, nous allons introduire le robot choisi, et ses caractéristiques et pourquoi il est très parfait pour un projet comme celui-ci.

1.2 Généralité sur la robotique

Le robot et la robotique

Pour le sens commun, un robot est un dispositif mécanique articulé capable de faire certaines fonctions humaines telles que la manipulation d'objets ou la locomotion, dans le but de se substituer à l'homme pour la réalisation de certaines tâches matérielles. Cette réalisation est plus ou moins autonome selon les facultés de perception de l'environnement dont est doté le robot. La robotique est l'ensemble des activités de construction et de mise en œuvre des robots.

1.2.1 Les différents types des robots

1.2.1.1 Robots mobiles

Robots capables de se déplacer dans un environnement. Ils sont équipés ou non de manipulateurs suivant leur utilisation.

1.2.1.2 Robots manipulateurs

C'est des robots ancrés physiquement à leur place de travail et généralement mis en place pour réaliser une tâche précise et répétitive. Ce sont des manipulateurs automatiques programmés qui se substituent à l'homme pour l'accomplissement de tâches répétitives (tels que les bras manipulateur, médicaux, les robots industriels...).

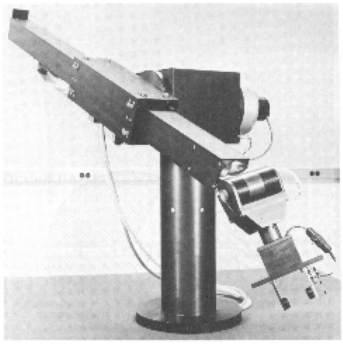


FIGURE 1.1 – Le bras articulé « The Stanford Arm ». Photo du « Coordinated Science Lab, University of Illinois at Urbana-Champaign ». [34]

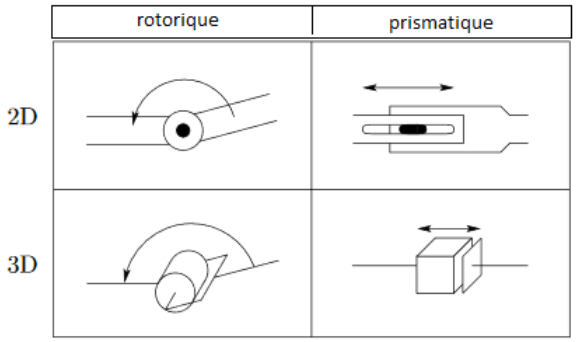


FIGURE 1.2 – Différents types de liaisons mécaniques.

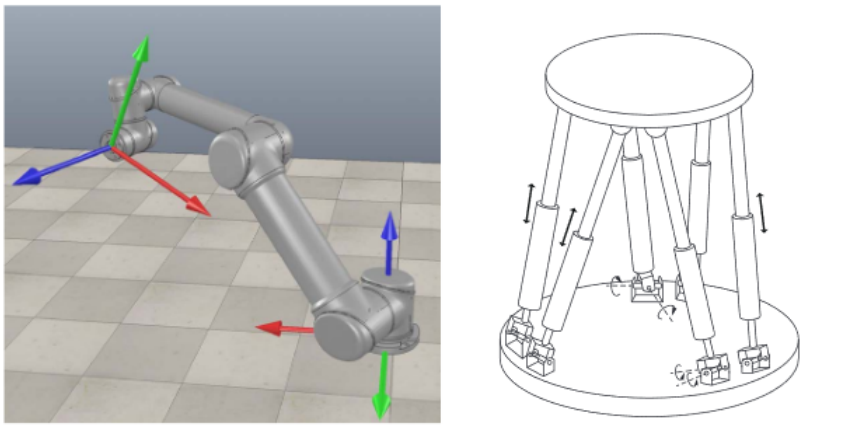
1.3 Les bras manipulateur

Les manipulateurs de robots sont composés de liens reliés entre eux pour former une chaîne cinématique. Les articulations sont généralement rotatives (révolutives) ou linéaires (prismatiques). Un articulation rotative est comme une charnière et permet une rotation relative entre deux maillons. Une articulation prismatique permet un mouvement relatif linéaire entre deux maillons. Nous désignons les articulations révolutives par R et les articulations prismatiques par P, et les dessinons comme illustré à la figure 1.2

L'actionnement des articulations, généralement par des moteurs électriques, fait alors bouger le robot et exercer ses efforts de la manière souhaitée.

Le terme Effecteur (organe terminal) est utilisé pour décrire l'interface entre le manipulateur (bras) et l'environnement. La grande majorité des effecteurs sont simples : pinces (soit à deux mâchoires), aimant, camera dans certains cas. Il peut être contrôlé à distance par un ordinateur ou par un humain.

Les liens d'un mécanisme de robot manipulateur peuvent être organisés en série, comme le bras à chaine ouverte illustré à la figure 1.3 (a). Les mécanismes du robot peuvent également avoir des liens qui forment des boucles fermées, comme la plate-forme Stewart-Gough illustrée à la figure 1.3 (b). Dans le cas d'une chaîne ouverte, toutes les articulations sont actionnées, tandis que dans le cas de mécanismes à boucles fermées, seul un sous-ensemble des articulations peut être activé.



(a) Un manipulateur industriel à chaîne ouverte, visualisé en V-REP

(b) Plateforme Stewart – Gough. Les boucles fermées sont formées à partir de la plate-forme de base, à travers les pieds, à travers la forme supérieure et à travers les pieds jusqu'à la plate-forme de base

FIGURE 1.3 – Mécanismes de robot à chaîne ouverte et à chaîne fermée.

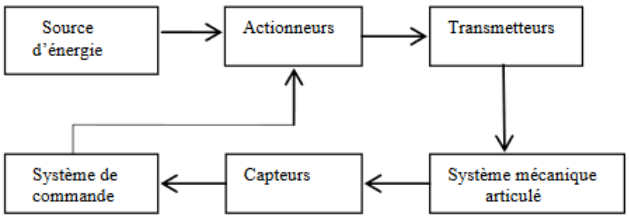


FIGURE 1.4 – Schéma synoptique d'un robot manipulateur

1.3.1 Anatomie d'un robot

Les interactions entre un système mécanique articulé et un ensemble d'organes associés forment un robot manipulateur. Le schéma synoptique de la Fig. 1.4 montre les différentes parties d'un robot.

1.3.2 Classification des robots manipulateurs

Les robots manipulateurs peuvent être classés selon plusieurs critères, tels que leur source d'alimentation, ou la manière dont les articulations sont actionnées, leur géométrie ou structure cinématique, leur domaine d'application prévu ou leur méthode de contrôle. Une telle classification est utile principalement pour déterminer quel robot convient à une tâche donnée. Par exemple, un robot hydraulique ne conviendrait pas aux applications de manipulation des aliments ou de salle blanche.

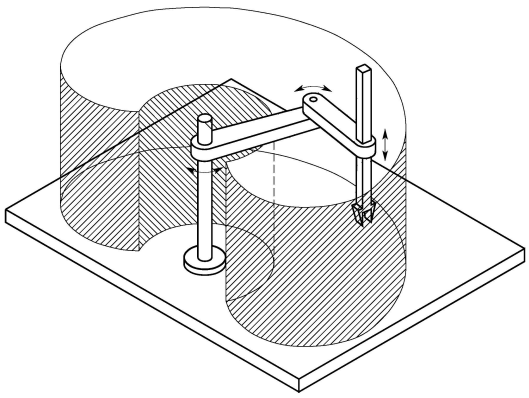


FIGURE 1.5 – Espace de travail d'un robot de type Scara.

1.3.3 Degrés de liberté (ddl)

Le nombre de degrés de liberté d'un manipulateur c'est le nombre des variables de position indépendantes qui devront être précisées afin de localiser toutes les pièces du mécanisme. C'est un terme général utilisé pour n'importe quel mécanisme. Dans le cas de robots industriels typiques, le nombre des articulations est égal au nombre de degrés de liberté, car le manipulateur est habituellement une chaine cinématique ouverte, et parce que chaque position d'articulation est généralement définie par une seule variable [34].

La liaison mécanique définit le type de degré de liberté (ddl) ou (dof) (Fig.1.2).

1.3.4 Espace de travail :

L'espace de travail d'un manipulateur est tous les point atteignables par l'organe terminal lorsque le manipulateur exécute tous les mouvements possibles. L'espace de travail est contraint par la géométrie du manipulateur ainsi que par les contraintes mécaniques sur les articulations1.5.

1.4 Modélisation de bras articulés

Le problème principal dans la modélisation est de trouver une relation entres les consignes données dans l'espace opérationnel de la tâche et des postures des éléments du robot dans l'espace articulaire.

La modélisation est l'ensemble des méthodes permettant de représenter les aspects géométriques de base de la manipulation robotique, les aspects dynamiques de la manipulation et les divers capteurs disponibles dans les systèmes robotiques modernes.

Sur la base de ces modèles mathématiques, nous serons en mesure de développer des méthodes pour planifier et contrôler les mouvements des robots afin d'effectuer des tâches spécifiées

La modélisation du robot est très importante, elle nous permet d'avoir une vision globale de l'état et du comportement du robot manipulateur.

1.4.1 Modèle géométrique

1.4.1.1 Modèle géométrique direct

Le (MGD) est l'ensemble des relations qui permettent d'exprimer la situation de l'organe terminal, c'est-à-dire les coordonnées opérationnelles du robot, en fonction de ses coordonnées articulaires. Dans le cas d'une chaîne ouverte simple, il peut être représenté par la matrice de transformation $0Tn$ [17] :

$$^{0}T_{n} = {}^{0}T_{1}(q1)^{1}T_{2}(q2)...^{n-1}T_{n}(q_{n})$$

Ce modèle ne prend pas en compte la vitesse de déplacement ni les forces et les moments qui créent le mouvement. L'expression de la situation de l'organe terminal du bras manipulateur en fonction de sa configuration est obtenue à l'aide de l'équation suivante :

$$X = f(q)$$

q étant le vecteur des variables articulaires tel que :

$q = [q_1, q_2, ...q_n]$

Les coordonnées opérationnelles sont définies par :

$X = [x_1, x_2, ...x_m]$

Plusieurs méthodes et notations ont été proposées pour trouver la solution de ce problème [?].

Une convention couramment utilisée pour sélectionner des cadres de référence dans des applications robotiques est la convention Denavit-Hartenberg, ou DH. Dans cette convention, chaque transformation homogène A_i est représentée comme un produit de quatre transformations.

$$A_i = Rot_{z,\theta i} Trans_{z,di} Trans_{x,ai} Rot_{x,\alpha i}$$

$$A_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i c\alpha_i & a_i c\theta_i \\ c\theta_i & c\theta_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

où les quatre quantités $\theta_i, a_i, d_i, \alpha_i$ sont des paramètres associés à la liaison i et l'articulation i 1.6.

1.4.1.2 Modèle Géométrique Inverse

Au contraire du modèle géométrique direct, le problème inverse consiste à calculer les coordonnées articulaires correspondant à une situation donnée de l'organe terminal, tel que :

$$q = f^{-1}(X)$$

La résolubilité du MGI, c'est-à-dire l'existence d'un nombre fini de solutions est fondamentale en matière de conception. Supposons que la situation x d'un bras manipulateur à n liaisons soit exprimée par un nombre

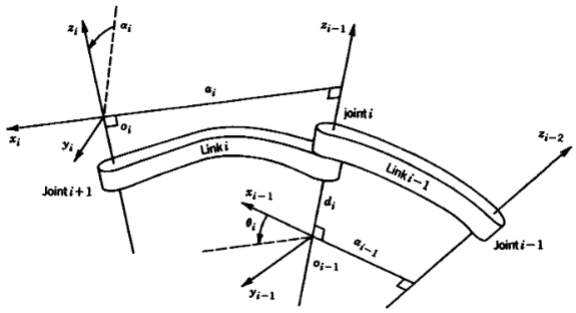


FIGURE 1.6 – Paramètres de Denavit-Hartenberg.

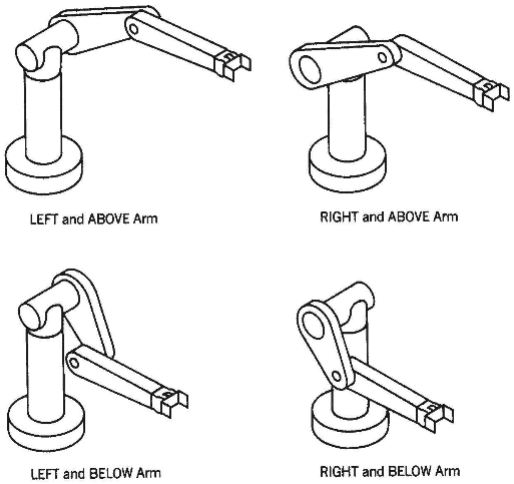


FIGURE 1.7 – Quatre solutions de la modèle géométrique inverse pour le manipulateur PUMA [33].

m minimal de paramètres. Supposons par ailleurs que x soit une situation accessible par le bras manipulateur, c'est-à-dire que la situation appartient à l'espace de travail 1.5.

Alors (dans la majorité des cas) :

— si $n < m$, il n'existe pas de solution au MGI.

— si $n = m$, il existe un nombre fini de solutions en dehors de certaines configurations, appelées configurations singulières.

— si $n > m$, il existe une infinité de solutions.

On sait que dans les cas où $n < 6$ les bras manipulateurs sont tous solubles, c'est-à-dire qu'il existe une solution connue au MGI. C'est aussi le cas de la plupart des structures à six liaisons, notamment celles possédant un poignet sphérique (trois dernières liaisons rotoïdes concourantes). Enfin, seul le calcul du MGI permet de connaitre le nombre de solutions [6].

Il n'existe pas de méthode analytique systématique pour calculer le MGI. Le mieux est $la\, methode\, de\, paul$ qui consiste juste à reprendre les équations du MGD, préalablement calculé et de mener le calcul à l'envers. Le calcul se fait alors au cas par cas. Il est généralement aisé pour un bras manipulateur à moins de six axes.

1.4.2 Modèle Cinématique

1.4.2.1 Modèle Cinématique Direct

Le modèle cinématique se base sur le calcul variationnel, il permet de décrire les variations élémentaires des coordonnées opérationnelles en fonction des variations élémentaires des coordonnées articulaires.

Le modèle cinématique direct est décrit par l'équation :

$$\dot{X} = J(q)\dot{q} \tag{1.1}$$

où $J(q)$ est la matrice jacobienne de dimension $(m \times n)$ du mécanisme, égale à $\frac{\partial X}{\partial q}$ et fonction de la configuration articulaire q. La même matrice jacobienne intervient dans le calcul du modèle différentiel direct qui donne les variations élémentaires dX des coordonnées opérationnelles en fonction des variations élémentaires des coordonnées articulaires dq, soit :

$$dX = J(q)dq$$

Le jacobien est une matrice qui peut être considérée comme la version vectorielle de la dérivée ordinaire d'une fonction scalaire. Le jacobien est l'une des quantités les plus importantes dans l'analyse et le contrôle du mouvement du robot. Il en surgit de manière virtuelle tous les aspects de la manipulation robotique : dans la planification et l'exécution de trajectoires lisses, dans la détermination de configurations singulières, dans l'exécution d'un mouvement anthropomorphique coordonné, dans la dérivation des équations dynamiques du mouvement, et dans la transformation de forces et couples de l'effecteur terminal aux articulations du manipulateur.

Singularité

Le rang d'une matrice n'est pas nécessairement constant. En effet, le rang de la matrice jacobienne du manipulateur dépendra de la configuration q. Les configurations pour lesquelles le $rang(J)$ est inférieur à sa valeur maximale sont appelés singularités ou configurations singulières. Identifier les singularités des manipulateurs est important pour plusieurs raisons :

1. Les singularités représentent des configurations à partir desquelles certaines directions de mouvement peuvent être inaccessible
2. Aux singularités, la vitesse de l'effecteur final limitée peut correspondre à des vitesses articulaires limitées.
3. Les singularités correspondent à des points de l'espace de travail du manipulateur qui peuvent être inaccessibles sous de petites perturbations des paramètres de liaison, telles que la longueur, le décalage, etc... [33].

Redondance

Un manipulateur est qualifié de redondant cinématiquement lorsqu'il possède un nombre de DLL supérieur au nombre de variables nécessaires pour décrire une tâche donnée. Un manipulateur est intrinsèquement redondant lorsque la dimension de l'espace opérationnel est plus petite que la dimension de l'espace articulaire $(m < n)$.

1.4.2.2 Modèle Cinématique Inverse

L'objectif du modèle cinématique inverse est de calculer, à partir d'une configuration q donnée, les vitesses articulaires $\dot{q}$ qui assurent au repère terminal une vitesse opérationnelle $\dot{X}$ imposée. Cette définition est analogue à celle du modèle différentiel inverse :

$$dq = J^{-1}(q)dX \tag{1.2}$$

Ce dernier permet de déterminer la différentielle articulaire dq correspondant à une différentielle des coordonnées opérationnelles dX spécifiée. Pour obtenir le modèle cinématique inverse, on inverse le modèle cinématique direct en résolvant un système d'équations linéaires.

Il est peut-être un peu surprenant que le modèle cinématique inverse soit conceptuellement plus simple que la position inverse. Lorsque la jacobienne est carrée (c'est-à-dire $J \in R^{n \times n}$) et non singulière, ce problème peut être résolu en inversant simplement la matrice jacobienne pour donner.

$$\dot{q} = J^{-1}(q)\dot{X}$$

Si $n < 6$, $\dot{X}$ ne peuvent pas être obtenues (le robot a une singularité).

Pour éviter les singularités, une solution consiste à augmenter le nombre de degrés de liberté du mécanisme. Le robot devient redondant et avec un critère approprié, il est possible de déterminer un mouvement hors singularité. Il existe cependant des singularités inévitables qui doivent être prises en compte par le concepteur de la commande [17].

Si $n > 6$, alors nous disons que le robot est redondant.

Pour un tel mécanisme, la matrice J est de dimension $(m \times n)$ avec $n > m$, en supposant que les coordonnées articulaires et opérationnelles utilisées soient indépendantes $(n = N, m = M)$. Plusieurs méthodes de résolution du système 1.1 sont envisageables. Une solution classique consiste à utiliser une pseudo-inverse avec un terme d'optimisation [17].

Il est courant d'utiliser la pseudo-inverse J^{+}de la matrice J :

$$\dot{q} = J^{+}(q)\dot{X}$$

tel que :

$$J^{+} = J^{T}(JJ^{T})^{-1}$$

1.4.3 Modele dynamique

La dynamique du robot s'intéresse à la relation entre les forces agissant sur un mécanisme de robot et les accélérations qu'elles produisent. En règle générale, le mécanisme du robot est modélisé comme un système à corps rigide, et la dynamique du robot est l'application de la dynamique du corps rigide aux robots. Les deux principaux problèmes de la dynamique des robots sont :

Dynamique directe : prend les forces, élabore les accélérations.

Dynamique inverse : prend les accélérations, élabore les forces

La modélisation dynamique consiste à établir les relations entre les efforts des actionneurs et les mouvements qui en découlent, autrement dit, à expliciter les équations différentielles du second ordre que sont les équations du mouvement [17].

l'espace d'etat d'un bras manipulateur

l'état du manipulateur est un ensemble d'equations différentielles qui associer une description de la dynamique et de manipulateur. Dans le cas d'un bras manipulateur, la dynamique est Newtonienne et peut être précisée en généralisant l'équation familière $F = ma$. Ainsi, un état du manipulateur peut être spécifié en donnant les valeurs des variables articulaires q et des vitesses articulaires $\dot{q}$ (l'accélération est liée à la dérivée des vitesses articulaires). La dimension de l'espace d'état est donc $2n$ si le système a n DLL [34].

On représente le modèle dynamique par une relation de la forme :

$$\Gamma = f(q, \dot{q}, \ddot{q}, f_e) \tag{1.3}$$

avec :

- Γ : vecteur des couples/forces des actionneurs, selon que l'articulation est rotoïde ou prismatique. Dans la suite, on écrira tout simplement couples
- q : vecteur des positions articulaires
- q : vecteur des vitesses articulaires
- q : vecteur des accélérations articulaires
- f_e : vecteur représentant l'effort extérieur (forces et moments) qu'exerce le robot sur l'environnement.

L'équation 1.3 est un modèle dynamique inverse car il définit l'entrée du système en fonction des variables de sortie. Il est souvent appelé le modèle dynamique [13, 17].

Le modèle dynamique direct est celui qui exprime les accélérations articulaires en fonction des positions, vitesses et couples des articulations. Il est alors représenté par la relation :

$$\ddot{q} = g(q, \dot{q}, \Gamma, f_e)$$

Plusieurs approches ont été proposées pour modéliser la dynamique des robots, les formalismes les plus souvent utilisés sont: le formalisme de Lagrange-Euler et le formalisme de Newton-Euler [13].

1.4.3.1 Forme générale du model dynamique

L'énergie cinétique du système est une fonction quadratique des vitesses articulaires :

$$E = \frac{1}{2}\dot{q}^T A \dot{q} \tag{1.4}$$

L'énergie potentielle étant fonction des variables articulaires q, le couple Γ peut se mettre, à partir l'équation 1.4 et le formalisme de Lagrange-Euler [13], sous la forme :

$$\Gamma = A(q)\ddot{q} + C(q, \dot{q})\dot{q} + Q(q)$$

tel que:
— A : est la matrice $(n \times n)$ de l'énergie cinétique
— $C(q, \dot{q})\dot{q}$: vecteur de dimension $(n \times 1)$ représentant les couples/forces.
— $Q = [Q_1 \dots Q_n]^T$: vecteur des couples/forces de gravité.
Les éléments de A, C et Q sont fonction des paramètres géométriques et inertiels du mécanisme. Les équations dynamiques d'un système mécanique articulé forment donc un système de n équations différentielles du second ordre, couplées et non linéaires.

1.5 Robot kuka LBR iiwa

1.5.1 Introduction

Le LBR iiwa est le premier robot sensitif, et donc apte à la collaboration homme-robot, fabriqué en série. LBR signifie "Leichtbauroboter" (robot léger), iiwa signifie "intelligent industrial work assistant". Une nouvelle ère de la robotique industrielle sensitive commence; la base pour des processus de production nouveaux et sûrs pour l'avenir. Pour la première fois, l'homme et le robot peuvent exécuter des tâches sensibles en étroite collaboration. De nouveaux horizons de travail s'ouvrent et la voie est libre pour plus de rentabilité et une efficacité des plus élevées. Le LBR iiwa, sensitif et collaboratif, existe en deux versions avec des charges de 7 et 14 kilogrammes.

1.5.2 Caractéristiques

Grâce à ses capteurs de couple conjoints, le LBR iiwa peut détecter immédiatement le contact et réduit instantanément son niveau de force et de vitesse. Les autres caractéristiques sont indiquées dans le tableau 1.1.

Réactions rapides

À l'aide de ses capteurs de couples, le LBR iiwa détecte immédiatement les contacts et réduit aussitôt la force et la vitesse. Il manipule les pièces fragiles sans risques de pincement et de cisaillement trop prononcés grâce à la régulation de position.

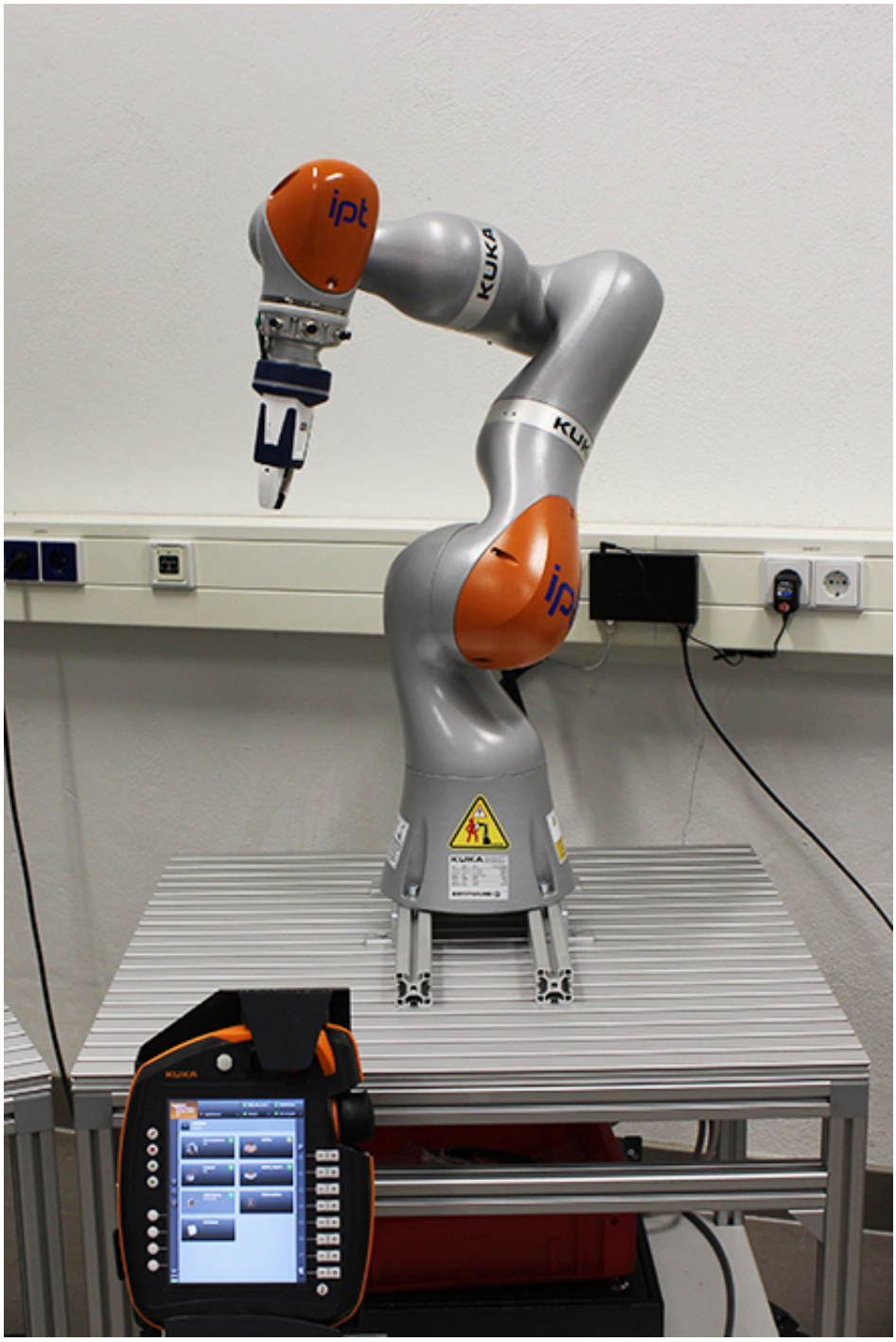


Figure 1.8 – Kuka LBR iiwa 7 R800

Catactéristique	LBR iiwa 7 R800	LBR iiwa 7 R800 CR	LBR iiwa 14 R820
Charge	7 kg	7 kg	14 kg
Portée max.	800 mm	800 mm	820 mm
Modèle	Standard	Standard	Standard
Version environnement	Standard	Cleanroom	Standard
Position(s) de montage	Mur ,Plafond, Sol	Sol	Mur ,Plafond, Sol
Mode de protection	IP 54	IP 54	IP 54

TABLE 1.1 – Catactéristiques des modèles des robots kuka LBR iiwa.

Capacité d'apprentissage

Choisissez l'un des trois modes et programmez le LBR iiwa par simulation. Montrez-lui la position souhaitée, il se souvient des coordonnées du point sur la trajectoire. Pour les pauses, vous l'interrompez et le guidez tout simplement en le touchant.

Sensitif

En tant que poids léger avec régulation extrêmement performante, le LBR iiwa détecte des contours très rapidement sous contrôle d'efforts. Il détecte la position de montage correcte et monte les pièces rapidement avec une précision de couple de ±2% du moment maximum. Le LBR iiwa trouve des petites pièces fines sans votre aide en moins de temps qu'il faut pour le dire.

Autonome

Le contrôleur KUKA Sunrise Cabinet du LBR iiwa simplifie la mise en service rapide, même pour des tâches complexes. Faites du LBR iiwa la troisième main de l'homme et laissez-le se charger des tâches peu ergonomiques et monotones de façon fiable et autonome.

1.6 Conclusion:

Dans ce chapitre, nous avons donné un aperçu général sur la robotique : les différents types des robots, leurs classifications ainsi que leurs espaces de travail. Nous avons aussi expliqué comment modiliser un bras manipulateur et trouver le modèle géométrique, cinématique et dynamique,

Le modèle géométrique nous sert a trouver la position finale qui'il faudra atteindre. D'autre part, le modèle cinématique nous sert à trouver les vitesses de chaque articulation.

Le modèle dynamique nous sert à trouver la commande pour atteindre notre objectif à partir du calclul d'énergie, de force et de puissance.

Chapitre 2

L'apprentissage automatique pour la vision par ordinateur

2.1 Introduction

Bien que la manipulation d'objets soit relativement facile pour les humains, saisir des objets arbitraires de manière fiable reste un défi ouvert pour les robots. Les récents progrès de la robotique et des systèmes automatisés ont conduit à l'expansion des capacités autonomes et à l'utilisation de machines plus intelligentes dans des applications toujours plus variées [10]. La capacité d'adaptation à des environnements changeants est une compétence nécessaire pour les robots. L'apprentissage automatique joue un rôle clé dans la création de telles solutions robotiques à usage général. En raison des récents résultats positifs des méthodes d'apprentissage en profondeur dans les applications de vision par ordinateur et de robotique, de nombreux chercheurs en robotique ont commencé à explorer l'application des méthodes d'apprentissage en profondeur dans leurs recherches.

Ce chapitre donne une introduction de la théorie de l'apprentissage automatique plus précisément l'apprentissage profond et ses utilisations dans les domaines de vision par ordinateur et la robotique.

2.2 La vision par ordinateur

La vision par ordinateur est un domaine scientifique interdisciplinaire qui traite de la façon dont les ordinateurs peuvent acquérir une compréhension de haut niveau à partir d'images ou de vidéos numériques. Du point de vue de l'ingénierie, il cherche à comprendre et à automatiser les tâches que le système visuel humain peut les faire.

Les tâches de vision par ordinateur comprennent des méthodes d'acquisition, de traitement, d'analyse et de compréhension d'images numériques, et d'extraction de données de grande dimension du monde réel afin de produire des informations numériques ou symboliques, La discipline scientifique de la vision par ordinateur s'intéresse à la théorie des systèmes artificiels qui extraient l'information des images. Les données d'image peuvent prendre de nombreuses formes. La discipline technologique de la vision par ordinateur cherche à appliquer ses théories et ses modèles à la construction de systèmes de vision par ordinateur

2.2.1 Forme d'image

L'image est définie comme étant une fonction $f(x, y)$ à deux dimensions (Fig. 2.1), où x et y sont les coordonnées spatiales, et l'amplitude à tous points $f(x, y)$ correspondant à l'intensité ou au niveau de gris. Lorsque les points (x, y) et l'amplitude sont discrétisés, on parle d'image numérique ou digitale.

2.2.1.1 La forme RGB

Dans le modèle RGB, une image se compose de trois plans d'image indépendants, un dans chacune des couleurs primaires : rouge, vert et bleu. La spécification d'une couleur particulière consiste à spécifier la

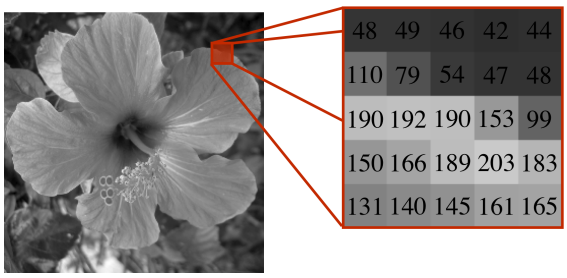


FIGURE 2.1 – une sous-image de taille $5x5$

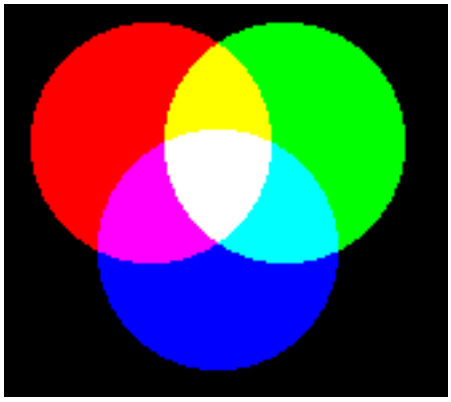


FIGURE 2.2 – La figure montre le mélange additif de primaires rouge, vert et bleu pour former les trois couleurs secondaires jaune (rouge + vert), cyan (bleu + vert) et magenta (rouge + bleu) et blanc ((rouge + vert) + bleu).

quantité de chacun des principaux composants présents. Il s'agit d'un modèle additif, c'est-à-dire que les couleurs présentes dans la lumière s'ajoutent pour former de nouvelles couleurs, et convient par exemple pour le mélange de lumière colorée. L'image de la figure 2.2 montre le mélange additif de couleurs primaires pour former certains couleurs secondaires.

2.2.1.2 La frome HSV

Contrairement à RGB, le modèle HSV a des valeurs pour la teinte de couleur, le degré de saturation et l'intensité. Ce type de représentation a été développé pour être plus intuitif que RVB, bien qu'il souffre de certaines difficultés de calcul d'image, comme une discontinuité dans la valeur de teinte à l'angle 360° (Fig. 2.3). Cependant, cela peut présenter des avantages pour certains types de calculs d'image où il est utile d'avoir la teinte, la saturation et l'intensité séparément.

2.2.2 Traitement d'images

Le traitement d'image est principalement lié à l'utilisation et à l'application de fonctions mathématiques et de transformations sur des images, indépendamment de toute inférence intelligente effectuée sur l'image elle-même. Cela signifie simplement qu'un algorithme effectue certaines transformations sur l'image telles que le lissage, la netteté, le contraste, l'étirement sur l'image.

2.2.2.1 Blurring & Smoothing

Dans le traitement d'image, le flou "Blurring" (également appelé lissage "Smoothing") est le résultat du flou d'une image par une fonction Gaussienne 2.1. C'est un effet largement utilisé dans les logiciels graphiques, généralement pour réduire le bruit de l'image et réduire les détails. L'effet visuel de cette technique de flou

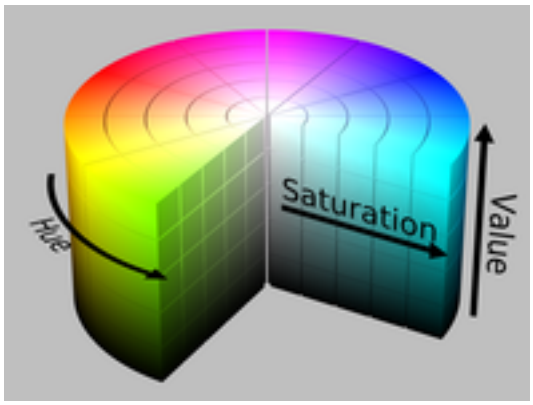


FIGURE 2.3 – La représentation HSV (Hue,Saturation,Value)

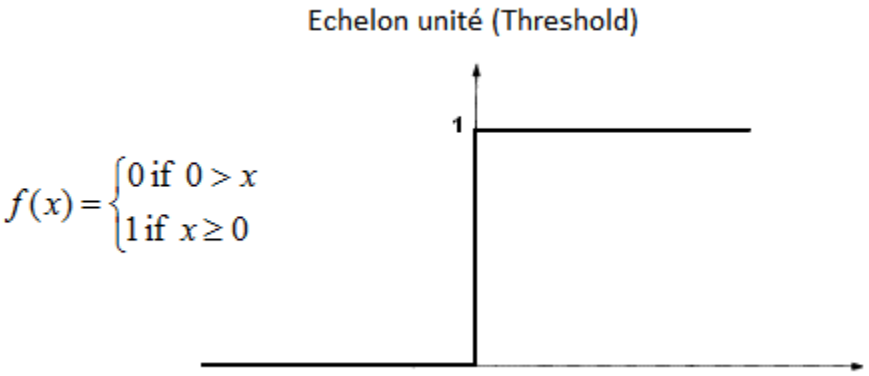


FIGURE 2.4 – La fonction Echelon unité qui représente la seuillage "thresholding"

est un flou lisse ressemblant à celui de la visualisation de l'image à travers un écran translucide. Le lissage Gaussien est également utilisé comme étape de prétraitement dans les algorithmes de vision par ordinateur afin d'améliorer les structures d'image à différentes échelles [32].

La formule d'une fonction gaussienne à une dimension est :

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{2.1}$$

2.2.2.2 Thresholding

Le seuillage "Thresholding" est une méthode essentielle dans le traitement d'image [32]. Il s'agit d'une opération non linéaire qui convertit une image en échelle de gris en une image binaire où les deux niveaux sont attribués à des pixels inférieurs ou supérieurs à la valeur de seuil spécifiée. En d'autres termes, si la valeur de pixel est supérieure à une valeur de seuil, une valeur lui est attribuée (peut être blanche), sinon une autre valeur (peut être noire), c'est-à-dire "premier plan" et "arrière-plan" sont est attribuée.

Bien que le plus souvent appliqué aux images en niveaux de gris, il peut également être appliqué aux images en couleur. La forme générale d'un seuillage est représentée dans la figure 2.4.

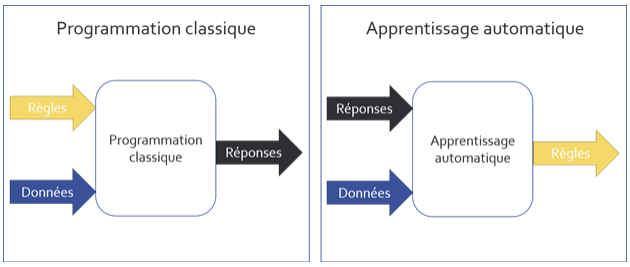


FIGURE 2.5 – La différence entre la programation classique et l'apprentissage automatique [11].

2.2.2.3 Histogramme

Un histogramme d'image est un type d'histogramme qui agit comme une représentation graphique de la distribution tonale dans une image numérique. Il trace le nombre de pixels pour chaque valeur totale. En regardant l'histogramme pour une image spécifique, un spectateur pourra juger la distribution tonale entière en un coup d'œil [32].

2.2.2.4 OpenCV

OpenCV (Open Source Computer Vision Library) est une bibliothèque de logiciels open source de vision par ordinateur et d'apprentissage automatique. OpenCV a été conçue pour fournir une infrastructure commune pour les applications de vision par ordinateur et pour accélérer l'utilisation de la perception de la machine dans les produits commerciaux. La bibliothèque contient plus de 2500 algorithmes optimisés, qui incluent un ensemble complet d'algorithmes classiques de vision par ordinateur et d'apprentissage automatique [32].

2.3 Apprentissage automatique

l'apprentissage automatique ou bien « Machine Learning » est un domaine de l'informatique, il diffère des approches informatiques traditionnelles. Dans l'informatique traditionnelle, les algorithmes sont des ensembles d'instructions programmées explicitement utilisées par les ordinateurs pour calculer ou résoudre des problèmes. Les algorithmes d'apprentissage automatique permettent à la place aux ordinateurs de s'entraîner sur les entrées de données et d'utiliser l'analyse statistique afin de produire des valeurs qui se situent dans une plage spécifique.

L'objectif de l'apprentissage automatique est généralement de comprendre la structure des données et de les intégrer dans des modèles qui peuvent être compris et utilisés par les gens.

Dans l'apprentissage supervisé, l'ordinateur est équipé d'entrées d'échantillon qui sont étiquetées avec leurs sorties souhaitées.. Dans l'apprentissage non supervisé, les données ne sont pas étiquetées, l'algorithme d'apprentissage est laissé à trouver des points communs parmi ses données d'entrée (Fig. 2.5).

2.3.1 Apprentissage profond

Bien sûr, il peut être très difficile d'extraire de telles caractéristiques abstraites de haut niveau à partir de données brutes, c'est pourquoi l'apprentissage en profondeur ou bien « Deep Learning » vient résoudre ces problèmes, ce qui permet à l'ordinateur de construire des concepts complexes à partir de concepts plus simples.

2.3.1.1 Perceptron :

un perceptron est une représentation mathématique et informatique d'un neurone biologique , Le neurone formel est conçu comme un automatisme soutenu d'une fonction de transfert qui transforme ses entrées en sortie selon des règles précises. Par exemple, un neurone somme ses entrées, compare la somme résultante à une valeur seuil, et répond en émettant un signal si cette somme est supérieure ou égale à ce seuil (modèle

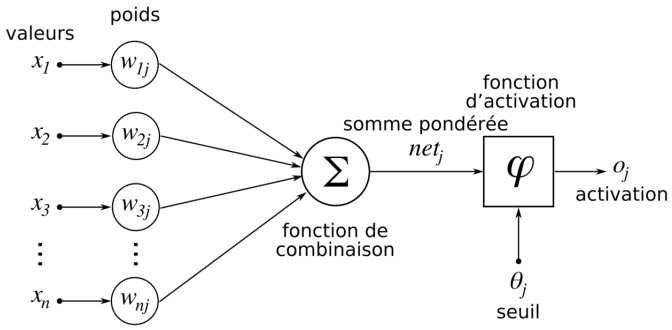


FIGURE 2.6 – Structure d'un neurone artificiel.[3]

ultra-simplifié du fonctionnement d'un neurone biologique) (Fig. 2.6). La recherche sur les réseaux de neurones est principalement guidée par les développements en ingénierie et en mathématiques plutôt qu'en biologie [7].

Le neurone est entraîné en sélectionnant soigneusement les poids pour produire une sortie souhaitée pour chaque entrée.

En considérant un signal d'entrée $x = [x_1, ..., x_n]$, le neurone artificiel renvoie la valeur d'activation (sortie) y :

$$y = f(b + w_i x_i)$$

Dans cette formulation, les w_i sont communément appelés les poids et b est appelé le biais. La fonction f est nommée fonction d'activation ou plus rarement fonction de transfert.

2.3.1.2 Fonction d'activation

La fonction d'activation détermine la sortie finale de chaque neurone. Le choix de la fonction d'activation est un élément essentiel de la conception du réseau neuronal. L'importance des fonctions d'activation non linéaires devient significative lorsque l'on passe du perceptron monocouche aux architectures multicouches.

Les fonctions d'activation les plus utilisées par les developeurs des réseaux de neurones sont les fonctions *ReLu*, *sigmoïd* et $hyperbolic - tangent$ (Fig. 2.7).

2.3.1.3 Réseaux de neurones Artificiels

Un Réseau de neurones Artificiel est un réseau de neurones multicouche qui comprend généralement trois types de couches : une couche d'entrée, une ou plusieurs couches masquées et une couche de sortie (Fig. 2.8). La couche d'entrée transmet généralement les données sans les modifier. La plupart des calculs se produisent dans les couches cachées. La couche de sortie convertit les activations de couche cachées en une sortie, telle qu'une classification.

L'exemple essentiel d'un modèle d'apprentissage en profondeur est le réseau profond à action directe « feedforward deep network », ou perceptron multicouche « multilayer perceptron (MLP) ». Un perceptron multicouche n'est qu'une fonction mathématique mappant un ensemble de valeurs d'entrée à des valeurs de sortie. La fonction est formée en composant de nombreuses fonctions plus simples. Nous pouvons penser que chaque application d'une fonction mathématique différente fournit une nouvelle représentation de l'entrée [7].

2.3.1.4 Fonction de perte

Les réseaux de neurones apprennent en minimisant l'erreur avec une fonction de perte, qui est la méthode d'évaluation de la façon dont l'algorithme spécifique modélise les données. Si les prévisions dépassent les résultats réels, la fonction de perte produira une très grande erreur.

Il n'y a pas de fonction de perte unique pour les algorithmes dans le deep learning. Le choix d'une fonction de perte pour un problème spécifique dépend de différents facteurs, tels que le type d'algorithme d'apprentissage choisi, le type de la sortie prédie.

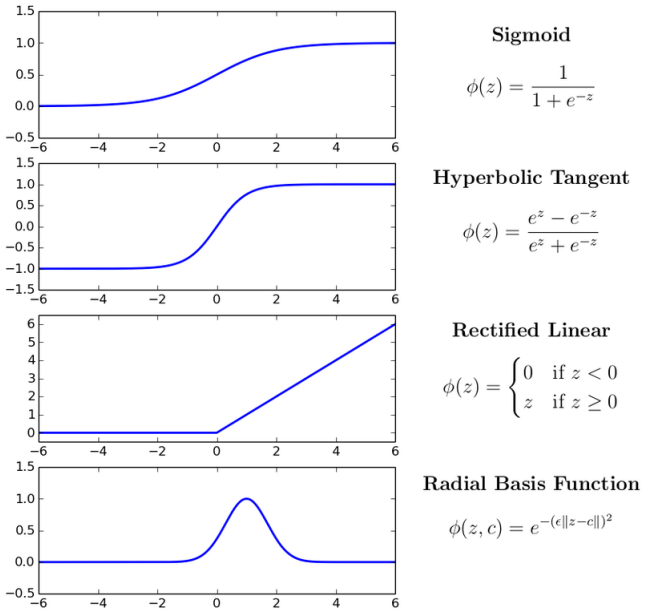


FIGURE 2.7 – Fonctions d'activation de réseau neuronal communes.[**?**]

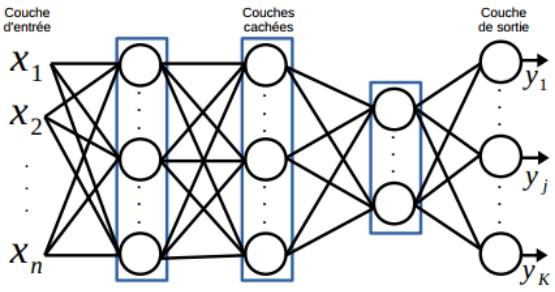


FIGURE 2.8 – Un exemple de perceptron multicouches constitué de trois couches cachées. Chaque cercle représente un neurone formel. Les neurones dans le même rectangle font partie de la même couche cachée.

En gros, les fonctions de perte peuvent être classées en deux grandes catégories selon le type de tâche d'apprentissage que nous traitons, les pertes de régression et les pertes de classification. . Dans la classification, nous essayons de prédire la sortie d'un ensemble de valeurs catégorielles finies, d'autre part, la régression traite de la prédiction d'une valeur continue, par exemple le prix de maison.

Il existe plusieurs fonctions de perte (ou « Loss fucntions » en anglais) utilisables pour le deep learning. Ces fonctions sont dépendantes de la tâche que le réseau doit effectuer (classification, régression...), par exemple :

— la moindre carée (mean squred error) pour la régression :

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}$$

— l'entropie croisée (Cross Entropy Loss) pour la classification :

$$CEL = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)$$

2.3.1.5 Formation d'un réseau de neurones

Un réseau de neurones est formé en sélectionnant les poids (weights) de tous les neurones afin que le réseau apprenne à approximer les sorties cibles à partir d'entrées connues. Il est difficile de résoudre analytiquement les poids des neurones d'un réseau multicouche. L'algorithme de rétropropagation *backpropagation* fournit une solution simple et efficace pour résoudre les poids de manière itérative [7].

Backpropagation : nous avons supposé qu'une fonction étant différentiable, nous pouvons calculer explicitement sa dérivée. En pratique, une fonction de réseau neuronal consiste en de nombreuses opérations de matrices enchaînées, chacune ayant une dérivée simple et connue. Par exemple, il s'agit d'un réseau f composé de trois opérations matricielles , a, b et c, avec les matrices de poids $W1$, $W2$ et $W3$:

$$f(W1, W2, W3) = a(W1, b(W2, c(W3))))$$

Le calcul nous dit qu'une telle chaîne de fonctions peut être dérivée en utilisant l'identité suivante, appelée la règle de chaîne : $f(g(x)) = f'(g(x)) * g'(x)$ L'application de la règle de chaîne au calcul des valeurs de gradient d'un réseau neuronal donne naissance à un algorithme appelé *Backpropagation* ou rétropropagation [11].

Dans la première phase de l'algorithme, un vecteur d'entrée est propagé vers l'avant d'un réseau neuronal. La sortie du réseau est comparée à la sortie souhaitée (qui devrait être connue pour les exemples de formation) en utilisant une fonction de perte. la valeur d'erreur de la couche de sortie est simplement la différence entre la sortie actuelle $\hat{y}$ et la sortie souhaitée y. [7].

Sur-ajustement et sous-ajustement **:** Même lorsque nous travaillons sur un projet d'apprentissage automatique, nous sommes souvent confrontés à des situations où nous rencontrons des performances inattendues ou des différences de taux d'erreur entre l'ensemble de formation et l'ensemble de test, pour cela, nous avons le sur-ajustement **Overfitting** et le sous-ajustement **Underfitting** , qui sont peut-être responsables de cette mauvaise performance des algorithmes d'apprentissage automatique.

Votre modèle constitue un sous-ajustement des données de formation lorsque le modèle donne des résultats médiocres sur les données de formation. Cela est dû au fait que le modèle n'est pas en mesure de saisir la relation entre les exemples en entrée et les valeurs cibles. Si modèle constitue un sur-ajustement sur les données de formation lorsque le modèle offre de bons résultats sur les données de formation mais des résultats médiocres sur les données d'évaluation. Cela est dû au fait que le modèle mémorise les données qu'il a vues et n'est pas en mesure de généraliser aux exemples nouveaux.

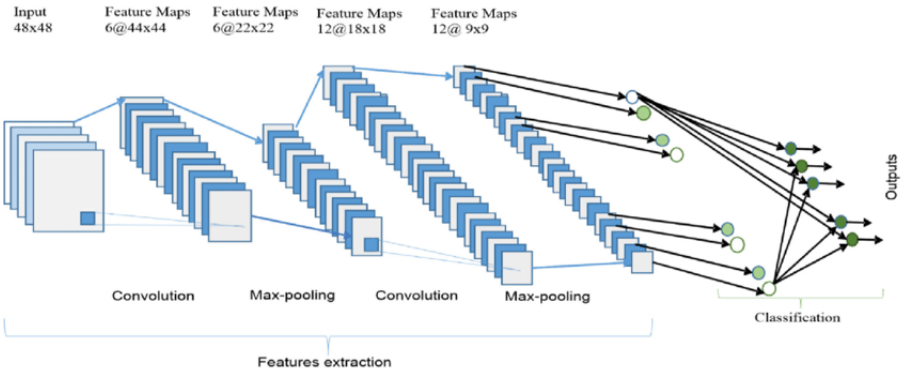


FIGURE 2.9 – L'architecture globale du réseau neuronal convolutif (CNN) comprend une couche d'entrée, plusieurs couches de convolution alternée et de regroupement maximal, une couche entièrement connectée et une couche de classification [5].

Pour´eviter le sur-apprentissage, nous utilisons le dropout,c'est une methode qui consiste à désactiver certains neurones à chaque étape de la descente de gradient, et pour le sous-ajustement nous devons augmenter la taille de données pour la formation et le test [2].

2.3.2 Réseaux de neurones convolutifs (CNN) :

Le premier réseau de neurones convolutif (CNN) a été introduit à la fin des années 80 par [21] C'est le premier réseau de neurones pour la reconnaissance d'images. Ce réseau permettait la reconnaissance de chiffres manuscrits. L'idée est de passer l'image dans une succession de filtres convolutifs apportant une description réduite et pertinente de l'image. Ces caractéristiques sont, par la suite, envoyées à un perceptron multicouches composé de couches cachées et d'une couche de sortie complètement connectées permettant la classification du chiffre présent dans l'image. Les filtres de convolution et les couches complètement connectées sont appris simultanément.

Les CNN sont un type particulier de réseaux de neurones applicables facilement à des images pour capter spatialement de l'information. De plus, de par leur structure convolutive, ils permettent de prendre en entrée des données de grande dimension ce qui est une limite du perceptron multicouches. Une image à trois canaux (RGB) de taille 224 × 224 pixels représente un vecteur d'entrée de taille 150 528 pour un perceptron multicouches. Cela implique 150 528 poids à apprendre pour chaque neurone de la couche cachée connectée aux entrées, ce qui est compliqué à apprendre. Les CNN peuvent être vus comme un assemblage de modules en série permettant l'extraction de caractéristiques de manière hiérarchique à partir des pixels d'une image (Fig. 2.9).

2.3.2.1 Différents modules d'un réseau de neurones convolutif

Convolution 2D

Une couche de convolution est un composant fondamental de l'architecture CNN qui effectue l'extraction d'entités, qui transmettent généralement la propagation sur un ensemble de données d'apprentissage, sous forme d'images, de voix..

La convolution est un type d'opération linéaire spécialisé utilisé pour l'extraction des caractéristique, avec un petit tableau de nombres, appelé noyau « *kernel* », appliqué à travers l'entrée, et un tableau de nombres, appelé tenseur « *tensor* ». Un produit entre chaque élément du noyau et le tenseur d'entrée est calculé à chaque emplacement du tenseur et additionné pour obtenir la valeur de sortie à la position correspondante du tenseur de sortie (Fig. 2.10), appelé une carte des caractérstiques, cette procédure est répétée en appliquant plusieurs noyaux pour former un nombre arbitraire de cartes d'entités, qui représentent différentes caractéristiques des tenseurs d'entrée ; différents noyaux peuvent donc être considérés comme différents extracteurs de caractéristiques . Deux hyper paramètres clés qui définissent l'opération de convolution sont la

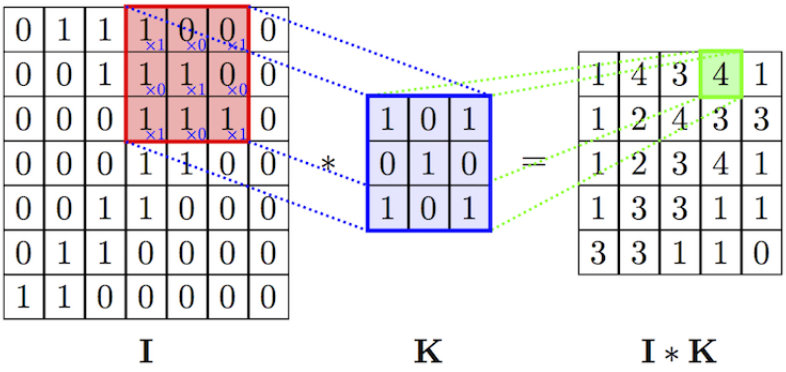


FIGURE 2.10 – Illustration de la convolution. Étant donné une image entrée I, un filtre de convolution (ou noyau de convolution) K . Le résultat correspond à la somme de la multiplication des éléments du noyau par ceux de l'entrée : dans cet exemple $1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 1 = 4$. Dans le cadre des CNN, la sortie d'une convolution est appelée carte de caractéristiques. Le nombre de cartes de caractéristiques dépend du nombre de filtres appliqués sur l'entrée [37].

taille et le nombre de noyaux. Le premier est généralement 3×3, mais parfois 5×5 ou 7×7. Le second est arbitraire et détermine la profondeur des cartes d'entités en sortie [37].

L'opération de convolution discrète entre une image f et une matrice de filtre g est définie comme :

$$h[x, y] = f[x, y] * g[x, y] = \sum_n \sum_m f[n, m]g[x - n, y - m]$$

En effet, le produit scalaire du filtre g et une sous-image de f (de mêmes dimensions que g) centrée sur les coordonnées x, y produit la valeur de pixel de h aux coordonnées x, y. La taille du champ récepteur est ajustée par la taille de la matrice de filtre. L'alignement successif du filtre sur chaque sous-image de f produit la matrice de pixels de sortie h. Dans le cas des réseaux de neurones, la matrice de sortie est également appelée carte de caractéristiques "feature map" [4].

Pooling

Pour rendre le réseau plus facile à gérer pour la classification, il est utile de réduire la taille de la carte d'activation dans l'extrémité profonde du réseau. Il existe deux façons de réduire la taille du volume de données. Une façon consiste à inclure une couche de pooling après une couche convolutionnelle.

L'idée principale de Pooling est le sous-échantillonnage afin de réduire la complexité pour d'autres couches. Dans le domaine du traitement d'image, cela peut être considéré comme similaire à la réduction de la résolution. La mise en commun n'affecte pas le nombre de filtres. Le regroupement maximal est l'un des types de méthodes de regroupement les plus courants (Fig. 2.11). Il partitionne l'image en rectangles de sous-région et ne renvoie que la valeur maximale de l'intérieur de cette sous-région. L'une des tailles les plus couramment utilisées dans la mise en commun maximale est 2×2 [4].

Fully-connected layer

La couche entièrement connectée est similaire à la façon dont les neurones sont disposés dans un réseau neuronal traditionnel. Chacun des nœuds des dernières trames de la couche de Pooling est connecté en tant que vecteur à la première couche de la couche entièrement connectée "Fully-connected layer" [37].

L'inconvénient majeur d'une couche entièrement connectée est qu'elle inclut de nombreux paramètres qui nécessitent des calculs complexes dans les exemples de formation. Par conséquent, nous essayons d'éliminer le nombre de nœuds et de connexions. Les nœuds supprimés et la connexion peuvent être satisfaits en utilisant la technique de dropout [4].

Cette partie du CNN est importante car elle est responsable de la sortie, la sortie de cette couche est spécifique et dépend du type de sortie y.

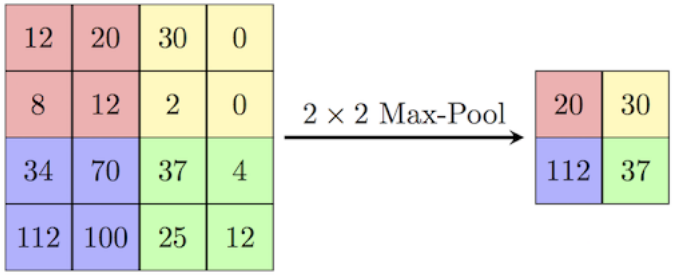


FIGURE 2.11 – Illustration du Max Pooling. Dans cet exemple, le noyau de pooling est de taille 2 × 2 et est appliqué tous les deux pixels (stride = 2). Le maximum des quatre éléments sur une fenêtre de l'entrée est gardé

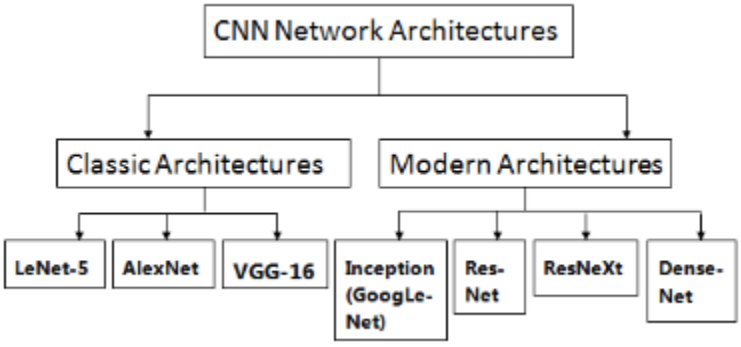


FIGURE 2.12 – Les architecture CNN [16].

2.3.3 Les architectures neuronales classiques

Les réseaux de neurones convolutifs (CNN) ont été introduits il y a 20 ans, pour la détection et la reconnaissance d'objets visuels. L'utilisation des CNN a permis une extraction et une classification robustes des caractéristiques. De nombreuses contributions à la structure CNN ont récemment été rapportées pour construire des DCNN (Fig.2.12).

AlexNet AlexNet contient 5 couches convolutives ainsi que 2 couches entièrement connectées pour les fonctionnalités d'apprentissage , il a un regroupement maximal après les première, deuxième et cinquième couches convolutives. Au total, il possède 650 000 neurones, 60 millions de paramètres et 630 millions de connexions. L'AlexNet a été le premier à montrer que l'apprentissage en profondeur était efficace dans les tâches de vision par ordinateur [4].

VGG Une nouvelle architecture très profonde [16] appelée réseau VGG a été introduite. Bien qu'il soit plus profond comme le montre la figure 2.13, il est pourtant plus simple que les architectures précédentes. La simplicité a été obtenue en proposant une nouvelle stratégie pour concevoir des réseaux très profonds basés sur des blocs de construction de taille égale. L'augmentation de la profondeur est compensée en appliquant uniquement de petits filtres de convolution de taille 3 × 3. Les images d'entrée de taille 224 × 224 passent à travers une séquence de couches convolutives. Les détails des couches sont présentés à la figure 2.13. La fonction non linéaire ReLU est utilisée. La normalisation de la réponse n'a pas contribué et a consommé plus de mémoire et de temps.

ResNet Ce CNN permet l'apprentissage de réseaux très profonds (plus de 150 couches). La difficulté à apprendre des réseaux aussi profonds est notamment liée à la rétropropagation du gradient. Plus le ré- seau

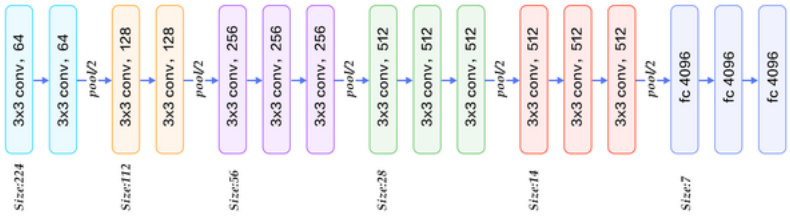


FIGURE 2.13 – L'architecture du VGG.

est profond, plus le gradient est faible pour la mise à jour des poids des couches de plus bas niveau (les premières couches). L'idée développée dans ResNet est l'utilisation de connexions résiduelles permettant une meilleure optimisation des réseaux très profonds. Une connexion résiduelle permet de passer l'entrée dans deux filtres de convolution mais également de passer directement cette entrée aux couches suivantes.. Avec cette architecture, les auteurs démontrent l'intérêt d'apprendre des réseaux très profonds de par leurs performances et proposent une manière pour les apprendre efficacement [16].

2.3.4 Détection d'objets

Pour acquérir une compréhension complète de l'image, nous ne devons pas nous concentrer uniquement sur la classification des différentes images, mais également essayer d'estimer avec précision les concepts et les emplacements des objets contenus dans chaque image. Cette tâche est appelée détection d'objet , En tant que l'un des problèmes fondamentaux de la vision informatique, la détection d'objets est capable de fournir des informations précieuses pour la compréhension sémantique des images et des vidéos, Cependant, en raison des grandes variations des points de vue, des poses, des occlusions et des conditions d'éclairage, il est difficile de réaliser parfaitement la détection d'objets avec une tâche de localisation d'objet supplémentaire. Une attention particulière a été portée à ce domaine ces dernières années. La définition du problème de la détection d'objet est de déterminer où se trouvent les objets dans une image donnée (localisation d'objet) et à quelle catégorie chaque objet appartient (classification d'objet) [40].

Les détecteurs d'objets récents basés sur le deep learning peuvent être classés en deux grandes catégories, les détecteurs à un étage et les détecteurs à deux étages.

Les modèles de la famille R-CNN, Fast RCNN et Faster RCNN [31] sont des detcteurs à deux étages, Ils sont basés sur la structure de région de proposition. La détection se déroule en deux étapes : (1) Premièrement, le modèle propose un ensemble de régions d'intérêt par recherche sélective ou réseau de propositions régionales. Les régions proposées sont nombreuses car les candidats potentiels à la boîte englobante peuvent être infinis. (2) Ensuite, un classificateur ne traite que les candidats de région.

L'autre catégorie saute l'étape de proposition de région et exécute la détection directement sur un échantillonnage dense des emplacements possibles. Elle utilise la structure de Regression/Classification, C'est plus rapide et plus simple, mais cela pourrait potentiellement ralentir un peu les performances. Dans cette section nous expliquons comment fonctionne un algorithme de détection d'objets en une étape.

notre objectif est d'applique un détecteur des objets pour un robot, donc nous avons besoin que la détection soit rapide et en temps réel, pour cela nous concentrons sur la deuxième catégorie, les détecteurs à un étage, qui utilise une structure connue comme la régression/Classifiaction, dans cette section nous expliquons en detail comment l'algorithme de Régression/Classfication fonctionne.

2.3.4.1 Détecteurs à deux étages

Après l'amélioration de l'architecture du réseau de détection d'objets dans R-CNN à Fast R_CNN. Le temps de formation et de détection du réseau diminue considérablement, mais le réseau n'est pas assez rapide pour être utilisé comme un système en temps réel car il faut environ (2 secondes) pour générer une sortie sur une image d'entrée [31].

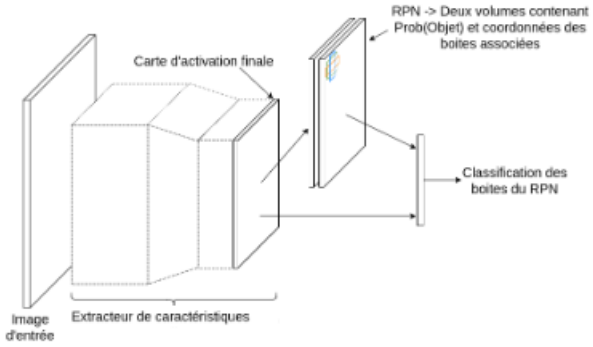


FIGURE 2.14 – Faster R-CNN. Un extracteur de caractéristiques produit une carte d'activation, utilisée pour prédiredes boîtes et probabilités d'objet via un réseau de proposition de régions. Ces détections sont ensuite classifiées pour produire un vecteur de scores de classes associé aux boîtes [20].

Faster RCNN

De la même manière que dans Fast R-CNN, le détecteur faster R-CNN [31] combine les régions et la classification proposées à travers une architecture de réseau classique qui opère un ensemble de convolutions sur l'image d'entrée. La carte d'activation générée alimente alors un RPN "Region Proposition Network" qui utilise une fenêtre glissante pour prédire, à chaque position, un ensemble de cases caractérisées à la fois par leur probabilité de contenir un objet et par leurs coordonnées relatives à des "ancres" d'aspects prédéfinis Ces régions proposées ainsi que la carte d'activation utilisée alimentent ensuite des couches destinées à la classification et à la prédiction des coordonnées des cadres de délimitation correspondants dans l'image d'origine. Cette architecture de détection est illustrée sur la figure 2.14. La particularité de Faster R-CNN réside dans le fait que les couches de convolution situées en amont du RPN mais aussi les couches de classification sont partagées et optimisées pour ces deux problèmes, en même temps, par un apprentissage "alterné" des deux branches. Dans ces réseaux, le temps de test est d'environ 0,2 seconde par image.

2.3.4.2 Détecteurs à un étage

Les détecteurs à un étage basés sur une régression/classification globale, mappant directement des pixels de l'image aux coordonnées de la boîte englobante et aux probabilités de classe, peuvent réduire les dépenses en temps. Nous concentrons sur deux modèles importants, You Only Look Oncde (YOLO) [30] et Single ShotMultiBox Detector (SSD) [24].

YOLO

le modèle yolo est la première tentative de construction d'un détecteur d'objets en temps réel rapide,Redmon et al. [28] ont proposé un cadre, qui utilise toute la carte la plus caractéristique pour prédire à la fois les confidences pour plusieurs catégories et les boîtes englobantes à l'aide d'un reseau de classification pré-formé modifié pour atteindre cet objectif.

L'idée de base de YOLO est présentée à la figure 2.15. YOLO divise l'image d'entrée en une grille $S \times S$. Si le centre d'un objet tombe dans une cellule, cette cellule est responsable de la détection de l'existence de cet objet. Chaque cellule prédit l'emplacement des boîtes englobantes **B**, un score de confiance et une probabilité de classe d'objets conditionnée à l'existence d'un objet dans la boîte englobantes.

Un score de confiance indique la probabilité que la cellule contienne un objet : $Pr(Object)xIoU(pred, truth)$; où Pr = probabilité et IoU = Intersection Over Union. Si la cellule contient un objet, elle prédit une probabilité que cet objet appartienne à chaque classe C_i, $i = 1, \ldots, K$: $Pr(Class_i|Object)$. À cet étage, le modèle ne prévoit qu'un seul ensemble de probabilités de classe par cellule, quel que soit le nombre de boîtes englobantes, **B**. Au total, une image contient des boîtes englobantes $S \times S \times B$, chaque boîte correspondant à 4 prédictions de localisation, 1 score de confiance et K probabilités conditionnelles pour la classification des

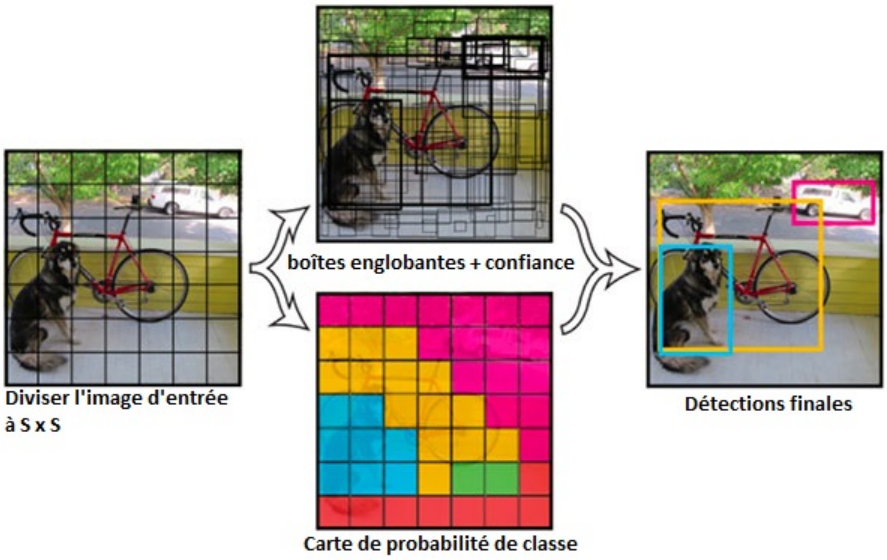


FIGURE 2.15 – L'idée principale de YOLO

objets. Les valeurs de prédiction totales pour une image sont $S \times S \times (5B + K)$, qui est la forme matriceielle de la couche conv finale du modèle [40].

Fonction de perte :

La perte se compose de deux parties, la perte de localisation pour la prédiction de décalage de boîte englobante et la perte de classification pour les probabilités de classe conditionnelles . Les deux parties sont calculées comme la somme des erreurs au carré.

Pendant l'entraînement, la fonction de perte suivante est optimisée [40].

$$L = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

$$L_{cls} = \sum_{i=0}^{s^2} \sum_{j=0}^{B} (1_{ij}^{obj} + \lambda_{noobj}(1 - 1_{ij}^{obj}))(C_i - \hat{C}_i)^2 + \sum_{i=0}^{s^2} \sum_{c \epsilon C} 1_i^{obj} (p_i(c) - \hat{p}_i(c))^2$$

$$L = L_{loc} + L_{cls}$$

Dans une certaine cellule i, $(x_i; y_i)$ désigne le centre de la boîte par rapport aux limites de la cellule de la grille, $(w_i; h_i)$ sont la largeur et la hauteur normalisées par rapport à la taille de l'image, C_i représente les scores de confiance, 1_i^{obj} indique l'existence d'objets et 1_{ij}^{obj} indique que la prédiction est effectuée par le prédicteur de la jième boîte englobante. Notez que seulement lorsqu'un objet est présent dans cette cellule de grille, la fonction de perte pénalise les erreurs de classification. De même, lorsque le prédicteur est responsable de la boîte de vérité au sol (c'est-à-dire le plus haut IoU de tout prédicteur dans cette cellule de grille est atteint), les erreurs de coordonnées de la boîte englobante sont pénalisées [40].

Yolo peut traiter des images en temps réel à 45 FPS et une version simplifiée Fast YOLO peut atteindre 155 FPS avec de meilleurs résultats que les autres détecteurs en temps réel.

YOLOv3 : Introduit pour la première fois en 2016, YOLO a traversé un certain nombre d'itérations différentes, parlant sur YOLO9000 [29] (c'est-à-dire YOLOv2), Grâce à une formation conjointe, les auteurs ont formé YOLO9000 simultanément sur la base de données de classification ImageNet et la base de données de détection COCO, puis en 2018 redmon et Ferhadi ont publié le YOLOv3 [30]. YOLOv3 est nettement plus grand que les modèles précédents, mais, le meilleur de la famille de détecteurs d'objets YOLO.

La première amélioration apportée avec YOLOv3 est l'utilisation de la classification multi-étiquettes, qui est différente de l'étiquetage mutuel exclusif utilisé dans les versions précédentes. Il utilise un classificateur logistique pour calculer la probabilité que l'objet soit d'une étiquette spécifique. Les versions précédentes utilisent la fonction softmax pour générer les probabilités à partir des scores. Pour la perte de classification,

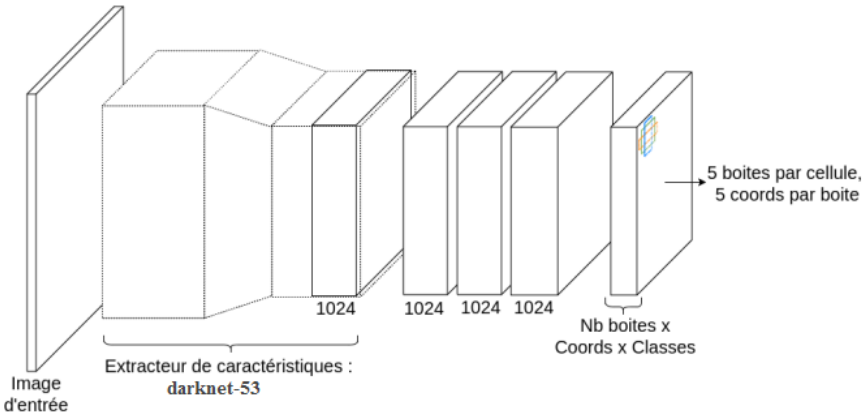


FIGURE 2.16 – L'image d'entrée passe par un extracteur de caractéristiques **Darknet53** ; trois couches convolutionnelles lui sont greffées. Le réseau se base ensuite sur des ancres d'aspects et ratios prédéfinis pour pré- dire des boîtes englobantes et scores de confiance [20].

il utilise la perte d'entropie croisée binaire pour chaque étiquette, au lieu de l'erreur quadratique moyenne générale utilisée dans les versions précédentes.

Un autre amélioration apportée est l'utilisation de la convolution avec des boîtes d'ancrage. Les couches entièrement connectées responsables de la prédiction de la boîte de délimitation sont supprimées et nous déplaçons la prédiction de classe du niveau de la cellule de la grille vers le niveau de la boite de délimitation (Fig. 2.16). L'adoption des boîtes d'ancrage fait une légère diminution du mAP de 0,3% mais améliore le rappel de 81% à 88% augmentant les chances de détecter tous les objets dans l'image.

L'amélioration de YOLOv3 a permis de créer un nouvel extracteur de caractéristique CNN nommé **Darknet-53** (Fig. 2.1). Il s'agit d'un CNN à 53 couches qui utilise un réseau de connexions à sauter inspiré de ResNet . Il utilise également 3 x 3 et 1 x 1 couches convolutives. Il a montré une précision de l'état de l'art, avec une vitesse considérablement bonne.

lors de la préparation de cette thèse, une autre version de YOLO a été publiée [9], YOLOv4, d'après le papier YOLOv4 améliore l'AP et le FPS de YOLOv3 de 10% et 12% respectivement. Dans ce projet nous avons utilisé seulement YOLOv3.

SSD

YOLO a du mal à traiter les petits objets en groupe, ce qui est dû à de fortes contraintes spatiales imposées aux prédictions des boîtes englobantes. Visant ces problèmes, Liu et al. a proposé un Single Shot Detector (SSD) [24], au lieu des grilles fixes adoptées dans YOLO, le SSD tire parti d'un ensemble de boîtes d'ancrage (Anchors Boxes) par défaut avec différents rapports d'aspect et échelles pour discrétiser l'espace de sortie des boîtes englobantes. Pour gérer des objets de différentes tailles, le réseau fusionne les prédictions de plusieurs cartes d'entités avec différentes résolutions (Fig. 2.17).

Le SSD n'a besoin que d'une image d'entrée (et les boîtes de "Ground Truth" pour chaque objet pendant la formation). À chaque emplacement pour plusieurs cartes de caractéristiques (p.ex 8×8 et 4×4 dans la figure 2.17), Liu et al [24] a imposé un petit ensemble de boîtes par défaut "ancres" de différentes échelles et rapport d'aspect, pour chaque boîte d'ancre, ils ont utilisé la convolution (1×1) pour prédire à la fois les décalages ($\Delta(x_1; y_1; x_2; y_2)$) et les confidences pour toutes les catégories d'objets ($(c_1; c_2; \cdots; c_p)$). Pendant le temps de formation, ces boîtes doivent d abord correspondre à la Ground Truth. Par exemple, deux boîtes ont été jumelées un avec le chat et l'autre avec le chien, qui sont traités comme positifs et les autres négatifs. Ensuite, la perte a calculé avec une somme pondérée entre la perte de localisation (par exemple, Smooth L1) et la perte de confiance (par exemple, Softmax), et le Backpropagation fait la minimisation de l'erreur.

l'architecture du SSD est illustrée à la figure 2.18. Compte tenu de l'architecture du backbone(Extracteur de caractéristiques), le SSD ajoute plusieurs couches de fonctionnalités à la fin du réseau, qui sont responsables

	Type	Filtres	Taille	Sortie
	convolutif	32	3x3	256x256
	convolutif	64	3x3/2	128x128
1x	convolutif	32	1x1	
	convolutif	64	3x3	
	Résiduel			128x128
	convolutif	128	3x3/2	64x64
2x	convolutif	64	1x1	
	convolutif	128	3x3	
	Résiduel			64x64
	convolutif	256	3x3/2	32x32
8x	convolutif	128	1x1	
	convolutif	256	3x3	
	Résiduel			32x32
	convolutif	512	3x3/2	16x16
8x	convolutif	128	1x1	
	convolutif	512	3x3	
	Résiduel			
	convolutif	1024	3x3/2	8x8
4x	convolutif	512	1x1	
	convolutif	1024	3x3	
	Résiduel			8x8
	Avgpool	Global		
	Connected	1000		
	Softmax			

TABLE 2.1 – darknet-53 [30].

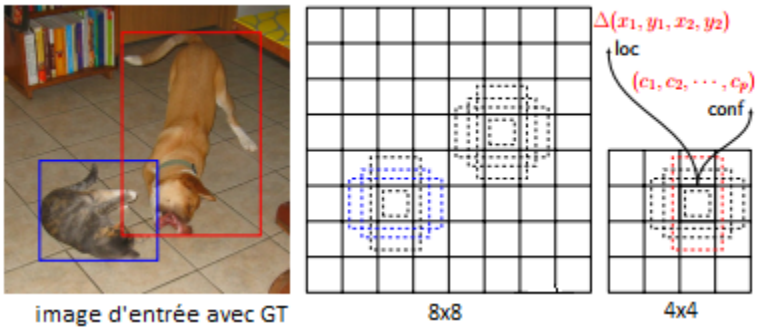


FIGURE 2.17 – boîtes par défaut de l'SSD aux cartes de caractéristiques 8x8 et 4x4

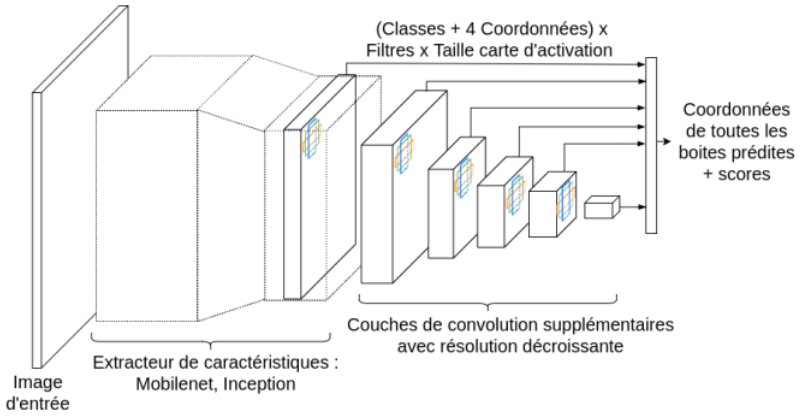


FIGURE 2.18 – SSD. L'image passe par un extracteur de caractéristiques convolutionnel, puis des couches de résolution décroissante y sont ajoutées. A chaque étape, un ensemble de boîtes englobantes et scores de classes sont générés [20].

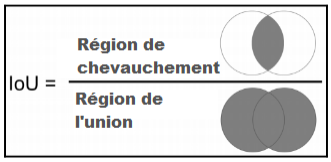


FIGURE 2.19 – Les deux cercles représentent les boîtes englobantes de la vérité terrain (ground truth)et des prédictions. L'IoU est calculée comme un rapport entre la zone de chevauchement et la zone de l'union.

de la prévision des décalages par rapport aux boîtes par défaut avec différentes échelles et proportions et leurs confidences associées.

Intersection Over Union L'IoU est le rapport de la zone de chevauchement de la vérité terrain et de la zone prévue à la zone totale. Voici une explication visuelle de la métrique2.19 :

Non Maximal Supression SSD utilise une suppression non maximale pour supprimer les prédictions en double pointant vers le même objet. SSD trie les prédictions en fonction des scores de confiance. À partir de la prédiction de confiance la plus élevée, SSD évalue si des boîtes de limite précédemment prédites ont une IoU supérieure à 0,45 avec la prédiction actuelle pour la même classe. Si elle est trouvée, la prédiction actuelle sera ignorée. Tout au plus, nous conservons les 200 premières prédictions par image.

le SSD surpasse considérablement le Faster R-CNN en termes de précision sur PASCAL VOC et COCO, tout en étant trois fois plus rapide. Le SSD300 (la taille de l'image d'entrée est de 300 × 300) fonctionne à 59 FPS, ce qui est plus précis et efficace que YOLO. Cependant, le SSD n'est pas compétent pour traiter les petits objets.

fonction de perte :

La perte de localisation entre la boîte prédite I et la boîte de Ground Truth g est définie comme la perte *smooth* L_1 avec c_x, c_y comme le décalage par rapport à la boîte englobante par défaut d de largeur w et de hauteur h.

$$L_{loc}(x,l,g) = \sum_{i\epsilon Pos}^{N} \sum_{m\epsilon\{cx,cy,w,h\}} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m)$$

La perte de confiance est la perte de faire une prédiction de classe. Pour chaque prédiction de match positive, nous pénalisons la perte en fonction du score de confiance de la classe correspondante.Pour les

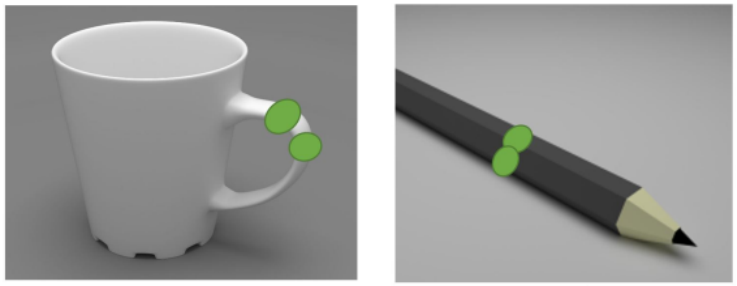


FIGURE 2.20 – Exemple de représentation de point de saisie illustre en modèle D de "Mug" et "Pencil" [10].

prédictions de correspondance négatives, nous pénalisons la perte en fonction du score de confiance de la classe "0" : la classe "0" classifie qu'aucun objet n'est détecté.

2.3.5 Détection de prise en robotique

La détection de prise est identifiée comme la capacité de reconnaitre les points de saisie ou les poses de saisie pour un objet dans une image donnée (Fig. 2.20), une compréhension réussie décrit comment un effecteur terminal robotique peut être orienté au-dessus d'un objet pour maintenir en toute sécurité l'objet entre sa pince et le ramasser. En tant qu'êtres humains, nous utilisons notre vue pour identifier visuellement les objets dans notre voisinage et savoir comment les approcher afin de les ramasser. De manière similaire, les capteurs de perception visuelle sur un système robotique peuvent être utilisés pour produire des informations sur l'environnement qui peuvent être interprétées dans un format utile. Une technique de cartographie est nécessaire pour classer chaque pixel de la scène en fonction de son appartenance ou non à une saisie réussie..[10].

2.3.5.1 Travaux connexes

La détection de saisie en robotique est explorée depuis longtemps [8]. Récemment, l'apprentissage en profondeur offre la possibilité de détecter des saisies directement à partir d'images RGB ou RGB-D , avec sa puissante capacité d'extraction de caractéristiques et à apprendre des fonctionnalités utiles à partir des données. Ils traitent les prises comme un type spécifique d'objet et transfèrent des algorithmes de détection d'objet pour saisir la détection, et obtenir des performances de pointe sur des ensembles de données de saisie d'un seul objet tels que Cornell Grasp Dataset, CMU Grasp Dataset et Jacquard Dataset [38].

La méthode la plus populaire pour la détection de saisie structurée était l'approche à fenêtre glissante proposée par Lenz et al. [22]. Cette méthode a donné une précision de détection de 75% et un temps de traitement de 13,5 s par image. Des résultats similaires ont été rapportés par les études de Wang et al. [36] qui ont suivi une approche similaire. Guo et al. [41] ont utilisé la méthode du rectangle de référence pour identifier les régions saisissables d'une image. Cette méthode a été adaptée à partir des réseaux de neurones de la proposition de région. Cette méthode a été largement considérée comme inadaptée lorsqu'une vitesse de détection en temps réel est nécessaire. Comme alternative, Redmon et al [27] ont proposé la méthode de one-shot-detction. il ont utilisé une régression directe représente les paramètres du rectangle de saisie orientée dans les coordonnées du plan de l'image. Dans la première approche, les auteurs proposé d'utiliser des techniques d'apprentissage par transfert pour prédire la représentation de la saisie [27]. Ils ont signalé une précision de détection de $84,4\%$ en 76 millisecondes par image. Ce résultat a produit une augmentation importante des performances par rapport à la méthode de la fenêtre glissante. Suite à cela, plusieurs travaux [38, 18] apportent une série d'améliorations et obtiennent de meilleures performances.

Suivant une stratégie similaire, Kumra et al. [18] ont signalé une amélioration grâce à laquelle une précision de détection de $89,21\%$ pour leur détecteur de saisie multimodale avec une vitesse de traitement de 100 millisecondes par image a été atteinte. Par conséquent, il était évident que la plupart des travaux de détection en un coup suivaient des techniques d'apprentissage par transfert pour utiliser des architectures de réseaux neuronaux pré-entraînées[12].

Bien que la méthode préférée pour one-shot-detction soit la régression directe de la représentation de saisie, Chu et al. [12] ont rapporté une précision de détection de $94,4\%$ avec des images RGB. S'appuyant sur le concept de Zhou et al. [41] ont proposé d'utiliser les boîtes d'ancrage pour des régions prédéfinies des images. Ils ont atteint une précision de détection de $97,74\%$ pour leur travail, Lu et al. [25], ont réalisé une étude de probabilité de prise sur la base CNN pour obtenir une précision de détection de $75,6$ pour les nouveaux objets et $76,6$ pour les objets précédemment vus au cours de la formation.

Une méthode pour apprendre une fonction de robustesse de préhension optimale a été proposée par Mahler et al. [26] . Les auteurs ont compilé un ensemble de données connu sous le nom de Dex-Net 2.0. Ils ont formé un réseau convolutive de qualité de préhension (GQ-CNN) qui a été utilisé pour apprendre une métrique de robustesse pour les candidats à la préhension. Ils ont testé leur CNN avec leur ensemble de données qui a atteint une précision de $98,1\%$ pour la détection de saisie.

2.4 Conclusion

Le deep learning a connu un succès remarquable dans les applications de la vision par ordinateur telles que la classification, la détection et la localisation. Jusqu'à présent, cependant, il n'a pas été adopté très largement dans les applications robotiques, bien qu'il s'agisse maintenant d'un domaine de recherche en croissance rapide.

Le réseau de neurones fonctionne comme le cerveau humain, mais d'une manière très complexe et difficile, nous avons aujourd'hui pu former un réseau pour simuler le cerveau humain et également effectuer des tâches complexes même pour lui.

Dans ce chapitre, nous avons vu les dernières avancées de l'apprentissage en profondeur, qui ont révolutionné le domaine de l'intelligence artificielle, et la plus intéressante est la détection d'objets, qui sera utile dans les voitures autonomes et aidera également les robots à manipuler leur environnement d'une très bonne manière.

En raison des exigences d'une puissance de calcul plus élevée et de grands volumes de données de formation, il est toujours difficile de mettre en œuvre une approche d'apprentissage de bout en bout pour l'activité de saisie robotique complète.

Chapitre 3

Méthodologie de travail

3.1 Introduction

Ce chapitre donne une introduction à la méthodologie de ce memoire. L'ensemble du développement est divisé en deux blocs principaux formant un système uni pour faire une tâche que nous pouvons considérer comme une tâche difficile.

la première partie, on peut l'appeler comme la partie de la vision par ordinateur, cette partie basée sur un détecteur d'apprentissage en profondeur former à l'aide d'une base de données spécialement pour la détection de prise, le détecteur sera le résultat de la comparaison entre les deux détecteurs à un étage les plus efficaces, une fois la formation terminée, de nouveaux échantillons (différents de ceux en formation) sont testés à travers le détecteur et les résultats de détection définissent la précision de ce modèle.

Ce dernier utilisant un Raspberry Pi et une caméra RGB peut trouver les coordonnées de la position de prise pour chaque objet dans la scène, Il est important de souligner que tout ça fonctionne en temps réel.

la deuxième partie est la parte de robotique, les coordonnées qui sont arrivées de la partie de la vision va transformer en coordonnées articulaires à l'aide d'un algorithme de cinématique inverse. Les étapes sont indiquées sur la figure 3.11, chaque étape sera détaillée dans les sections suivantes.

3.2 Problématique de Vision par Ordianteur :

Cette section représente la première étape du travail du système, la détection des points de saisie par un modèle d'apprentissage en profondeur. En s'entrainant sur deux bases de données différentes, le modèle résultant pourrait détecter la classe et les points de saisie d'un objet dans la scène. Le principe de fonctionnement est détaillé dans la figure 3.2.

Dans cette section, nous nous concentrerons sur l'analyse du résultat de l'expérience et la comparaison de la précision des différents modèles sur les bases de données les plus connues. En même temps, nous allons donner un aperçu général sur certaines bases de données, les statistiques de ces ensembles de données sont présentées dans le tableau 3.1. Enfin, nous allons choisir le modèle le plus performant pour la formation.

3.2.1 La base de données

Il est très important de prendre en compte le choix d'un ensemble de données d'apprentissage et de test, notamment la taille, mais également si elles correspondent à l'environnement et au but de l'application. Ici, nous nous préoccupons de détecter la catégorie et la position de saisie (prise) pour un objet donné, pour cela nous avons choisi la base de données Cornell Grap Dataset avec une autre base de données connue, Les statistiques de ces ensembles de données sont présentées dans le tableau 1

Pré-entrainement L'apprentissage par transfert est un concept opposé à l'apprentissage à partir de zéro. Lorsqu'ils sont formés à partir de zéro, les réseaux sont généralement initialisés avec des variables aléatoires. En revanche, dans l'apprentissage par transfert, ils utilisent certains facteurs d'un modèle pré-entraîné. Le

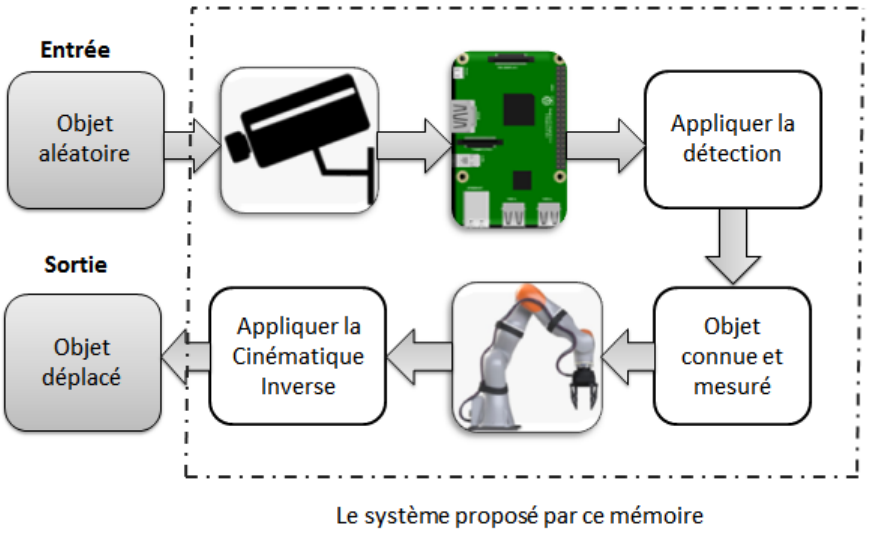


FIGURE 3.1 – Les étapes de fonctionnement du système.

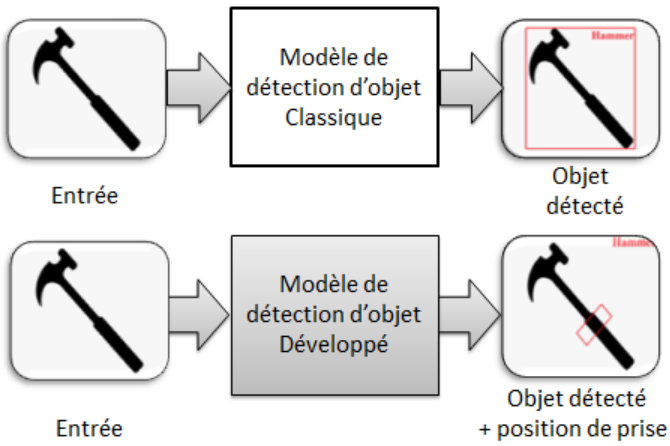


FIGURE 3.2 – Schéma représentant la différence entre un détecteur d'objet normal (classique) et un détecteur d'objet développé (détecte la classe et la position de prise).

FIGURE 3.3 – Échantillons d'images de la base de données Pascal VOC avec détection d'objets

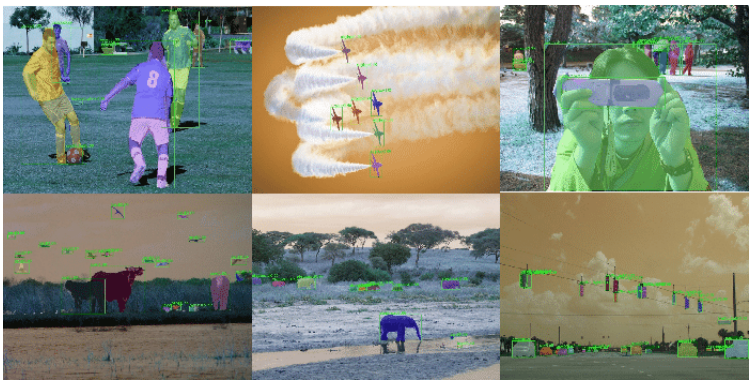


FIGURE 3.4 – Échantillons d'images de la base de données MSCOCO avec détection et segmentation d'objets

pré-entrainement est nécessaire lorsque les données disponibles spécifiques au domaine sont limitées, comme dans l'ensemble de données de CGD.

Les bases de données de classification sont souvent utilisés comme étape de pre-entrainnement pour amorcer les réseaux pour l'apprentissage de la détection d'objets. Ici, nous considérons les bases de données Pascal VOC et COCO pour ce but.

Base de données Pascal VOC

Cet ensemble de données contient les données du PASCAL Visual Object Classes Challenge 2007, alias VOC2007, correspondant aux épreuves de classification et de détection. Un total de 9963 images sont inclus dans cet ensemble de données, où chaque image contient un ensemble d'objets, sur 20 classes différentes, soit un total de 24640 objets annotés (Fig. 3.3). Dans l'épreuve de classification, l'objectif est de prédire l'ensemble d'étiquettes contenues dans l'image, tandis que dans la tâche de détection, l'objectif est de prédire la boite de délimitation et l'étiquette de chaque objet individuel [14].

MSCOCO

COCO est un ensemble de données de détection, de segmentation et de sous-titrage d'objets à grande échelle. Cette version contient des images, des cadres de délimitation et des étiquettes pour la version 2014 (Fig. 3.4). Certaines images du train et des ensembles de validation n'ont pas d'annotations. Coco 2014 et 2017 utilisent les mêmes images, mais des divisions train/val/test différentes. Le partage de test n'a pas d'annotations (seulement des images). Coco définit 91 classes mais les données n'utilisent que 80 classes [23].

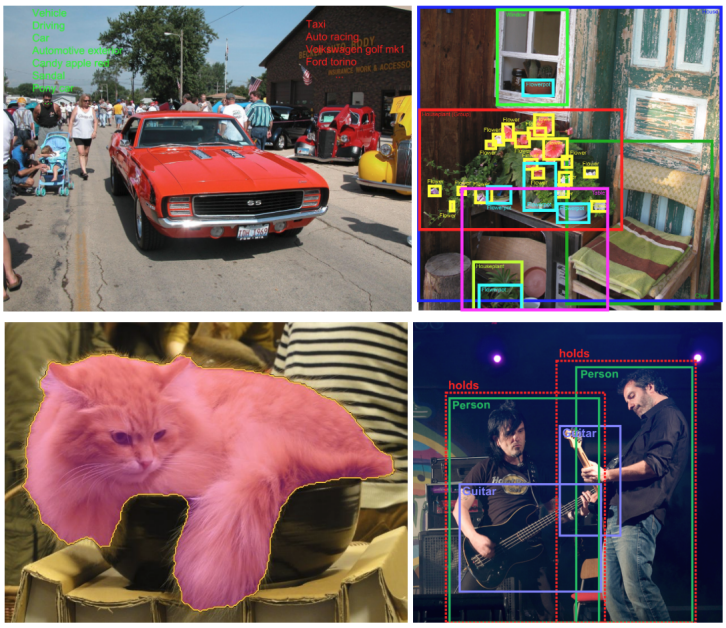


FIGURE 3.5 – Échantillons d'images de la base de données Open Images avec détection et segmentation d'objets

Google Open Images

Open Images est un ensemble de données d'environ 9 millions d'images annotées avec des étiquettes au niveau de l'image, des boites de délimitation d'objets, des masques de segmentation d'objets, des relations visuelles et des récits localisés. Il contient un total de 16 millions de cadres de délimitation pour 600 classes d'objets sur 1,9 million d'images, ce qui en fait le plus grand ensemble de données existant avec des annotations d'emplacement d'objet. Les boites ont été en grande partie dessinées manuellement par des annotateurs professionnels pour assurer l'exactitude et la cohérence. Les images sont très diverses et contiennent souvent des scènes complexes avec plusieurs objets (8,3 par image en moyenne). Dans la version 5, ils ont ajouté des masques de segmentation pour 2,8 millions d'instances d'objets dans 350 classes. Dans la V6, ils ont ajouté 507 000 récits localisés : descriptions multimodales d'images constituées de traces synchronisées de voix, de texte et de souris sur les objets décrits. Enfin, l'ensemble de données est annoté avec 59,9 millions d'étiquettes au niveau de l'image couvrant 19 957 classes [19].

L'idée que d'avoir un seul jeu de données avec des annotations unifiées pour la classification des images, la détection d'objets, la détection des relations visuelles, la segmentation d'instances et les descriptions d'images multimodales permettra d'étudier ces tâches conjointement et de stimuler les progrès vers une véritable compréhension de la scène (Fig. 3.5).

Cornell Grap Dataset

La CGD est apparue dans un certain nombre d'études de recherche menées au cours des dernières années, ce qui pourrait donner à penser qu'elle présente une diversité raisonnable d'exemples de prises généralisées. La tendance récente à utiliser le RGB-D pour apprendre à prédire les saisies a été couverte par l'ensemble de données CGD grâce à l'inclusion de données en nuage de points par ses créateurs. Un sens des bonnes et des mauvaises prises était également nécessaire pour différencier une meilleure compréhension des alternatives. Par conséquent, le CGD pourrait être sélectionné comme un ensemble de données approprié pour sa qualité et son adaptabilité. Le CGD a été largement utilisé dans [39, 36, 35, 27, 25, 26, 18, 38].

La CGD a été créé avec un ensemble des rectangles à saisir avec 280 types d'objets différents. Cet ensemble de données se compose de 1035 images de 280 objets différents, avec plusieurs images prises de chaque objet

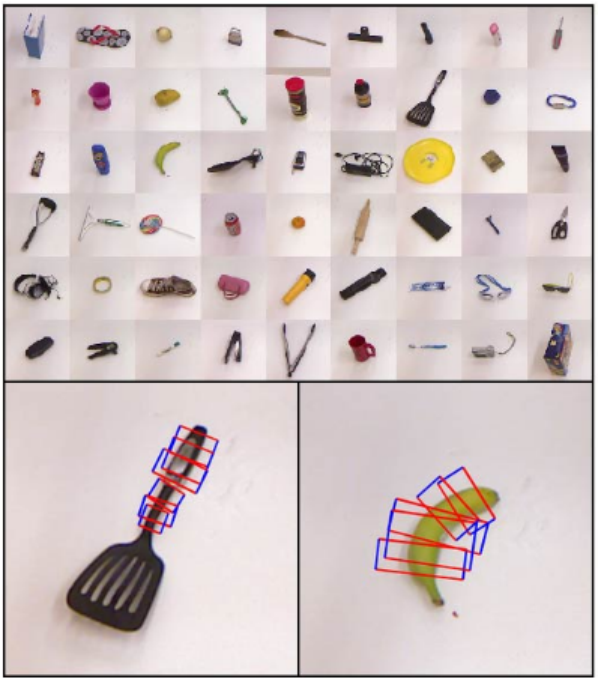


FIGURE 3.6 – L'ensemble de données de saisie Cornell contient une variété d'objets, chacun avec plusieurs prises étiquetées. Les positions de saisie sont données sous forme de rectangles orientés en 2D[27].

Ensemble de donnees	Entrainement		Validation		test	
	images	objet	images	objet	images	objet
VOC-2007	2501	6301	2510	6307	4952	14976
VOC-2012	5717	13609	5823	13841	10991	-
MSCOCO-2015	82783	604907	40504	291875	81434	-
MSCOCO-2018	118287	860001	5000	36781	40670	-
OID-2018	1743042	14610229	41620	204621	125436	625282
Cornell Grasp Dataset	1035	280	-	-	-	-

TABLE 3.1 – Certains ensembles de données de détection d'objets bien connus et leurs statistiques [42].

dans différentes orientations ou poses, Chaque image a également un nuage de points associé et une image d'arrière-plan. (Une seule image d'arrière-plan est utilisée pour les images d'objets numériques). L'ensemble de données brutes ci-dessous (Fig. 3.6) comprend : (a) des images, (b) des rectangles de saisie, (c) des nuages de points, (d) des images d'arrière-plan,

tableau :

3.2.2 Choix de l'architecture

Selon le chapitre 2, afin de mettre en œuvre la détection en temps réel, le détecteur approprié est un détecteur à un seul étage, car les détecteurs à un étage sont plus rapides et atteignent des performances raisonnables, dans cette sous-section, nous comparons les modèles YOLOv3 & SSD sur les base de données Pascal VOC 07+12 et COCO, puis nous choisissons le meilleur modèle pour la formation.

Les modèles pré-entrainés sont une merveilleuse source d'aide pour les personnes qui cherchent à apprendre un algorithme ou à essayer un modèle existant. En raison de contraintes de temps ou de contraintes de calcul, il n'est pas toujours possible de créer un modèle à partir de zéro, c'est pourquoi des modèles pré-entraines existent ! Le modèle pré-entrainé peut être utilisé comme une référence pour améliorer le modèle existant ou tester votre propre modèle par rapport à celui-ci. Le potentiel et les possibilités sont vastes.

Méthode	mAP	FPS	la taille de Batch	résoulution d'entrée
SSD300	74,3	59	8	300x300
SSD512	**76,9**	**22**	**8**	**512x512**
YOLOv1	63,4	21	1	448x448
YOLOv2	73,4	40	1	554x554

TABLE 3.2 – Résultats sur l'ensemble de test PASCAL VOC 2012.[29]

modèle	formation	teste	mAP	FPS	résoulution d'entrée
SSD300	COCO	test-dev	41,2	45	300x300
SSD512	COCO	test-dev	46,5	19	512x512
YOLOV2	COCO	test-dev	48,1	40	608x608
YOLOV3	**COCO**	**test-dev**	**57,9**	**20**	**608x608**

TABLE 3.3 – Résultats sur l'ensemble de test COCO test-dev.[30]

Dans cette expérience, nous essayons de faire une détection d'objets à l'aide de la bibliothèque Opencv et des modèles d'apprentissage en profondeur pré-entrainés de YOLOv3 et SSD. Ces modèles pré-entrainés ont été formés par des contributeurs sur github. Concernant le YOLOv3, nous avons utilisé le modèle fourni par le site officiel de YOLO, le modèle formé utilise le darknet53 comme extracteur de caractéristique, afin que les paramètres ne soient pas modifiés par rapport à celui utilisé dans l'article, la même chose pour le SSD, sauf que le modèle est livré avec VGG16 comme un extracteur de caractéristique.

3.2.3 Comparaison entre les deux modèles

Il est très difficile d'avoir une comparaison juste entre les différents détecteurs d'objets. Il n'y a pas de réponse directe sur quel modèle est le meilleur. Pour les applications réelles, nous faisons des choix pour équilibrer la précision et la vitesse.

Ici nous allons essayer de comparer les deux algorithmes entre précision et sensibilité sur l'ensemble de données COCO, nous avons essayé quelques images de test de COCO, mais dans ce document, nous allons montrer seulement quelques photos 3.4.

Comme ils l'ont mentionné dans le document, le SSD300 atteint 74,3 % mAP à 59 FPS tandis que le SSD500 atteint 76,9 % mAP à 22 FPS sur le test VOC Pascal 07+12, ce qui surpasse YOLOv1 (63,4 % mAP à 45 FPS) et YOLOv2 (73,4 % mAP à 40 FPS).3.2

En même temps, YOLOv3 (le système de détection d'objets de l'état de l'art3.3), Sur un Titan X il traite des images à 30 FPS et a une mAP de 57,9% sur test-dev.. qui surpasse SSD300 (41,2 % mAP à 46 FPS) et SSD512 (46,5 % mAP à 19 FPS).

Cette valeur indique que lorsqu'ils classent un objet comme voiture, il est très probable que cet objet soit une voiture. Donc, la capacité des algorithmes à détecter les vraies voitures est très élevée. Le pourcentage que les algorithmes classent les objets non-voiture comme voiture est très rare (0,34% pour SSD contre 0,17% pour YOLOv3). Mais en comparant la détection globale sur tous les objets, on note que YOLOv3 surpasse clairement SSD dans cet exemple (Fig. 3.4). Il mesure la sensibilité et la capacité de l'algorithme à détecter tous les objets de l'image. YOLOv3 est plus capable d'extraire toutes les instances des objets dans une image, le SSD manque certaines instances plus que YOLOv3. Cela donne une idée globale de la robustesse de l'algorithme.

Concernant le temps de traitement, nous avons mesuré le temps de traitement pour quelques images qui contiennent des objets différents avec des échelles différentes. Pour YOLOv3, le temps de traitement de chaque image varie entre 1,8 s et 3,2 s avec une moyenne de 2,5 s. Pour le SSD, le temps de traitement de chaque image varie entre 0,4 s et 1,9 s avec une moyenne de 1,15 s. Ceci montre un petit écart entre les deux algorithmes dans le traitement d'une image par temps. Nous avons également noté que le temps de traitement dépend de la taille de l'image. La moyenne est délattée en fonction de la taille.

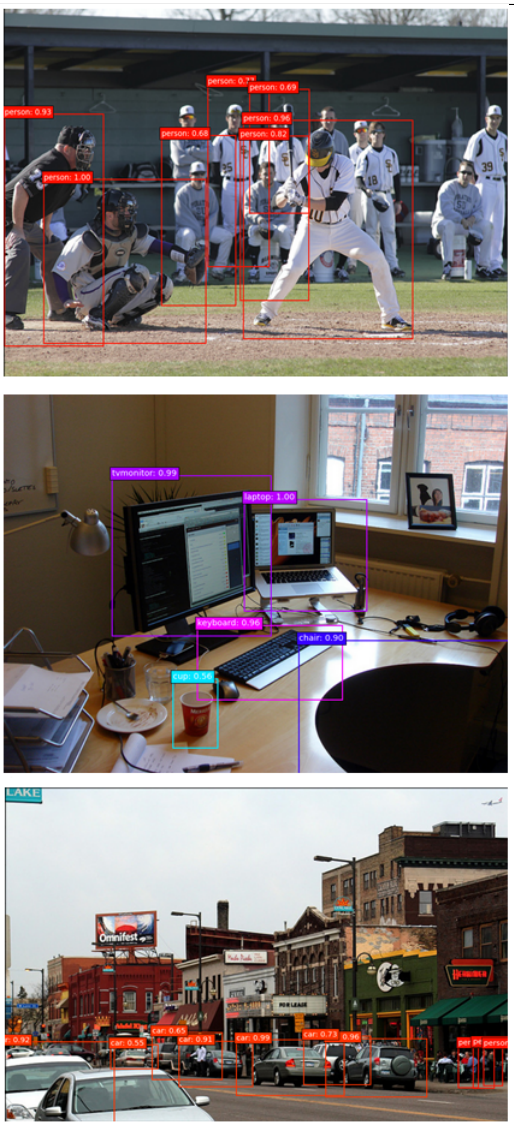
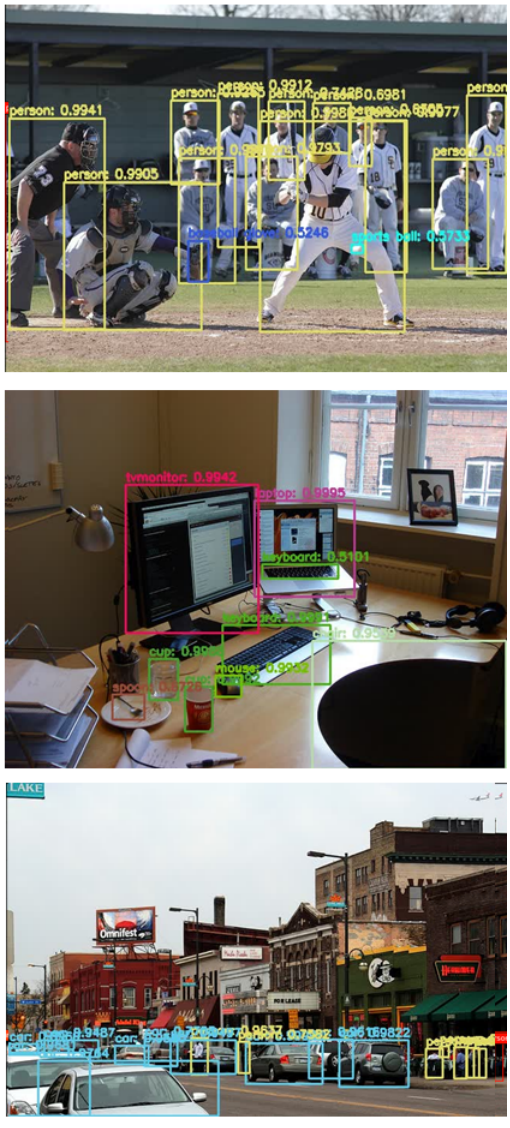
SSD	YOLOv3
	

TABLE 3.4 – Comparaison entre les deux modèles (YOLOv3 et SSD).

Conclusion

La comparaison a révélé que les deux algorithmes sont comparables en précision, ce qui signifie qu'ils ont tous les deux une grande capacité à classer correctement les objets de l'image. Mais nous avons constaté que YOLOv3 surpasse le SSD en sensibilité, ce qui signifie que YOLOV3 est plus capable d'extraire tous les objets dans l'image avec une grande précision. Concernant le temps de traitement pour une détection sur une image, nous avons également constaté que le SSD peut traiter un peu plus rapidement que YOLOv3.

Pour cette comparaison nous avons choisi le YOLOv3 comme le meilleur modèle pour cette contribution.

3.2.4 Formation de modèle

Dans cette partie, nous allons faire l'entrainement sur les deux bases de données (Open Images & Cornell Grasp), afin que les bases de données soient prêtes à s'entrainer sur YOLOv3 en utilisant le framework Darknet, nous avons fait quelques opérations sur le fichier d'annotation qui est fourni par ces ensembles de données. Après ces opérations, les jeux de données sont prêts à être fournis dans le modèle.

3.2.4.1 Préparation de poste de travail

Google Colab

Colaboratory, souvent raccourci en "Colab", est un produit de "Google Research". Colab permet à n'importe qui d'écrire et d'exécuter le code Python de son choix par le biais du navigateur. C'est un environnement particulièrement adapté au machine learning, à l'analyse de données et à l'éducation. En termes plus techniques, Colab est un service hébergé de "Jupyter Notebooks" qui ne nécessite aucune configuration et permet d'accéder gratuitement à des ressources informatiques, dont des GPU.

Les ressources de Colab ne sont pas illimitées, et l'accès n'est pas garanti. De plus, les limites d'utilisation sont susceptibles de fluctuer. Ces contraintes sont nécessaires pour maintenir un accès gratuit aux ressources de Colab.

Darknet framework

Darknet est un framework de réseau neuronal open source écrit en C et CUDA. Il est rapide, facile à installer et prend en charge le calcul CPU et GPU. Le Darknet est un framework relativement moins populaire, dans lequel sa popularité a augmenté après l'introduction de YOLO depuis qu'il a été initialement introduit avec lui. Bien que les implémentations de YOLO aient également été portées sur d'autres frameworks, en particulier caffe, mais ce n'est pas aussi stable que Darknet. Étant donné que la plupart des développements dans ce domaine sont effectués par la communauté open source, il faudra un certain temps avant qu'une version stable de YOLO avec toutes les fonctionnalités soit disponible.

3.2.4.2 Entrainement

Tout d'abord, il est important de considérer que Colab utilise 3 types d'exécution, CPU, GPU et TPU (Tensor Processing Unit), il est spécialement conçu pour l'apprentissage automatique. Le temps d'accès au GPU et au TPU est limité, en raison de la charge sur les serveurs de Google, 12 heures pour une session. Nous avons choisi le type d'exécution en mode GPU, le modèle GPU de Colab est Nvidia Tesla T4 16 Go.

Comme nous l'avons mentionné, nous allons utiliser le Darknet framework, donc nous devons le construire sur VM (Machine virtuelle de Colab), en suivant les insturctions de ce site [1], nous l'avons construit et nous le testons avec un modèle pré-entraîné sur COCO . Pour la formation avec Darknet, nous avons besoin de quatre fichiers, "obj.names", "yolov3_custom.cfg", "obj.data" et un autre fichier contient les images avec des annotations converties au format yolov3. La technique d'apprentissage par transfert a été appliquée en utilisant le réseau pré-formé avec l'ensemble de données COCO.

Sur Open Image dataset

Dans Darknet, contrairement à de nombreux autres frameworks, les paramètres de formation sont inclus dans le même fichier qui décrit l'architecture du réseau. Ce fichier est appelé fichier de configuration et porte l'extension ".cfg". Certains des paramètres mentionnés tels que le taux d'apprentissage, le nombre d'époques (poches), le pas et les échelles sont destinés uniquement à la formation et ne sont pas intéressants dans

ce travail. Certains paramètres sont utilisés pour de légères retouches d'image avant de l'introduire dans le réseau. Ceux-ci incluent l'angle, la saturation, l'exposition et la teinte.

Les paramètres restants spécifient la taille de l'image, le nombre de canaux de couleur et le nombre d'images introduites dans le réseau de manière parallèle. Ces paramètres ne seront pas abordés en détail car ils sont utilisés lors de la formation, ce qui n'est pas l'objet de ce mémoire.

Nous allons utiliser Open Image Dataset pour la détection d'objets. comme nous l'avons mentionné, cet ensemble de données contient plus des 600 classes (catégorie), et avec cette quantité des images et des annotations, c'est impossible de faire l'entrainement avec notre matériel disponible. Pour éviter ça, nous allons utiliser un super outil (OID toolskit) pour télécharger certaines catégories individuellement. Nous choisissons 5 catégories, ce sont 'Hammer', 'Screwdriver', ' Toy', 'Pen' , et 'Remote control'. En utilisant cet outil , nous avons obtenu environ 300 images et fichier d'annotation par chaque catégorie.

Concernant les paramètres, nous avons utilisé la configuration par défaut de yolov3. Nous avons optimisé la formation en utilisant la descente de gradient stochastique avec un momentum défini à 0,9. Le taux d'apprentissage est réglé à 0,001 et la décroissance du poids est réglée à 0,005. La valeur pour la hauteur et la largeur est définie sur 608. La taille du lot est définie sur 64. Concernant les paramètres YOLOv3, l'image d'entrée est subdivisée en 16*16 grilles. Les ancres qui chevauchent l'objet de moins d'une valeur seuil (0,5) sont ignorées.

L'apprentissage en entiera été poursuivi pendant 10000 époques avec une étape à chacune. L'entraînement du Yolov3 a duré environ 36 h, considérant que Colab n'offre que 12 h pour une séance, nous avons donc sauvegardé les poids à toutes les 100 époques, puis nous continuons l'apprentissage.

Sur Cornell Grasp Dataset

Concernant le Cornell Grasp Dataset, nous avons utilisé les mêmes paramétres sauf que nous n'avons pas de classes ici, seulement des boites et des angles, la raison est que cette base de données est dédiée uniquement pour la détection de prise est qu'elle ne contient pas de classe, pour cela nous avons formé un autre modèle de yolov3 sur Open Image pour faire la classification et la détection en même temps sur le même objet. La première tentative de formation sur Cornell Grasp Dataset a donné des résultats insatisfaisants. Par conséquent, un nouveau processus d'apprentissage a été initialisé avec une valeur de taux d'apprentissage de 0,0001 avec 10 000 itérations.

3.2.5 Détection de la région d'intérêt

3.2.5.1 Objectif

Le but lors de la recherche de régions d'intérêt est la forme complète d'un objet dans une image, pour utiliser cette forme pour détecter la position où l'objet peut être pris. Dans notre cas, ces régions ont été sélectionnées et définies manuellement, il ne reste plus que les utiliser pour détecter les régions similaires dans une nouvelle scène. Pour trouver ces régions d'intérêt dans une image, le premier plan de l'image est segmenté à partir de l'arrière-plan à l'aide du seuillage. L'image est réduite à une taille plus petite avant d'être floue. L'arrière-plan est ensuite soustrait et une image binaire est créée. Les régions d'intérêt sont des pixels blancs connectés dans la sortie d'image de thresholding.

3.2.5.2 Configuration manuelle de la région d'intérêt

Les images sont généralement représentées dans des espaces colorimétriques, le plus populaire étant RGB. Cela signifie que chaque pixel a une valeur rouge, verte et bleue. Par conséquent, pour chaque pixel de la régions d'intérêt, nous avons un ensemble de nombres, par exemple (255, 255, 255) qui indiquent le niveau de chaque composante de couleur dans cet pixel.

Pour connaitre la région d'intérêt d'un objet, nous allons utiliser un outil qui nous l'avons trouvé sur Github, cette outil à nous permet de séparer manuellement les couleurs de régions d'intérêt d'un objet des autres couleurs dans l'image, comme nous pouvons le voir sur la figure 3.7.

Nous avons deux fenêtres, l'une représente l'image d'origine et l'autre l'image avec le thresholding (binarisation 0 ou 1), la façon de convertir les pixels en zéro ou en un est basée sur la région d'intérêt que nous voulons définir, si un pixel est trouvé dans cette région, l'algorithme le convertira en zéro, sinon il le

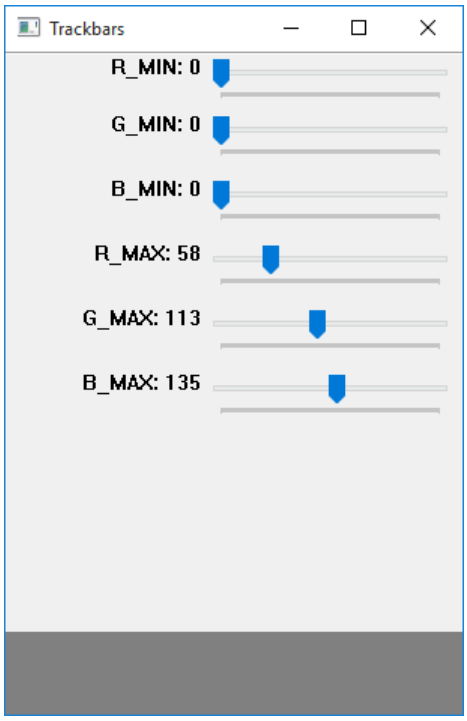


FIGURE 3.7 – Configuration manuelle pour sélectionner la régions d'intérêt d'un objet par un masque RGB.

convertira en un, le résultat est montré sur la figure 3.8. Avec cet outil, nous pouvons sélectionner la région d'intérêt exacte de couleurs à partir d'un objet souhaité.

3.3 Intégration

Pour la détection nous avons besoin d'un outil de petite taille et puissant en même temps, c'est pourquoi nous allons utiliser une simple carte en fait c'est un petit ordinateur appelé "Raspberry Pi" pour la mise en œuvre de la détection d'objets en temps réel, dans cette partie, nous allons présenter le Raspberry Pi et pourquoi c'est le meilleur choix pour nous.

3.3.1 Raspberry Pi

Raspberry Pi est un ordinateur complet de la taille d'une carte de crédit qui exécute le système d'exploitation Linux. Raspberry Pi a tous les ports de connexion nécessaires où l'utilisateur peut brancher des périphériques. Un moniteur peut être branché via une prise HDMI, une souris et un clavier aux ports USB et pour les haut-parleurs, Raspberry fournit une prise audio 3,5 mm. Dans le modèle 3 B , il existe également une prise Ethernet pour la connexion Internet et une carte de réseaux sans fil. En fait, c'est un petit ordinateur qui peut être utilisé pour les mêmes tâches qui peuvent être effectuées avec un ordinateur de bureau normal. Par exemple, lire des e-mails, surfer sur les sites Web, traiter de texte ou regarder des vidéos haute définition. Il est également très populaire dans différents types de projets électroniques et comme outil d'apprentissage de la programmation.

Évidemment, pour sa taille il ne faut pas s'attendre à des performances incroyables, mais implémenter des prototypes de projets à montrer, ou expérimenter avec Linux est plus que suffisant.

Pourquoi le Raspberry Pi ?

Compte tenu des exigences de performances relativement élevées du traitement d'image en général et de l'apprentissage profond en particulier, nous avons besoin d'une plate-forme embarquée puissante capable de

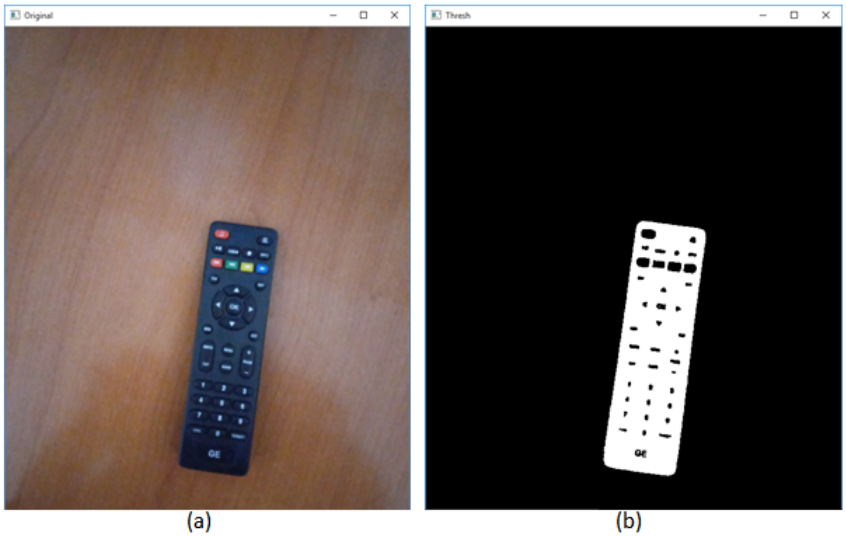


FIGURE 3.8 – (a) Entrée : l'image originale que nous voulons la traiter. (b) Sortie : l'image en binaire (0 ou 1) selon le seuil (Threshold) avec la région d'intérêt (marqué comme des 1) que nous avons défini dans la barre d'outils.

FIGURE 3.9 – Raspberry Pi 3 modèle B 1G de RAM

Carte mère	Raspberry Pi 3 Type B
CPU	Quad-core ARM Cortex-A53 1.2 GHz (Broadcom BCM2837)
RAM	1024 Mo
GPU	Dual Core VideoCore IV Multimedia Co-Processor
Ports	HDMI, 4x USB, RJ45, jack 3.5 mm, connecteurs pour APN et Crane tactile
Wi-Fi	b/g/n et Bluetooth 4.1
Taille/poids	85.0*56.0*17(mm)/40g

TABLE 3.5 – Caractéristiques de Raspberry Pi 3 modèle B

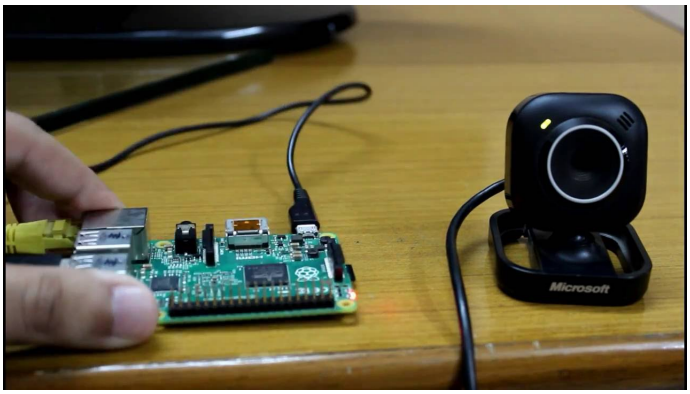


FIGURE 3.10 – Raspberry pi 3 modèle B connecté avec une camera Webcam de résolution "800x600".

gérer ces exigences, en examinant les autres cartes et leur caractère tel que l'Arduino, qui a conduit à décider que c'est le Raspberry Pi est le meilleur choix.Les spécifications matérielles prises en compte pour ce travail sont présentées dans le tableau 3.5.

Le Raspberry Pi 3 Modèle B (Fig. 3.9) a été utilisé dans ce projet. Il dispose d'un nombre de cœurs de processeur et d'une vitesse supérieurs par rapport aux modèles précédents. Les performances du Raspberry Pi 3 modèle B sont nettement inférieures à celles des ordinateurs portables et de bureau actuels. Cela pose un défi, car la plupart des tâches de vision par ordinateur sont couteuses en calcul. Le fait de n'avoir que 1 Go de RAM, partagé entre le CPU et le GPU, est un autre facteur qui pourrait être un défi.

3.3.2 Préparation de Raspberry

La conception du Raspberry Pi n'inclut pas de disque dur intégré ou de lecteur à état solide, mais utilise plutôt une carte SD pour le démarrage et le stockage à long terme. Cette carte est destinée à exécuter des systèmes d'exploitation basés sur Linux Debian. Le module Raspberry Pi que nous avons choisi dispose d'une carte micro SD SunDisk classe 4 de 16 Go préchargée avec le système d'exploitation de Raspberry "Rasbian". Ce système a été téléchargé à partir du site officiel de Raspberry, nous avons utilisé un autre outil appelé "Win32 Disk imager " pour graver l'image du système sur la carte SD.

La dernière version de Raspbian Wheezy est utilisée sur la carte. Après avoir installé le système d'exploitation sur la carte, nous connectons tous les composants matériels nécessaires et allumez l'alimentation.

Bien qu'il convient de mentionner que les applications de traitement d'image, même l'affichage de l'alimentation de la caméra en direct, nécessitent beaucoup de puissance de traitement du processeur et il est donc préférable qu'une partie puisse être épargnée en connectant simplement le Pi à un moniteur et à une souris et/ou un clavier sans fil et une camera webcam (comme le montre la figure 3.10).

D'autre part, une fois que le système est correctement configuré, la première chose à faire est de vérifier les paramètres réseau (sans fil ou via un câble RJ45) pour installer diverses mises à jour, packages et très certainement OpenCV, ce qui est très important dans les projets de traitement d'image, et il faut plus de 10 heures pour l'installer et le compiler (cmake), après l'avoir téléchargé, nous le laissons installer pendant la nuit, les étapes de préparation de Raspberry sont bien structuré sur la figure 3.11.

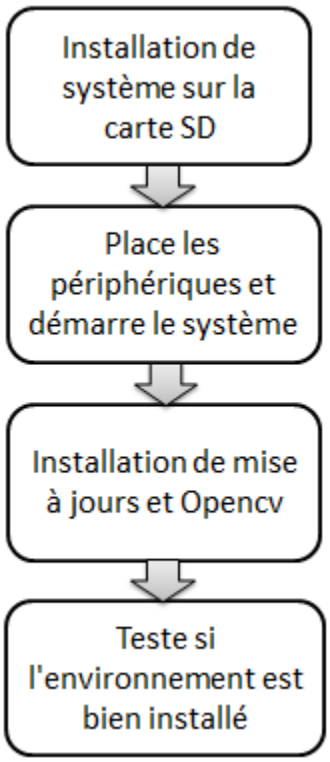


FIGURE 3.11 – Les étapes de préparation de Raspberry Pi pour l'expérience.

Une fois l'installation terminée, un test est nécessaire pour confirmer que l'environnement est prêt et opencv a été correctement installé, un simple script pour lire et afficher l'image suffit. D'après cela, le Raspberry est prêt pour la détection d'objets avec Opencv et un modèle yolov3 pré-entrainé.

3.4 Problématique de Robotique

Tout ce travail mène à cette partie, le robot recevra les coordonnées de Raspberry et effectue son mouvement et déplace l'objet. pour cela, nous allons donner les modèles de robots géométriques et cinématiques, pour nous aider à calculer la cinématique inverse. pour certaines raisons, nous avons utilisé un logiciel de simulation robotique " V-rep" pour simuler la tâche.

3.4.1 V-rep

V-REP "Virtual Robot Experimentation Platform", est un logiciel de simulation de robot 3D, avec environnement de développement intégré, qui vous permet de modéliser, éditer, programmer et simuler n'importe quel robot ou système robotique (par exemple capteurs, mécanismes, etc.).

Il offre une multitude de fonctionnalités qui peuvent être facilement intégrées et combinées via une API exhaustive et une fonctionnalité de script. V-REP est utilisé pour la surveillance à distance, le contrôle du matériel, le prototypage et la vérification rapides, le développement rapide d'algorithmes / l'ajustement des paramètres, le double contrôle de sécurité, l'éducation liée à la robotique ou les simulations d'automatisation d'usine. Interface du logiciel est montrée dans la figure 3.12.

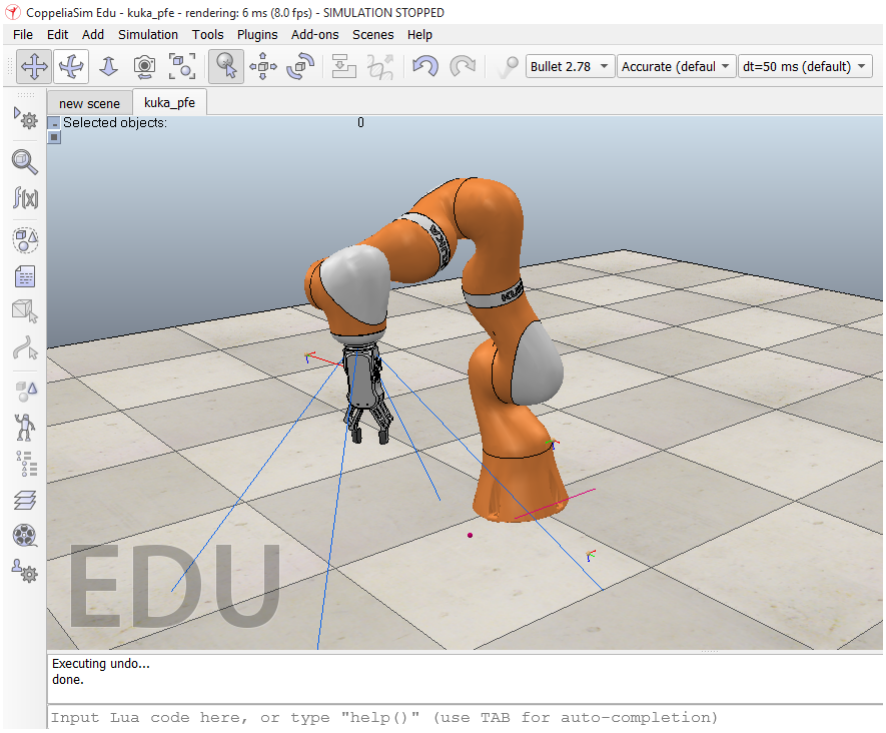


FIGURE 3.12 – Interface du utilisateur de V-rep avec le robot kuka lbr iiwa.

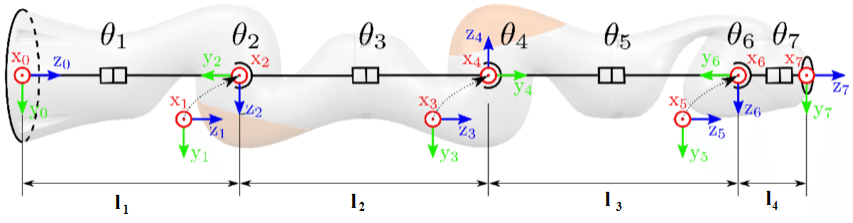


FIGURE 3.13 – Structure générique du manipulateur, variables conjointes et cadres DH affectés.

3.4.2 Modélisation de robot kuka

Le Kuka LBR iiwa est un manipulateur à sept degrés de liberté (DOF). Par conséquent, ce bras manipulateur articulé est redondant, avec au moins un DOF redondant, par rapport à la dimension de la tâche.

Modélisation géométrique

Le modèle géométrique représente une relation entre l'espace cartésien (coordonnées) et les articulations qui concernent le robot. Un ensemble possible de paramètres Denavit-Hartenberg (DH) qui décrivent la chaîne cinématique d'un manipulateur série kuka lbr iiwa sont énumérés dans le tableau 3.6 .

Le problème de la géométrique directe est facilement résolu une fois les paramètres DH déterminés. Quatre paramètres sont attribués à chaque joint, qui se transforment en une matrice de transformation qui établit la relation entre un référentiel attribué $(i-1)$ et le suivant (i),

i	$\alpha i(rad)$	$\theta i(rad)$	$di(mm)$	$ri(mm)$
1	0	θ_1	0	360
2	$-\pi/2$	θ_2	0	0
3	$\pi/2$	θ_3	0	420
4	$\pi/2$	θ_4	0	0
5	$-\pi/2$	θ_5	0	400
6	-$\pi/2$	θ_6	0	0
7	$\pi/2$	θ_7	0	126

TABLE 3.6 – Paramtres DH de robot kuka lbr iiwa [15]

$$ {}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i c\alpha_i & a_i c\theta_i \\ c\theta_i & c\theta_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Le produit de ces matrices de la base à la bride, ${}^0T_7 = {}^0\ T_1^1 T_2^2 T_3^3 T_4^4 T_5^5 T_6^6 T_7$, renvoie la pose du manipulateur dans l'espace des tâches. En appliquant cette formule :

$$X = f(q)$$

tel que : $q = [q_1, q_2, q_3, q_4, q_5, q_6, q_7]^T$et $X = [x, y, z, \theta, \phi, \varphi]^T$.

Le problème de la géométrique directe est très facile à résoudre pour les manipulateurs en série, tandis que le problème de la géométrique inverse est beaucoup plus difficile car il conduit à résoudre des systèmes d'équations polynomiales.

Modélisation cinématique

D'après que le modèle géométrique met une relation entre les positions de l'effecteur final et les articulations, le modèle cinématique représente la relation entre les vitesses opérationnelles (la vitesse de l'effecteur final) et les vitesses articulaires (la vitesse des articulations). En appliquant la méthode décrite au chapitre 1, le modèle cinématique peut être comme ceci :

$$\dot{x} = J(q).\dot{q}$$

qui relie le vecteur vitesse joint $\dot{q}$ au vecteur vitesse cartésien effecteur final $\dot{x}$ via la matrice jacobienne (6×7) $J(q)$. La matrice Jacobienne obtenu par la méthode :

$$J(q) = \left[\frac{\partial X}{\partial q}\right]$$

Selon le modèle géométrique et les lois du calcul, la matrice jacobienne pourrait être calculée comme celle-ci :

$$ J(q) = \begin{bmatrix} \frac{dx}{dq_1} & \frac{dx}{dq_2} & \frac{dx}{dq_3} & \frac{dx}{dq_4} & \frac{dx}{dq_5} & \frac{dx}{dq_6} & \frac{dx}{dq_7} \\ \frac{dy}{dq_1} & \frac{dy}{dq_2} & \frac{dy}{dq_3} & \frac{dy}{dq_4} & \frac{dy}{dq_5} & \frac{dy}{dq_6} & \frac{dy}{dq_7} \\ \frac{dz}{dq_1} & \frac{dz}{dq_2} & \frac{dz}{dq_3} & \frac{dz}{dq_4} & \frac{dz}{dq_5} & \frac{dz}{dq_6} & \frac{dz}{dq_7} \\ \frac{d\theta}{dq_1} & \frac{d\theta}{dq_2} & \frac{d\theta}{dq_3} & \frac{d\theta}{dq_4} & \frac{d\theta}{dq_5} & \frac{d\theta}{dq_6} & \frac{d\theta}{dq_7} \\ \frac{d\phi}{dq_1} & \frac{d\phi}{dq_2} & \frac{d\phi}{dq_3} & \frac{d\phi}{dq_4} & \frac{d\phi}{dq_5} & \frac{d\phi}{dq_6} & \frac{d\phi}{dq_7} \\ \frac{d\varphi}{dq_1} & \frac{d\varphi}{dq_2} & \frac{d\varphi}{dq_3} & \frac{d\varphi}{dq_4} & \frac{d\varphi}{dq_5} & \frac{d\varphi}{dq_6} & \frac{d\varphi}{dq_7} \end{bmatrix} $$

La matrice jacobienne demande beaucoup de calcul entre la dérivation et la simplification, c'est pourquoi nous avons utilisé Matlab pour cela, malgré la simplification, la matrice reste très large, c'est pourquoi on peut pas la mettre ici.

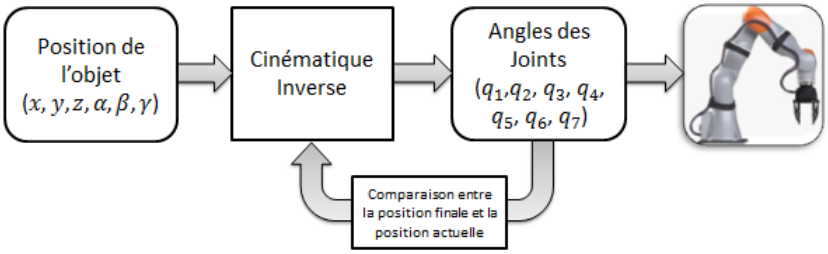


FIGURE 3.14 – Schéma qui montre le travail de la dernière partie de ce projet "le robot".

Modélisation Cinématique Inverse

Ce qui nous concerne par cette approche, c'est le modèle cinématique inverse, le modèle cinématique inverse donne les vitesses des articulations en fonction de vitesse de l'effecteur final. Dans le cas général, la forme de la cinématique inverse pourrait être comme ceci :

$$\dot{q} = J^{-1}(q).\dot{x}$$

Cette formule est valable lorsque la matrice jacobienne est inversible. Comme nous l'avons vu, le robot kuka est redondant, donc la matrice jacobienne n'est pas inversible (6x7). Il existe une méthode pour ce cas, le pseudo-inverse de la matrice jacobienne.

le pseudo-inverse pourrait être calculé avec cette relation :

$$J^{+} = J^{T}(JJ^{T})^{-1}$$

à la fin l'équation pourrait s'écrire ainsi :

$$\dot{q} = J^{+}(q)\dot{X}$$

Heureusement, le logiciel V-rep permet de calculer toutes ces matrices, une fois que nous avons défini la position de l'objet, le robot va directement à sa position, après quand nous l'avons codé bien sûr, le travail principal de cette partie montré dans la figure 3.14.

Configuration de IK (cinématique inverse) sur V-rep

Le problème de la cinématique inverse est une transformation des coordonnées de l'espace des tâches en coordonnées de l'espace des articulations. Pour un manipulateur en série, par exemple, le problème serait de trouver la valeur de toutes les articulations dans le manipulateur étant donné la position (et / ou l'orientation) finale de l'effecteur.

V-REP utilise des groupes (Fig. 3.15) pour résoudre les tâches de cinématique inverse, un groupe contient un ou plusieurs éléments. Le groupe définit les propriétés de résolution globales (comme l'algorithme de résolution à utiliser, etc.) pour un ou plusieurs éléments (Fig. 3.15(a)).

Les éléments IK spécifient des chaines cinématiques simples. Une chaîne cinématique est une liaison contenant une base, plusieurs maillons, plusieurs articulations, une pointe "Tip" et une cible "Target" (Fig. 3.15).

Sur V-rep nous avons créé deux points, l'un prend le nom "Tip " et l'autre "Target" ", le Tip représente la position actuelle de l'effecteur final, et le " Target " représente la position cible, si l'algorithme fonctionne bien, le "Tip" doit aller à la position "Target", comme indiqué sur la figure 3.16.

3.5 Matériel utilisé

Tout ce travail a été effectué par l'ordinateur portable Dell i5, sans GPU, fonctionne sur Windows 10, la préparation et le test de l'ensemble de données effectué par la distribution Anaconda qui utilise python 3.8.

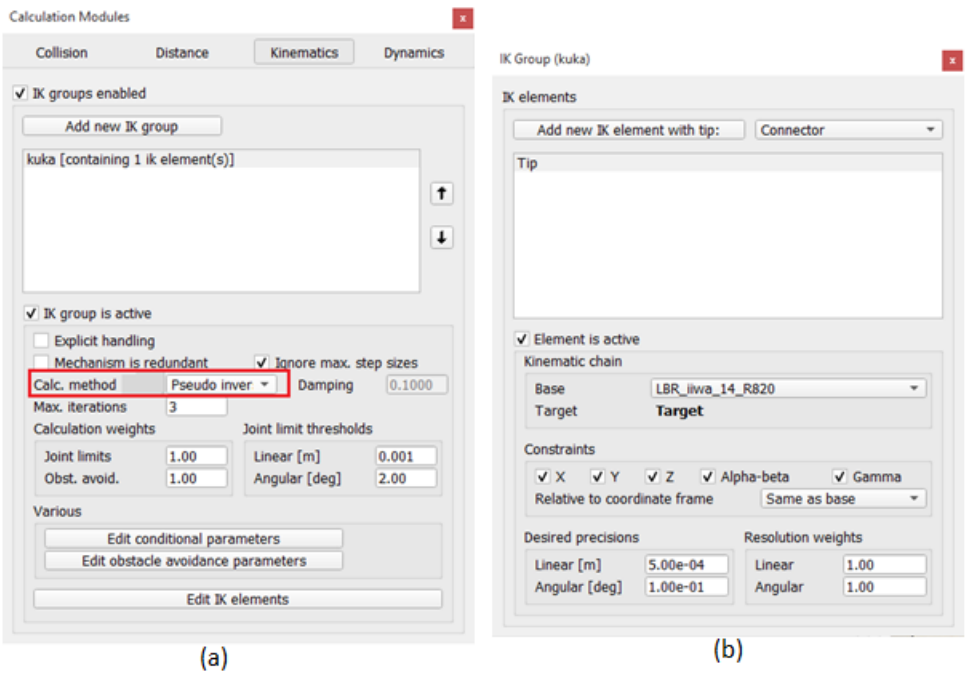


FIGURE 3.15 – Définition de la configuration cinématique inverse pour kuka dans V-rep, (a) représente le groupe IK, le rectangle rouge est la méthode utilisée pour le calcul "Pseudo-Inverse ", (b) représente les éléments IK, et la configuration entre eux.

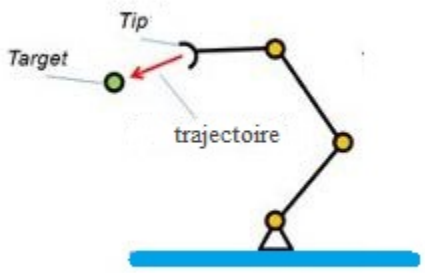


FIGURE 3.16 – Le point "Tip" doit arriver à la position du cible "Target" en suivant la trajectoire.

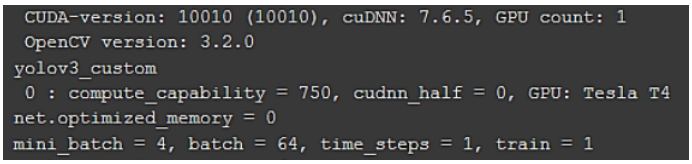


FIGURE 3.17 – les bibliothèques installés et le GPU utilisé dans Colab.

L'apprentissage s'exécute sur le Cloud (Google Colab), qui est livré avec le GPU Tesla T4, et les bibliothèques nécessaires telles que CUDA == 7.6.5 et Opencv == 3.2.0, le reste est présenté dans la figure 3.17.

Pour la simulation de robot, nous avons utilisé la dernière version de V-rep, qui est la version "CoppeliaSim Edu 4.0", la simulation et la préparation fonctionnent sur le même Hardware.

3.6 Conclusion

la méthodologie de ce mémoire a été expliquée dans ce chapitre, ainsi que les étapes de travail. En commençant par la partie vision, nous avons fait une comparaison entre les deux détecteurs d'objets d'une platine, en fonction des résultats de détection entre précision et sensibilité, nous avons choisi YOLOv3 comme le modèle le plus efficace, afin de l'entrainer sur deux jeux de données avec des paramètres différents .

Pour la Raspberry, nous avons nommé les caractéristiques de cette petite carte, et pourquoi elle est parfaite pour ce projet, et finalement nous l'avons préparée pour l'expérience.

La dernière étape était le robot, nous avons donné le modèle géométrique et cinématique du robot choisi. La plupart du travail a été fait en V-rep, nous avons expliqué comment nous configurons vrep pour faire la cinématique inverse.

Chapitre 4

Résultats et discussions

4.1 Introduction

Dans ce chapitre, résultat de cette mémoire sera montré et discuté, en commençant par la partie vision par ordinateur, le modèle obtenu par la formation sur des ensemble de données différentes sera testé et évalué, en utilisant une méthode spéciale pour calculer le précision moyenne qui décrit la valeur de détecteur d'objet sur toutes les classes.

Après cela, nous allons développer le modèle de sortie d'une détection classique à une détection de saisie avec certains outils de vision par ordinateur, une fois le modèle est prêt, nous allons faire la détection sur le Raspberry, considérant que le Raspberry est préparé pour l'expérience.

Enfin, la partie robotique sera ajoutée en dernier, les performances de la cinématique inverse seront discutées, également une implémentation du robot choisi sera appliquée avec une mini chaine d'approvisionnement, tout ce travail se fera en simulateur V-rep.

4.2 Résultas de vision par ordianteur

4.2.1 Performance de détection d'objet

4.2.1.1 Entrainement et validation

La stratégie de l'apprentissage consiste à former un modèle sur le jeu d'apprentissage de détection d'objets Open Images pour 100 00 itérations sur Google Colab, avec les paramètres décrits dans la section 3.2. Chaque session a durée environ 12 heures (le temps alloué pour une session sur Google Colab). la formation ne s'est pas très bien déroulée, nous avions du problème avec Colab, à chaque fois que nous devions télécharger les images à la VM, car le stockage de la VM est temporaire. Toutes les 100 itérations, les poids sont enregistrés sur Google Drive pour ne pas perdre la progression.

Chaque itération, la boite englobante, la confiance de l'objet et les pertes de classification sont calculées comme une somme de perte globale. La perte a commencé avec une grande valeur puis elle diminue par itérations à une valeur minimale dépendant du temps d'apprentissage et de la qualité de l'ensemble de données (voir la figure 4.1).

De toute évidence, la fonction de perte globale ne représente pas avec précision les performances finales d'un algorithme. Il se peut que l'erreur engendrée par la localisation des objets soit sur pondérée par rapport aux autres pertes. Cela semble approprié lorsque nous rappelons que les détections ne sont reconnues comme de vrais positifs "TP" que si elles dépassent un seuil IOU "threshold" fixe avec une vérité terrain, tandis que les scores de confiance et de classification des objets jouent un rôle important dans le filtrage des détections et la génération de précision.

4.2.1.2 Test et Evaluation

Afin de tester le détecteur d'objet, différentes approches ont été envisagées. Chaque fois qu'une image de test traverse le réseau, une nouvelle image est créée contenant les cadres de délimitation des objets détectés,

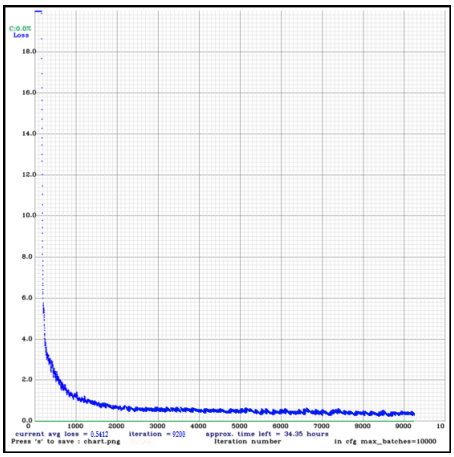


FIGURE 4.1 – Optimisation de la perte de l'entrainent de YOLOv3, avec Adam pour 10000 itérations, la moyenne des perte en 9100 itérations est égale à 0.5412, le temps restant est 34 heurs.

	Remote control	Screwdriver	Toy	Hammer	Pen	mAP
précision	75.51%	63.33%	55.56%	73.47%	64.75%	66.52 %

TABLE 4.1 – Précision moyenne "Ap" pour chaque classe d'objets dans les différentes catégories mesurées sur l'ensemble de validation avec Darknet formé sur l'Open Images.

la classe prédite par le réseau et la probabilité (ou confiance, en%) avec laquelle l'objet détecté a été classé dans cette classe. En parlant du test, nous avons choisi des échantillons aléatoires pour tester les performance du modèle à chaque classe, les résultats sont montrés dans la figure 4.2.

Le tableau 4.1 montre les performances finales du modèle sur l'ensemble de test de Open Images. Ici, nous voyons chaque précision moyenne de chaque classe, et dans la dernière ligne, nous voyons mAP (Mean Averge Precison) la précision moyenne de toutes les classes.

4.2.2 Performance de detection de prise

4.2.2.1 Sur Corenll Grasp Dataset

Les mêmes paramètres sont également utilisés pour cet ensemble de données, avec Adam comme une méthode d'optimisation, les valeurs de perte ont diminué de plus de 10000 itérations, mais nous sommes confrontés à un problème que nous n'avons pas pu résoudre, la valeur de perte coincée entre 3,8 et 4,2 même après 5000 itérations. Après plus de trois essais, le problème reste le même. Avec les limites de formation sur Colab, nous ne pouvions pas comprendre ce qui se passe, c'est pourquoi nous avons abandonné cette partie et nous avons cherché une autre solution pour rendre ce travail complet.

Détection de prise par la région d'intérêt

Tout d'abord, l'image a été passée par filtre de flou, il est fait pour réduire l'impact du bruit et toute netteté indésirable dans une image. Il existe plusieurs méthodes pour flouter les images, même si les plus remarquables sont peut-être le flou gaussien, le flou médian et le flou de boîte.

Lorsque l'image a été floutée, la différence absolue entre l'image actuelle et l'image d'arrière-plan, qui a également été réduite et floue, est calculée. Le calcul est effectué par élément. L'image différentielle résultante est ensuite convertie en une image binaire en évaluant quelles valeurs de pixel sont au-dessus d'un seuil établi. Un exemple d'image différentielle et l'image binaire correspondante peuvent être vus sur la figure 4.3. Une fois l'image binaire créée, les régions d'intérêt peuvent être trouvées en évaluant les pixels interconnectés. Deux pixels voisins sont considérés comme appartenant à la même région s'ils ont la même valeur.

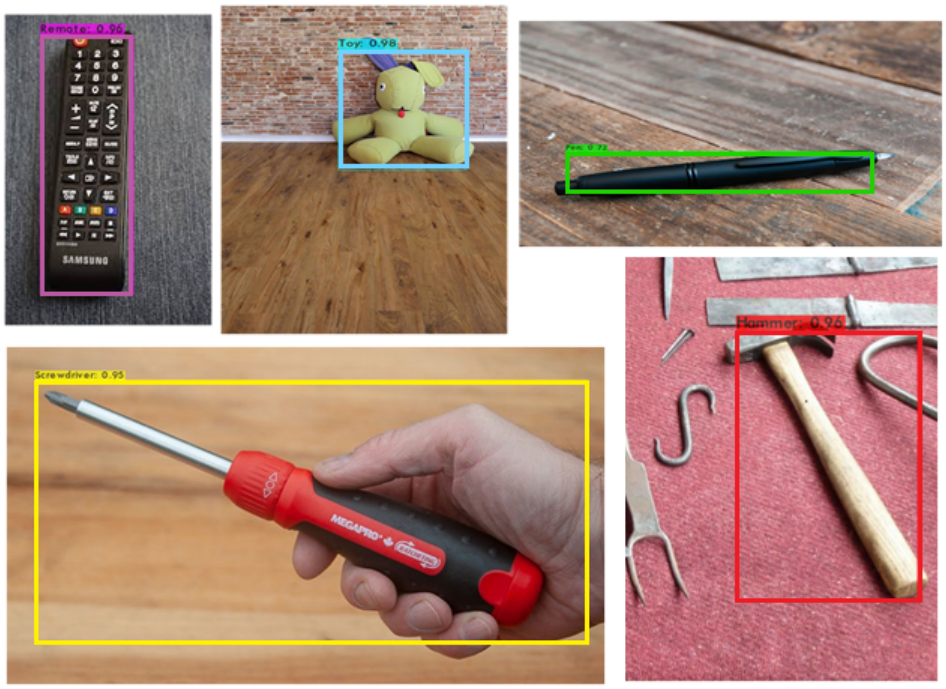


FIGURE 4.2 – Des échantillons aléatoires d'images prises sur Internet avec des détections (au-dessus d'un niveau de confiance de 0,5) générées par Darknet et yolov3.

La détection de région d'intérêt basé sur la gamme de couleurs a été la méthode choisie pour localiser la forme complète des objets dans l'image. Compte tenu de la différence entre les couleurs du même objet, cela semble bien fonctionner dans de nombreux cas. Un exemple de ceci peut être vu dans la figure 4.3. La méthode, cependant, n'est pas sans défauts.

En utilisant opencv, nous détectons la région à partir de l'étape de seuillage qui est présentée comme des pixels blancs, la région est un ensemble de pixels qui sont interconnectés, parfois nous aurons plus d'une région avec des pixels interconnectés, mais nous les négligerons, nous choisirons la plus grande région (contient les pixels les plus interconnectés) comme région d'intérêt. Cette région sera marquée avec un rectangle (Fig. 4.3(b)), nous avons programmé ce rectangle pour suivre la forme de la région dans n'importe quelle direction, car c'est notre but, qui est de détecter la forme complète de l'objet afin que nous puissions le saisir correctement.

Un problème survent si l'éclairage d'une scène ou si l'exposition au capteur d'image de l'appareil photo change. Cela produira une différence significative entre les nouvelles images et les images sur lesquelles nous avons définie le seuillage. Dans le pire des cas, il pourrait rendre la méthode suggérée inutilisable en comptant des pixels de détection aléatoires comme région d'intérêt (Fig .4.4(b)).

Enfin, nous ne pouvons pas détecter plus d'un objet dans une image, car l'algorithme doit traiter une image pour une seule région, si nous avons plus d'une région, l'algorithme va perturber et détecter les mauvaises régions comme des régions d'intérêt, cela rend la détection multi-objets non considérable (Fig. 4.4(a)).

4.3 Application

La formation du modèle de détection d'objets (yolov3) a été réalisée sur Colab, avec le Darknet framework, comme résultat de l'apprentissage, le modèle a été enregistré sur Google drive est un modèle de Darknet, suite à la procédure de travail avec ce modèle, nous avons besoin du fichier de configuration de formation, il semble que YOLO a également besoin du fichier de configuration pour la détection, comme l'entrainement, mais avec quelques modifications.

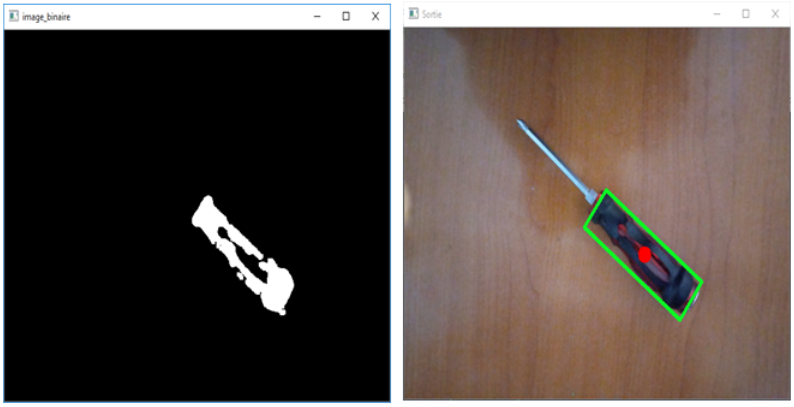


FIGURE 4.3 – (a) Sortie de "thresholding" (image binaire), les pixels blancs représentent la région d'intérêt. (b) Région marquée avec un rectangle pivoté sur l'image originale.

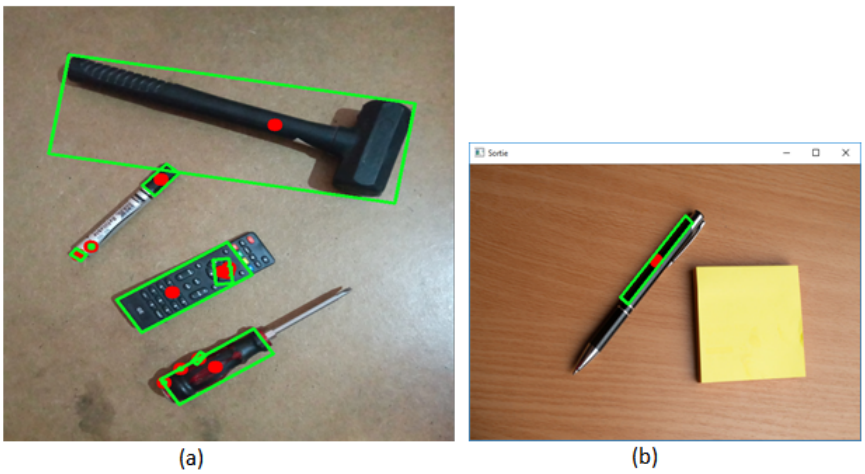


FIGURE 4.4 – Les défauts de détection de région d'intérêt par rapport à la gamme de couleurs.

FIGURE 4.5 – Résultat de détection d'objet avec YOLOv3 sur PC Dell avec Intel I5 3Go

Les derniers poids ont été transférés vers le Raspberry avec les fichiers nécessaires, la bonne chose à propos de la sauvegarde du modèle est de l'utiliser pour la détection sans besoin d'installer une seule bibliothèque d'apprentissage en profondeur, pour détecter avec le modèle yolov3 juste vous avez besoin une version OpenCV de 3.x ou supérieure, parce qu'il contient lui-même un module de deep learning. Comme le module est un module de deep learning, cela signifie que la détection même avec opencv prendra un certain temps. Dans un processeur Intel I5, la détection prend environ 2s en moyenne (Fig. 4.5), même avec ce temps elle ne peut pas être comptée comme une détection en temps réel, étant donné que le Raspberry n'est pas puissant comme I5, la détection pourrait prendre beaucoup plus de temps que dans Intel I5.

4.3.1 Sur Image

D'abord nous n'utilisons que la détection d'objet normale, puis nous ajouterons la détection de la région d'intérêt pour comparer les résultats, sachant que la détection de région en soi n'est pas une opération complexe et n'aura pas besoin d'une puissance de calcul comme le deep learning.

Nous avons testé la détection d'objet avec yolov3, le résultat est plutôt bon comme le premier test, mais nous avons un problème avec le temps de traitement, qui prend plus de 28s pour une seule image (voir la figure 4.6),

Comme nous le savons, la détection classique nous donne juste une boite approximative d'un objet, qui ne répond pas à nos besoins, et ne décrit pas la forme complète de cet objet, spécialement l'angle de rotation, comme nous le voyons dans la figure, la boite n'a pas tourné avec l'objet même s'il est tourné avec un petit angle, c'est un défi pour n'importe quel robot pour saisir un objet basé sur cette méthode.

Avec notre méthode pour détecter la forme complète d'un objet basé sur le "Thresholding" et la soustraction de background, nous pouvons détecter la forme exactement, nous pouvons voir la comparaison entre les deux méthodes sur la figure 4.7. La détection avec "Thresholding" nous a donné la localisation sans aucun signe de classe, donc si nous avons besoin de saisir un objet sans avoir besoin de savoir quel est, nous pouvons utiliser que la détection avec "Thresholding".

Notre méthode collecte entre la détection d'objet avec apprentissage en profondeur et la détection avec seuillage, pour obtenir la meilleure valeur de chaque méthode, nous exécutons la détection d'objet et le seuillage dans le même temps, comme nous le voyons dans la figure 4.8 , la catégorie et la forme complète de l'objet sont détectées.

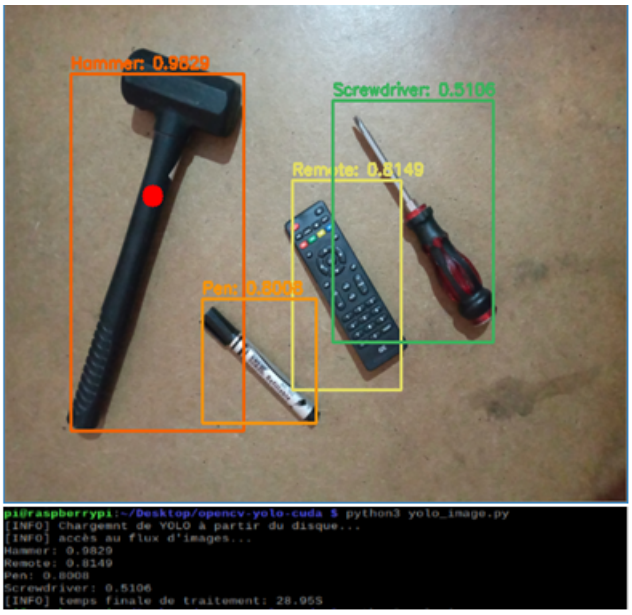


FIGURE 4.6 – Détection d'objet avec YOLOv3 sur un Raspberry Pi 3 Modèle B 1Go

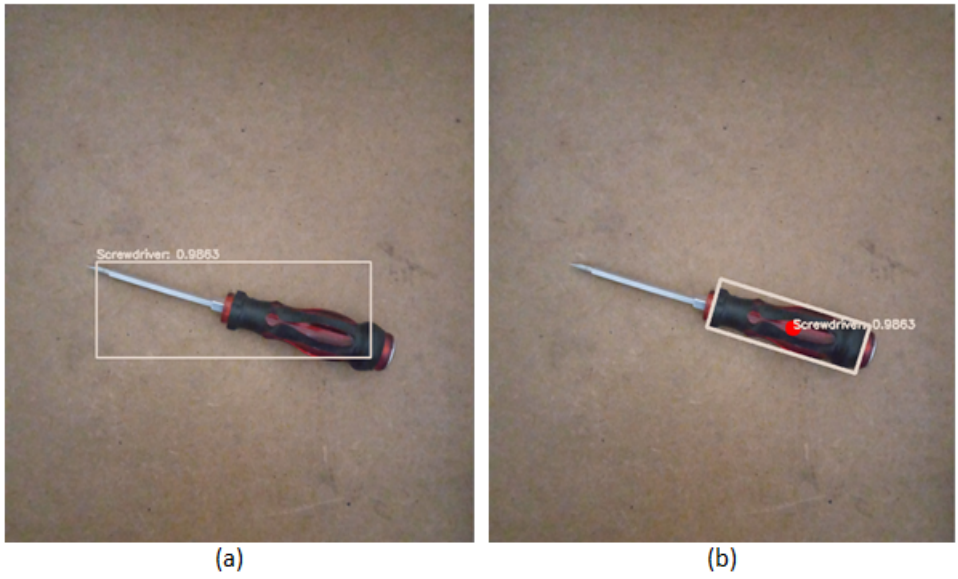


FIGURE 4.7 – (a) Résultat de détection d'objet avec YOLOv3 (b) Résultat de détection d'objet avec notre proposition (YOLOv3 avec la détection de région d'intérêt)

FIGURE 4.8 – Résultats de détection d'objet avec YOLOv3 et la région d'intérêt pour 4 catégories.

A la suite de cette opération, les coordonnées du centre (x, y) représentée en point rouge dans l'image de sortie et l'angle de la boite tournée (thêta) sont extraites pour les transférer au robot qui va appliquer la cinématique inverse pour se déplacer vers l'objet détecté.

4.3.2 Sur Vidéo

Le temps de calcul pour la détection d'une seule image est supérieur à 28s pour le Raspberry, c'est normal en raison de la capacité du processeur intégré à Raspberry, cela signifie que la détection en temps réel est absolument impossible, ce qui nous reste est de faire la détection avec un autre disposition, sur le PC Dell I5 nous avons traiter une image avec 2s, ce temps est court mais nous pouvons le gérer, disons que l'objet est fixe, et que le robot lui-même prendra un certain temps pour déplacer un objet.

Si nous allons travailler en temps réel, nous devrons limiter les FPS de la caméra, sinon nous programmerons la Raspberry pour faire la détection s'il y a un objet dans la scène, s'il n'y en a pas, la détection ne se produira pas.

Nous avons pris le 1er choix, avec un smartphone (androïde ou Iphone) nous avons téléchargé l'application "IP Webcam", où cette application nous donne l'accès pour utiliser l'appareil photo du smartphone, et a une option pour changer le FPS de la caméra.

Avec 2s par image, nous avons choisi 0,2 FPS (5 secondes pour traiter une image), la vidéo s'est lancée lentement, même avec une vitesse comme celle-ci (1/5 FPS) le pc ne pouvait pas la supporter, après une seconde il a commencé à faire des bruits, en plus de chauffer, cela affectera bien sûr les résultats de détection, comme nous l'avons vu sur la figure 4.9, l'algorithme détecte le marteau comme un stylo.

La catégorie d'objet n'est pas la seule à être affectée lorsque que nous avons mis la détection en temps réel, comme on peut le voir sur la figure 4.10, la région détectée n'est pas fixe (doit être fixe) pendant la vidéo, peut-être à cause de la variation de luminance , ou peut-être à cause des vibrations de la caméra afin que tous ces problèmes puissent être résolus avec une caméra fixe, et l'expérience est également réalisée dans une pièce fermée.

D'après cette expérience, nous avons obtenu la catégorie d'objet qui se trouve dans la scène, et à la place d'une boite englobante (résultat de la détection d'objet classique) nous avons obtenu un rectangle qui représente la prise, comme la position de la prise (x, y) est le centre de ce rectangle (coloré en rouge) et

FIGURE 4.9 – Mauvais exemple de détection d'objet avec notre modèle, du fait que la région est bien détectée mais la catégorie n'est pas juste.

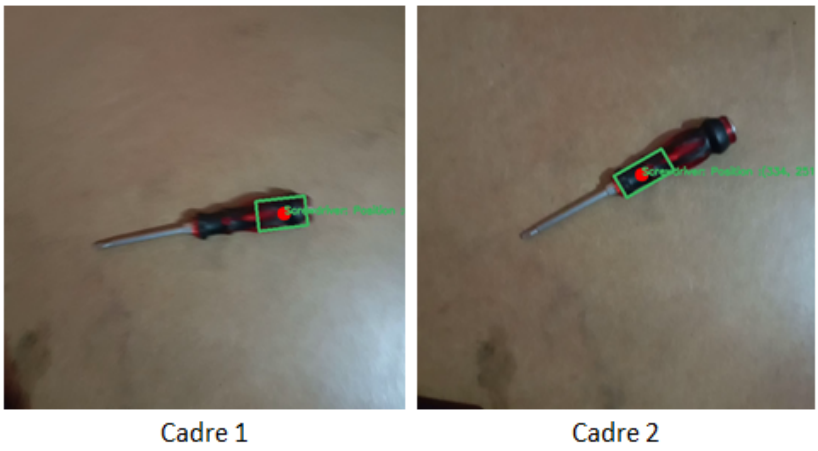


FIGURE 4.10 – Variation de la région détectée (encadrée en vert) en fonction de l'image (FPS) dans une vidéo.

FIGURE 4.11 – Résultat de détection en temps réel avec notre modèle développé, la sortie représente la position et l'orientation de prise et aussi l'ouverture du pince.

l'angle de prise (thêta) est l'angle de rotation, et en plus la distance d'ouverture de la pince est la largeur de cette boite (voir la figure 4.11).

4.4 Simulation sur V-rep

4.4.1 Performance de IK

Le problème de la cinématique inverse peut être vu comme celui de trouver les valeurs d'articulation correspondant à une position et / ou une orientation spécifiques d'un élément corporel donné (généralement l'effecteur terminal). V-REP utilise des groupes pour résoudre les tâches de cinématique inverse.

Comme nous l'avons montré dans le dernier chapitre, la méthode utilisée pour trouver les coordonnées des articulations à partir de la position et de l'orientation d'un point dans l'espace est le Pseudo-Jacobien, nous avons programmé V-rep pour trouver la solution avec cette méthode, en V-rep nous avons créé deux points, l'un prend le nom "Tip '" et l'autre "Target" ", le Tip représente la position actuelle de l'effecteur, et le" Target "" représente la position cible (position d'objet par example), si l'algorithme fonctionne bien, le "Tip" doit aller à la position du "Target".

Après la configuration du robot sur V-rep, si nous exécutons la simulation, le " Tip " qui représente le point final du robot (l'effecteur final) se déplacera vers la position d'un point qui est défini comme "Target". La trajectoire de mouvement respectera les contraintes des articulations du robot, si le point ou la "Cible" en position plus loin et que le robot ne peut l'atteindre à cause des contraintes sur les articulations, le robot aura une singularité et va trembler de façon folle. Pour éviter ce mauvais comportement, nous mettons le "Target" à une position accessible par le robot, et nous exécutons la simulation, le résultat apparait dans la figure 4.12.

le robot met très peu de temps pour atteindre la position de l'objet "Target", moins de 0,5 s, il est très rapide, voir la figure 4.13. De toute façon, on ne peut pas appliquer ça en réalité avec cette vitesse. La courbe a déviée vers une valeur minimale à 0,1 s, c'était à cause des contraintes sur les joints.

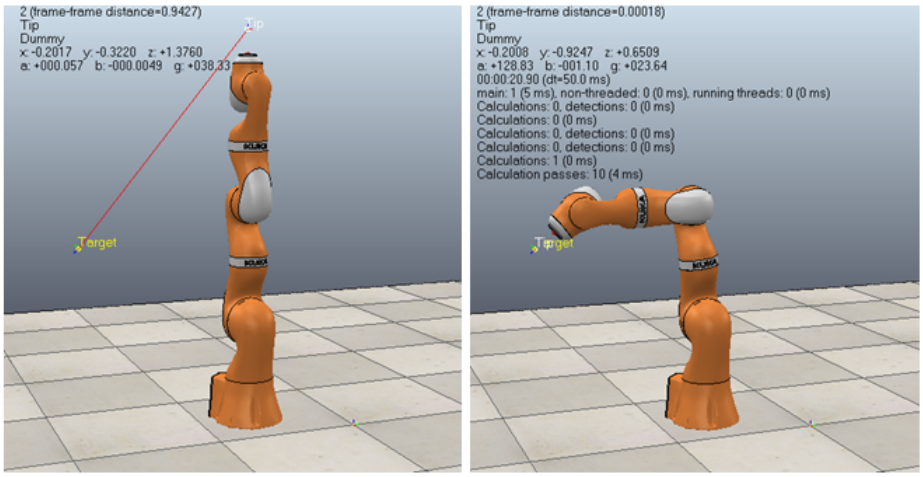


FIGURE 4.12 – (a) Le robot est configuré pour suivre le trajectoire en rouge (b) le robot a atteint la position de la cible "Target" suivant la trajectoire.

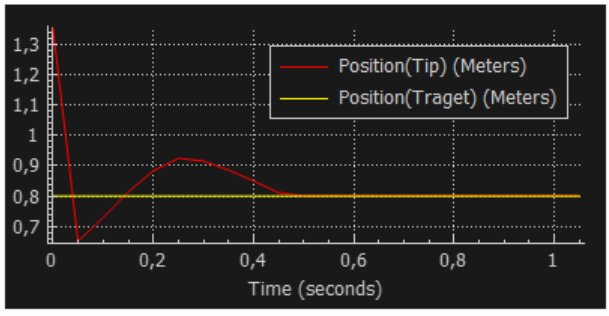


FIGURE 4.13 – La variation du deux points "Tip" et "Target" par rapport au temps.

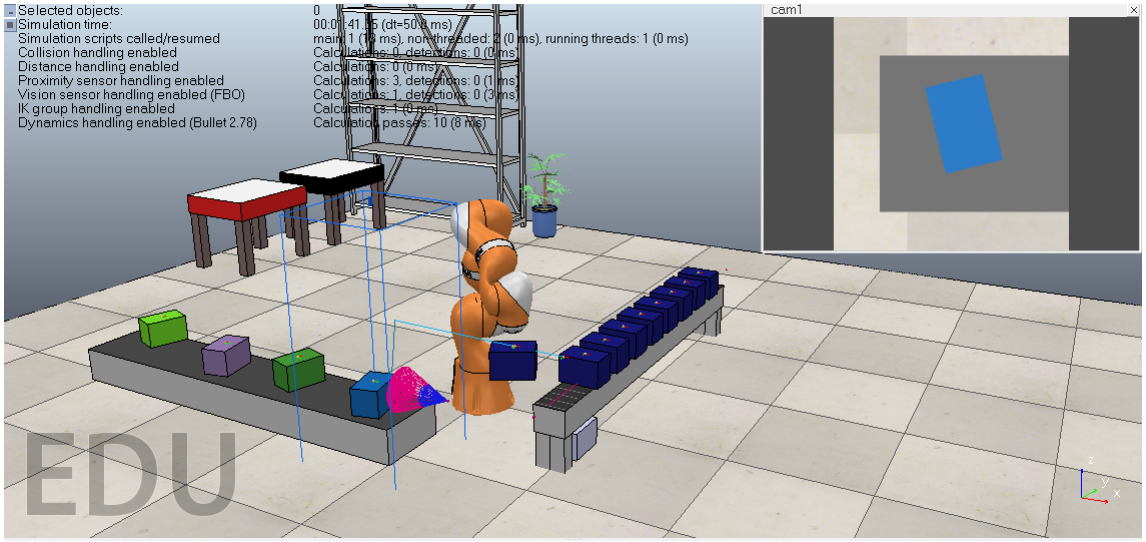


FIGURE 4.14 – Simulation d'une mini chaine d'approvisionnement avec robot Kuka sur V-rep.

4.4.2 Application de IK

Le champ d'application de la cinématique inverse est très large, en particulier dans les usines où de nombreux robots travaillent ensemble de manière très ordinaire, et tous ces robots sont automatisés pour faire un travail spécial, notre objectif est de mettre le robot dans un environnement avec d'autres machines, et nous programmons toutes ces machines pour suivre certains travaux.

Dans vrep, nous créons une mini chaîne d'approvisionnement pour notre robot, son travail est de déplacer un objet du tapis roulant vers une autre table de la même scène, en utilisant la cinématique inverse pour suivre le chemin que nous avons créé pour lui, pour faire des choses plus simple nous dépendions de vrep pour détecter la position et l'orientation de l'objet.

Nous avons mis la simulation similaire à ce que nous avons fait dans la partie de vision, nous avons utilisé un objet avec une couleur différente, et nous avons programmé le robot pour détecter une seule couleur parmi les autres, le scène a été montré dans la figure 4.14. Nous avons mis un connecteur reprenste l'éffecteur fnale, cette connecteur peut a la meme positon et l'orientation de le "Tip", notre but est cibler le " Target" sans avoir une pertubation a partie du pince, lorsque la pnce et longue, et nous rendre gennant.

Le résultat est très satisfaisant, nous avons crée à un environnement virtuel, nous pouvons éviter toutes les erreurs qui peuvent être produites en application réelle.

4.5 Discusion

4.5.1 Pourqoui nous l'avons utilisé

4.5.1.1 Darknet framewrok

Darknet est un framework de réseau neuronal open source écrit en C et CUDA. Il est rapide, facile à installer et prend en charge le calcul CPU et GPU. ... De plus, le framework peut être utilisé pour faire fonctionner les réseaux de neurones à l'envers dans une fonctionnalité nommée à juste titre Nightmare.

4.5.1.2 Python

La raison d'utiliser Python est assez simple. Au début de ce projet, nous avions avait peu d'expérience avec le deep learning et les réseaux neuronaux, et de grandes bibliothèques d'apprentissage automatique de haut niveau, Keras en particulier, existent pour Python, et non pour C ++. De plus, la communauté en ligne pour l'apprentissage automatique qui utilise Python est vraiment très bonne. Un certain nombre de bons tutoriels et guides existent.

4.5.2 Méthode de détection de prise

La détection de prise était un vraie Chalenge pour nous, au début, nous n'étions pas sûr avec quelle méthode nous allions travailler avec et quelle méthode va nous donner de bons résultats, c'est pourquoi nous avons essayé certaines méthodes et nous avons fini par choisir la méthode qui nous avons utilisé dans ce mémoire.

La base de données CGD, même si elle nous a donné des mauvais résultats, elle nous a pris beaucoup de temps de travail, parce que nous l'avons vu comme une méthode professionnelle et a déjà fait partie de notre recherche bibliographique, mais personne n'avait essayé de l'entrainer avec un détecteur d'un étage (YOLO & SSD), nous avons pensé que nous allions faire une meilleure contribution dans le domaine.

D'abord nous avons fait l'entrainement sur CGD avec YOLOv3 en utilisant le Darknet framework, mais l'entrainement n'allait pas comme nous avions voulu, la perte a stoppé de minimiser dans une grande valeur, après certains essais, nous l'avions abandonné et nous avons essayé l'entrainement sur un autre framework. Toujours YOLOv3 avec la bibliothèque de deep learning la plus populaire Tensorflow et keras, en utilisant python, la formation n'était pas facile comme nous l'avons pensé, mais nous avons réussi à la faire marcher, les résultats ont cependant étaient plus mauvais que la première méthode.

Nous avons pensé que le YOLOv3 n'était pas compatible avec cette base de données, c'est pourquoi nous avons décidé d'utiliser le SSD. La préparation nous a pris beaucoup de temps plus que les premières, parce que nous essayions une autre architecture complètement différente à ce que nous avons vu, après un certain nombre d'essais, nous avons réussi à lancer l'entrainement. Au contraire des expériences précédentes nous avons obtenu un meilleur résultat (la perte est autour de 0), mais il y a aucune prise détectée même sur les images qui ont fait partie de l'entrainement, cela nous a conduit a abandonner tous qui concernent cette base de données et a essayer avec une autre méthode complément différente.

cela nous a conduit de distingué que la base de données CGD n'est pas compatible avec les détecteurs d'un étage, car la détection d'un étage n'extrait que des caractéristiques, contrairement au détecteur d'un deux étage, ils vont extraire toute l'instance de l'objet, à la fin si nous voulons travailler avec la base de données CGD, nous avons besoin d'un détecteur comme le Faster RCNN.

4.5.3 Simulation sur vrep

La raison d'utiliser vrep est assez simple. Au début de ce projet, nous comptons utiliser un vrai robot (le robot dans LAT ou au pire nous construirons un robot à partir d'une imprimante 3D) dans un premier temps, mais pour des raisons concernant la pandémie (confinement des COVID -19), nous n'avons pas pu accéder au laboratoire. Dans ce cas, vrep est apparu comme une plateforme virtuelle notamment en robotique. Notre objectif était de terminer l'ensemble du projet en vrep, même le traitement d'image. Malheureusement, le temps alloué au projet n'était pas suffisant pour le faire.

4.6 Conclusion

Les résultats de ce mémoire ont été montrés et discutés, le modèle de détection d'objet (yolov3) sur OID a été testé et évalué, le test a été fait sur la même base de données, les résultats ont montré que le modèle est bien entraîné. Après l'échec de la formation à la base de données CGD, nous avons utilisé YOLOv3 pour détecter la position de prise d'un objet sur une image. Le Raspberry ne peut pas supporter la détection avec ce modèle (yolov3), surtout en temps réel, nous avons donc fait la détection sur un PC.

Enfin, la simulation en vrep a été montrée, les performances de la cinématique inverse ont été montrées, et nous avons donné un exemple pratique de mise en œuvre de la cinématique inverse.

Conclusion générale

Nous avons présenté une nouvelle méthode de détection pour la préhension robotique en vue RGB, basée sur la détection d'objets et le traitement d'image (Thresholding).

Dans les travaux de détection de prise précédents, il est important de souligner que les algorithmes ne détectaient que la position de prise pour un objet inconnu, avec notre méthode proposée, nous avons pu détecter la position de prise et la catégorie d'objets détectés par la caméra.

Travailler avec la base de données CGD nous a pris beaucoup d'investissement en temps et en effort, et cela nous a conduit à comprendre que la détection des prises en elle-même est une tâche complexe et nécessite un détecteur à deux étages comme le Faster RCNN, c'est pourquoi nous avons opté pour un plan B, qui nous a permis de terminer le travail et à réussir ce projet.

Nous avons rencontré plusieurs obstacles lors de la preparation de mémoire en raison de l'épidémie COVID-19 et des mesures de quarantaine pendant la période de recherche et de préparation, et pour cette raison, nous n'avons pas pu terminer la conception du robot, nous avons donc choisi le robot kuka LBR IIWA pour une utilisation dans la technologie de simulation.

La détection d'objets a donné de bons résultats, même lorsque nous avons ajouté la détection de prises, qui n'était qu'une méthode de localisation, sinon, l'expérience d'utilisation des deux méthodes était très excitante, et nous a donné les résultats que nous avions souhaités. Deuxièmement, nous avons essayé le processus de simulation sur Vrep et cette expérience a montré aussi de bons résultats.

Ce travail nous encourage à poursuivre nos recherches dans le domaine de la robotique et de la vision.

Bibliographie

[1] YOLO : Real-Time Object Detection.

[2] Underfitting and Overfitting in Machine Learning, November 2017. Library Catalog : www.geeksforgeeks.org Section : Advanced Computer Subject.

[3] RÃ©seau de neurones artificiels, April 2020. Page Version ID : 169602792.

[4] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. IEEE, 2017.

[5] Md Zahangir Alom, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul AS Awwal, and Vijayan K. Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3) :292, 2019. Publisher : Multidisciplinary Digital Publishing Institute.

[6] Bernard Bayle. Introduction a‘ la Robotique. page 32.

[7] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press, 2017.

[8] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact : A review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 348–353. IEEE, 2000.

[9] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4 : Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv :2004.10934*, 2020.

[10] Shehan Caldera, Alexander Rassau, and Douglas Chai. Review of deep learning methods in robotic grasp detection. *Multimodal Technologies and Interaction*, 2(3) :57, 2018. Publisher : Multidisciplinary Digital Publishing Institute.

[11] Francois Chollet. *Deep Learning mit Python und Keras : Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[12] Fu-Jen Chu, Ruinian Xu, and Patricio A. Vela. Real-world multiobject, multigrasp detection. *IEEE Robotics and Automation Letters*, 3(4) :3355–3362, 2018. Publisher : IEEE.

[13] Etienne Dombre and Wisama Khalil. *Modeling, performance analysis and control of robot manipulators*. Wiley Online Library, 2007.

[14] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2) :303–338, 2010. Publisher : Springer.

[15] Carlos Faria, Flora Ferreira, Wolfram Erlhagen, SÃ©rgio Monteiro, and Estela Bicho. Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance. *Mechanism and Machine Theory*, 121 :317–334, 2018. Publisher : Elsevier.

[16] Taraggy M. Ghanim, Mahmoud I. Khalil, and Hazem M. Abbas. Comparative Study on Deep Convolution Neural Networks DCNN-Based Offline Arabic Handwriting Recognition. *IEEE Access*, 8 :95465–95482, 2020. Conference Name : IEEE Access.

[17] Wisama Khalil and Etienne Dombre. *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.

[18] Sulabh Kumra and Christopher Kanan. Robotic grasp detection using deep convolutional neural networks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 769–776. IEEE, 2017.

[19] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, and Alexander Kolesnikov. The open images dataset v4. *International Journal of Computer Vision*, pages 1–26, 2020. Publisher : Springer.

[20] Claire Labit-Bonis, JÃ©rÃŽme Thomas, FrÃ©dÃ©ric Lerasle, and Francisco Madrigal. Suivi de passagers de bus par apprentissage profond. In *CongrÃšs Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP 2018)*, page 8p, 2018.

[21] Yann LeCun, LÃ©on Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998. Publisher : Ieee.

[22] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5) :705–724, 2015. Publisher : SAGE Publications Sage UK : London, England.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr DollÃ¡r, and C. Lawrence Zitnick. Microsoft coco : Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd : Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[25] Qingkai Lu, Kautilya Chenna, Balakumar Sundaralingam, and Tucker Hermans. Planning multi-fingered grasps as probabilistic inference in a learned deep network. In *Robotics Research*, pages 455–472. Springer, 2020.

[26] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0 : Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv :1703.09312*, 2017.

[27] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322. IEEE, 2015.

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[29] Joseph Redmon and Ali Farhadi. YOLO9000 : better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[30] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767*, 2018.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[32] Adrian Rosebrook. *Practical Python and OpenCV+ Case Studies : An Introductory, Example Driven Guide to Image Processing and Computer Vision*. PyImageSearch, 2016.

[33] Mark W Spong, Seth Hutchinson, and M Vidyasagar. Robot Dynamics and Control. page 303.

[34] Mark W. Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons, 2020.

[35] Ludovic Trottier, Philippe Giguere, and Brahim Chaib-Draa. Convolutional residual network for grasp localization. In *2017 14th Conference on Computer and Robot Vision (CRV)*, pages 168–175. IEEE, 2017.

[36] Jie Wei, Huaping Liu, Gaowei Yan, and Fuchun Sun. Robotic grasping recognition using multi-modal deep extreme learning machine. *Multidimensional Systems and Signal Processing*, 28(3) :817–833, 2017. Publisher : Springer.

[37] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks : an overview and application in radiology. *Insights into Imaging*, 9(4) :611–629, August 2018.

[38] Hanbo Zhang, Xuguang Lan, Site Bai, Xinwen Zhou, Zhiqiang Tian, and Nanning Zheng. Roi-based robotic grasp detection for object overlapping scenes. *arXiv preprint arXiv :1808.10313*, 2018.

[39] Jiahao Zhang, Miao Li, Ying Feng, and Chenguang Yang. Robotic grasp detection based on image processing and random forest. *Multimedia Tools and Applications*, 79(3) :2427–2446, 2020. Publisher : Springer.

[40] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning : A review. *IEEE transactions on neural networks and learning systems*, 30(11) :3212–3232, 2019. Publisher : IEEE.

[41] Xinwen Zhou, Xuguang Lan, Hanbo Zhang, Zhiqiang Tian, Yang Zhang, and Narming Zheng. Fully convolutional grasp detection network with oriented anchor box. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7223–7230. IEEE, 2018.

[42] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years : A survey. *arXiv preprint arXiv :1905.05055*, 2019.