الجمـــهوريـــة الجـــزائريـــة الديـمقـراطيـــــــة الشـــعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي والـبـحـث الـعـلـمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT

كـلـيـــــة التكنولوجيا
FACULTE DE TECHNOLOGIE
قسم الإلكترونيك
Département d'électronique

Mémoire de Master

Domaine : SCIENCES ET TECHNOLOGIES
Filière : TELECOMMUNICATIONS
Option : SYSTEMES DES TELECOMMUNICATIONS

Par :
KORIBA Sarah
BENCHOHRA Razane Sabrina

THEME

La carte Raspberry pi 4 pour des applications en traitement d'images

Dr. ***BIRANE Mouhoub***	*President*
Dr. ***ZITOUNI ABdelkader***	*Examinateur*
Dr. ***REGUIEGUE Mourad***	*Encadrant*

Année Universitaire 2022/2023

Résumé

La carte Raspberry Pi 4 est un mini-ordinateur à faible coût, avec un CPU de 1,5 GHz et une RAM allant de 4 à 8 GB. Il dispose de plusieurs ports avec une architecture assez complète ; cela nous permis de développer différentes applications dans plusieurs domaines. Le Raspberry Pi 4 peut être exploitée pour des applications en traitement d'images en raison de ses capacités de calcul et de ses interfaces de connexion en particulier les caméras. Cette carte peut être utilisée dans diverses applications de traitement d'images telles que la surveillance vidéo, l'analyse d'image, la vision par ordinateur, la reconnaissance faciale, la reconnaissance d'objet, la robotique.

Ce projet de fin de cycle a pour but d'exploité l'algorithme de détection d'objet YOLOv5 afin d'entrainer une nouvelle base de données qui contient 240 images pour obtenir une classification de huit classes des fruits. D'après les résultats obtenus, nous voyons clairement que le modèle est bien entrainé afin de connaître des fruits (détection et classification). Car nous avons obtenu une bonne détection des objets d'intérêt et une précision de classification va de 74% pour la classe Orange à 93% de celle de cerise.

Mots clés : Traitement d'images, Raspberry Pi 4, Python, YOLOv5, CNN.

ملخص

(Raspberry Pi 4) هو جهاز كمبيوتر صغير منخفض التكلفة مع وحدة معالجة مركزية 1.5 جيجاهرتز وذاكرة وصول عشوائي تتراوح من 4 إلى 8 جيجابايت وله عدة مأخذ مهمة هاته الخصائص تمكننا من تطوير تطبيقات مختلفة في عدة مجالات. يمكن استخدام Raspberry Pi 4 لتطبيقات معالجة الصور والفيديوهات نظرًا لقدراته الحسابية الكبيرة ومأخذ الادخال الخاصة بالكاميرات. يمكن استخدام هذه البطاقة في تطبيقات مختلفة لمعالجة الصور مثل المراقبة بالفيديو، تحليل الصور، التعرف على الأشخاص والأشياء وفي ميدان الروبوتيك.

يهدف هذا المشروع إلى استغلال خوارزمية التعرف عن الأشخاص والأشياء YOLOv5 من أجل تدريب قاعدة بيانات جديدة مكونة من 240 صورة لغرض الحصول على تصنيف لثماني فئات من الفاكهة. النتائج التي تم الحصول عليها تؤكد أن النموذج مدرب جيدًا من أجل معرفة أصناف الفواكه (تحديد موقع الفاكهة في الصور مع التصنيف) وقد تمكنا من الوصول الى تحديد جيد لموقع الفواكه في الصور ودقة التصنيف تتراوح من 74٪ لفئة البرتقال إلى 93٪ لفئة الكرز.

الكلمات المفتاحية : معالجة الصور،Raspberry Pi 4، YOLOv5، CNN ، بايتون.

Abstract

The Raspberry Pi 4 is a low-cost mini-computer with a 1.5 GHz CPU and RAM ranging from 4 to 8 GB. It has several ports with a fairly complete architecture; this allowed us to develop different applications in several areas. The Raspberry Pi 4 can be used for images processing applications due to its computational capabilities and connection interfaces especially cameras. This card can be used in various images processing applications such as video surveillance, image analysis, computer vision, facial recognition, object recognition, robotics.

This project aims to exploit the YOLOv5 object detection algorithm in order to train a new database containing 240 images in order to classify eight types of fruits. From the results obtained, we can conclude that our model is well trained in order to know fruits (detection and classification). We have obtained a good detection for the position of fruits in images and a classification accuracy ranges from 74% for the orange class to 93% of that of cherry.

Keywords: Image processing, Raspberry Pi 4, Python, YOLOv5, CNN.

Remerciement

Nous remercions avant tout ALLAH le tout puissant qui m’a offert volonté et la patience pour terminer ce présent travail.

Nous remercions nos parents et toutes les personnes qui nous ont soutenus, encouragées, conseilles.

Nous tenons tout d’abord à exprimer notre gratitude la plus profond de au monsieur **REGUIEGUE Mourad** d’avoir proposé ce sujet pour encadrer avec patience et bonne humeur.

Nous remercions tous les professeurs de département d’électronique et les membres de jury Mr.ZITOUNI Abdelkader et Mr.BIRANE Mouhoub qui nous faisaient l’honneur de jury nous travail.

Nous remercions vivement mes collègues qui m'ont soutenu pendant cette période.

Enfin, Nous remercions tous ceux et toutes celle qui mont aider de près ou de loin dans ce travail chacun à sa manière.

DÉDICACE

Ce travail est dédié à ceux et celles qui
m'ont donné le goût de la recherche, et qui
m'ont soutenu tout au long de mon
parcours. En particulier je le dédié aux
membres de ma famille présents ou partis :

À mes très chers parents que je ne saurai jamais

Remercier de leur longue patience, de leur soutien
À mes chères sœurs et mon frère à qui je souhaite tout le bonheur
du monde

A ma cher amie Razane

A mon oncle que je Remercier pour me aide

A tous les membres de ma famille
A mes amies et mes camarades
A toutes les personnes qui m'ont encouragé et soutenu

Merci à tous…

KORIBA Sara

DÉDICACE

Je dédie cet ouvrage

À ma fleur (maman) qui m'a soutenu et encouragé durant ces années d'études.

A la mémoire de mon cher père.

J'espère que mon père a une place au premier rang au paradis pour me voir diplômé

Je t'aime et tu me manques

A ma partenaire Sarah qui a enduré ma folie je t'aime ma chère.

A mes frères, ma famille et mes amis.

À ma deuxième famille, l'Association du syndrome de Mahdi Down, Ceux qui ont partagé avec moi tous les moments d'émotion lors de la réalisation de ce travail. Ils m'ont chaleureusement supporté et encouragé tout au long de mon parcours.

A tous ceux que j'aime.

BENCHOHRA Razane Sabrina

Sommaire

Chapitre II Traitement d'image avec python

Chapitre III Résultats et interprétations

Abréviations

BSD : Berkeley Software Distribution Licence (Licence de distribution de logiciels Berkeley)

CPU : Central Processing Unit (Unité centrale de traitement)

CSP : Cross-Stage Partial connections (Connexions partielles inter-étages)

CNN : Convolutional Neural Network (Réseau neuronal convolutif)

GPIO : General Purpose Input/Output (Entrée/Sortie à usage général)

GPU : Graphics Processing Unit (Unité de traitement graphique)

HDMI : High definition multimedia interface (interface multimédia haute définition)

IBM : International Business Machines

Numpy : Numerical Python

OS : Operating System

RAM : Random Access Memory (Mémoire vive)

SD : Secure Digital (Sécurisation Numérique)

Open cv : Open Computer Vision

Tkinter : Tool kit Interface

USB : Universal Serial Bus (Autobus de série universel)

WIFI : Wireless Fidelit (Accès sans fil)

YOLO : You Only Live Once

Liste des figures

Chapitre I

Chapitre II

Chapitre III

Liste des tableaux

Chapitre I

Introduction générale

Introduction générale

Le traitement d'images est l'ensemble des méthodes et techniques utilisées pour améliorer le contenu d'une image pour en extraire de l'information, par exemple identifier une séquence de texte ou un chromosome, éviter un obstacle (robotique), ou détecter des zones soumises à l'érosion (télédétection). Les techniques les plus connus utilisées pour le traitement d'images sont : extraction des caractéristiques de l'image, le rehaussement d'image (amélioration de la qualité d'image), le recalage d'image, la segmentation d'image et la classification.

La carte Raspberry Pi 4 est un mini-ordinateur à faible coût, avec un CPU de 1,5 GHz, une RAM de 4 à 8 GB ; on trouve aussi deux ports USB 3.0, et deux autres USB 2.0, deux connecteurs Micro-HDMI et des connectivités sans-fil : Bluetooth 5.0 L.B.E, Wifi. **[1, 2]**

Cette architecture complète, nous permis de développer différentes applications dans plusieurs domaines.

La carte Raspberry Pi 4 peut être exploitée pour des applications en traitement d'images en raison de ses capacités de calcul et de ses interfaces de connexion pour des caméras. Cette carte peut être utilisée dans diverses applications de traitement d'images telles que la surveillance vidéo, l'analyse d'image, la vision par ordinateur, la reconnaissance faciale, la reconnaissance d'objet, la robotique, etc.

L'objectif principal de ce projet est l'implémentation d'une chaine de traitement d'images basée sur l'apprentissage approfondi (Deep Learning) afin de réaliser en premier lieu une détection des objets d'intérêt puis de les classés.

Pour concrétiser cet objectif, nous avons organisé notre mémoire en trois chapitres de la manière suivante :

Dans le premier chapitre, on a mis en avant les notions principales des Raspberry Pi, ont ce basant sur les différents architectures qui existe, et les caractéristiques de la version Pi 4 qui sera exploité dans notre travail. Puis nous avons présenté le langage Python qui est un excellent outil pour développer une application d'ingénierie.

Dans le deuxième chapitre, nous avons présenté une introduction aux concepts liés au domaine du traitement d'images. Les différentes définitions qui y sont développées sont celles des connaissances élémentaires de cette discipline, mais combien même elles sont essentielles pour l'initiation aux traitements approfondis des images.

Dans le troisième chapitre, nous allons illustrer les étapes à suivre en exploitant la carte Raspberry Pi 4 afin appliquer en premier lieu une détection d'objets d'intérêts à partir d'une base de données d'images puis passer à une classification de ces derniers.

On clôturera ce mémoire par une conclusion générale tout en proposant des perspectives pour la continuité de ce travail.

Chapitre I

Généralités sur la carte Raspberry Pi

I.1. Introduction

Dans ce premier chapitre, nous allons présenter les notions générales sur la carte Raspberry pi et leur déférentes types, et on va expliquer la carte Raspberry pi 4 principalement, On va aussi expliquer le langage python et les Bibliothèques utilisées telle que Numpy, Tkinter et Open CV.

I.2.Historique [2]

Dans l'histoire de l'informatique, le Raspberry Pi a clairement laissé sa trace. C'est une histoire incroyable, un mec qui construisait des ordinateurs monocartes pour le plaisir, jusqu'à devenir une fondation reconnue qui vend maintenant plus de 15 millions de machines dans le monde entier.

L'histoire du Raspberry Pi commence en 2006 avec la création des premiers prototypes inspires du BBC Micro. Six ans plus tard, le premier Raspberry Pi est né. Le but principal était d'aider les jeunes à découvrir l'univers des ordinateurs pour un coût modeste (environ 30 €).

Eben Upton est un ingénieur britannique, créateur du Raspberry Pi et de la fondation Raspberry Pi.

IL a étudié la physique et l'ingénierie à l'université de Cambridge avant de travailler pour des entreprises prestigieuses comme Broadcom, Intel et IBM.

L'idée du Raspberry Pi Pendant qu'Eben Upton créait ses prototypes, il s'est rendu compte que l'éducation Britannique souffrait d'un problème les prix élevés des ordinateurs.

De ce fait, les jeunes étudiants ne s'intéressaient pas aux sciences de l'informatique et par la suite, ne choisissaient pas de métier dans ce domaine. Les sociétés anglaises manquaient alors de professionnel informatique.

Eben Upton a tenté d'apporter une solution, en voulant construire un ordinateur 10 fois moins cher..

En mai 2011, de l'université de Cambridge, David Braben a annoncé un nouveau prototype d'ordinateur destiné à stimuler l'enseignement de l'informatique de base dans les écoles La Fondation Raspberry Pi.

I.3. Raspberry Pi

I.3.1.Présentation du Raspberry pi

Le Raspberry Pi est une nano-ordinateur mono carte à processeur ARM. Cet ordinateur, de la taille d'une carte de crédit, est destiné à encourager l'apprentissage de la programmation informatique ; il permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux, notamment Debian, et des logiciels compatibles. Mais il fonction également avec le système d'exploitation Microsoft Windows.

Il est fourni nu, c'est-à-dire la carte mère seule, sans boîtier, alimentation, clavier, souris ni écran, dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération. Néanmoins des « kits » regroupant le « tout en un » sont disponibles à partir de quelques dizaines d'euros seulement. **[3]**

FigureI.1. Le Raspberry pi

Son prix de vente était estimé à 25 $ américains, soit 3500 DA, début mai 2011. Les premiers exemplaires ont été mis en vente le 29 février 2012 pour environ 2 496.14 DA, En septembre 2016, plus de dix millions de Raspberry Pi ont été vendus.

I.3.2. Alimentation de la carte Raspebrry Pi

Le Raspberry Pi s'alimente sous tension unique de 5V laquelle il peut consommer jusqu'à 1.8A selon les taches qu'il exécute. Cette alimentation doit être normalement fournie via le connecteur micro USB placé dans un angle de la carte.

L'utilisation d'un chargeur pour Smartphone équipé d'un câble micro USB qui délivre une tension de 5 volts avec au minimum 1.8A est suffisante pour alimenter notre carte Raspberry**. [4]**

I.3.3.Les modèles de Rasspbery pi

Raspberry Pi est disponible en différentes versions des mises à jour et des améliorations pour l'appareil Raspberry Pi modèle B d'origine, les modèles de base de Raspberry Pi disponibles : **[5]**

I.3.3.1.Raspberry Pi 1

Ce modèle contient 4 types B, B+, A et A+ :

Modèle B :

Le Raspberry Pi 1B est le premier modèle commercialisé de cette longue série. Révolutionnant le monde de la micro-informatique avec une carte mère très accessible, le Raspberry Pi 1B rencontre un grand succès rapidement. **[6]**

- Sortie : 2012
- Ports : 2x USB 2.0, Ethernet
- CPU : 700MHz
- RAM : 512MB
- Connectivité sans-fil : aucune

FigureI.2. Le Raspberry pi modèle B

Modèle B+ :

2ème de la série, le Raspberry Pi 1B+ reprend les caractéristiques du modèle précédent mais ajoute 2 ports USB, qui était une grande attente des utilisateurs jusqu'à sa sortie en juillet 2014.

- Sortie : 2014
- Ports : 4x USB 2.0, HDMI, Ethernet
- CPU : 700MHz

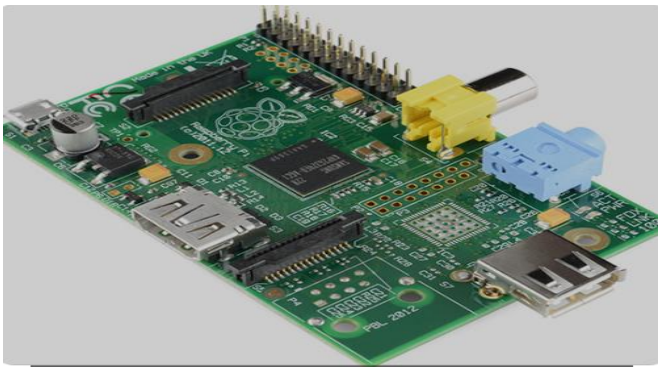
- RAM : 512MB
- Connectivité sans-fil : aucune

FigureI.3. Le Raspberry pi modèle B+

Modèle A :

Le Raspberry Pi 1A est un modèle obsolète qui se voulait penser pour être plus léger que le modèle de base. Il offrait une puissance réduite, et moins de ports, qui réjouissaient les portefeuilles.

- Sortie : 2014
- Ports : 1x USB 2.0, HDMI
- CPU : 700MHz
- RAM : 256MB
- Connectivité sans-fil : aucune



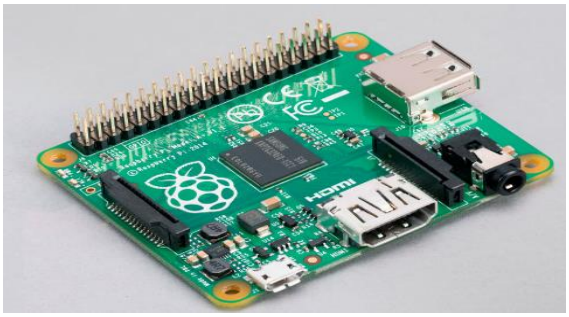
FigureI.4. Le Raspberry pi modèle A

Modèle A+ :

Un nouveau format de carte mère avec le Raspberry Pi 1A+. Dans un format plus réduit, on retrouve un micro-ordinateur aussi puissant que les modèles B et B+ mais avec uniquement

un port USB et une disparition du port Ethernet. Il est pensé comme une variante plus économique pour des projets plus légers et souvent hors ligne.

- Sortie : 2014
- Ports : 1x USB 2.0, HDMI
- CPU : 700MHz
- RAM : 512MB
- Connectivité sans-fil : aucune

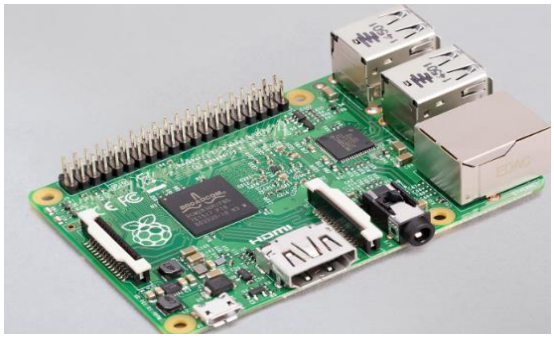


FigureI.5.Le Raspberry pi modèle A+

I.3.3.2.Raspberry Pi 2

Un nouveau processeur et un doublement de la RAM. Le Raspberry Pi 2B à un processeur cadencé à 900 MHz et 1Gb de mémoire RAM. On retrouve toujours les 4 ports USB, le HDMI, 1 port Ethernet et une nappe pour brancher un écran.

- Sortie : 2015
- Ports : 4x USB 2.0, HDMI, Ethernet
- CPU : 900MHz
- RAM : 1GB
- Connectivité sans-fil : aucune

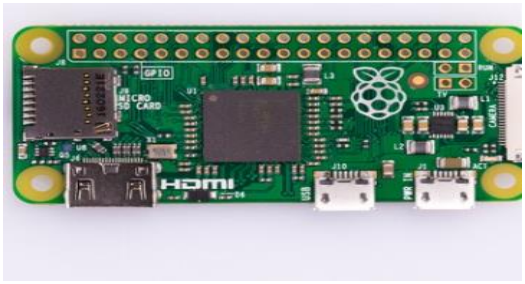


FigureI.6. Le Raspberry pi 2

I.3.3.3.Raspberry Pi Zéro

Le Raspberry Pi Zéro, est une version plus puissante, avec moins de RAM et de connectivité et plus petite du Raspberry Pi 2. Exit les ports USB classiques et disparition du port Ethernet. En revanche, on retrouve 1 port micro-USB et 1 port micro-OTG. Les ports GPIO natifs disparaissent et nécessite l'achat d'un header GPIO en complément.

- Sortie : 2015
- Ports : 1x micro-USB et 1x micro-USB OTG, mini HDMI
- CPU : 1GHz
- RAM : 512MB
- Connectivité sans-fil : aucune



FigureI.7. Le Raspberry pi zéro

I.3.3.4.Raspberry Pi Zéro W

Le Zéro profite d'une mise à jour de connectivité. On retrouve la même puissance que le modèle d'origine mais avec un « W » en plus qui, indique l'arrivée d'une version Wifi mais aussi Bluetooth. Le Raspberry Pi Zéro W est le modèle idéal pour tous les projets connectés nécessitant une carte-mère compacte.

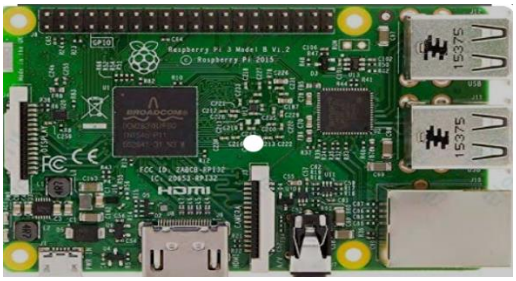
- Sortie : 2017
- Ports : 1x micro-USB et 1x micro-USB OTG, mini-HDMI
- CPU : 1,4GHz
- RAM : 512MB
- Connectivité sans-fil : Bluetooth 4.1LE, Wifi

FigureI.8. Le Raspberry pi zéro W

I.3.3.5.Raspberry Pi 3

Nouvelle génération de Raspberry, Parmi les nouveautés intégrées au Raspberry Pi 3, on retrouve : un CPU plus rapide mais surtout l'entrée dans le monde du sans-fil avec une connectivité Bluetooth et Wifi nativement.

- Sortie : 2016
- Ports : 4x USB 2.0, HDMI
- CPU : 1,2GHz
- RAM : 1GB
- Connectivité sans-fil : Bluetooth 4.1LE, Wifi



FigureI.9. Le Raspberry pi 3

I.3.3.6.Raspberry Pi 4

Pour notre projet, nous avons choisi le Raspberry Pi 4, Le Raspberry Pi 4 est le modèle de référence du moment. Il s'agit de la version la plus aboutie Pi. Désormais disponible en 4 formats, doublant à chaque fois la RAM de 1GB à 8GB, le Raspberry Pi 4 offre le processeur le plus puissant de la série (après le Pico sorti en 2021).

- Sortie : 2019
- Ports : 2X USB 3.0, 2X USB 2.0, Ethernet, 2xMicro-HDMI
- CPU : 1,5GHz
- RAM : 1GB ; 2GB, 4GB ou 8 GB
- Connectivité sans-fil : Bluetooth 5.0 L.B.E, Wifi

FigureI.10. Le Raspberry pi 4

Il peut se brancher directement sur un écran ou un moniteur. Il sert dans le domaine du langage et de la programmation informatique : par exemple, pour créer un serveur web domestique, gérer une imprimante 3D à distance ou encore concevoir un robot**. [7]**

Il joue aussi un rôle central dans le paramétrage de réseaux VPN, le rétrogaming et les dispositifs de vidéosurveillance.

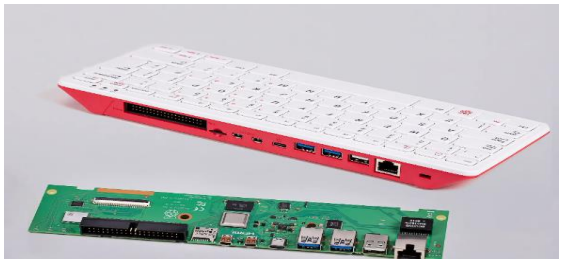
Le choix d'un Raspberry Pi 4 dépend des usages escomptés. En matière de puissance et de configuration technique, on peut contrôler la taille de la mémoire vive exprimée en Go RAM, le modèle de processeur, sans oublier la fréquence maximale observée (en GHz). Egalement le type de RAM graphique, ainsi que la compatibilité du système d'exploitation, notamment si on travaille sous Linux.

Un Raspberry Pi 4 peut être vendu seul. Il existe néanmoins des kits de démarrage qui comprennent différents accessoires. Parmi les équipements à privilégier, on retrouve le bloc d'alimentation, une carte micro SD, un câble HDMI (ou micro HDMI) et des dissipateurs. Pour les connectiques, un ou plusieurs ports USB (2.0 ou 3.0) sont préconisés, tout comme une sortie HDMI ou son équivalent micro. Le Raspberry Pi 4 reconnaisse aussi les technologies Bluetooth et Wifi**. [7]**

I.3.3.7.Raspberry Pi 400

Le Raspberry Pi 400 est pour les gens qui cherchent des solutions plus accessibles pour développer leurs projets de Raspberry Pi, plus simplement. Le Pi 400 change radicalement de design et vient intégrer un boitier natif avec clavier. Le Pi 400 reprend les mêmes composants que le Raspberry Pi 4 dans un format différent avec 1 seul port USB 2.0 et un processeur cadencé à 1,8GHz.

- Sortie : 2020
- Ports : 1X USB 3.0, 2X USB 2.0
- CPU : 1,8GHz
- RAM : 4GB
- Connectivité sans-fil : Bluetooth 5.0 L.B.E, Wifi



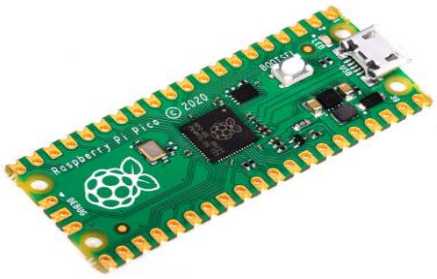
FigureI.11.Le Raspberry pi 400

I.3.3.8.Raspberry Pi Pico

Dernier né de la famille des Raspberry Pi, le Raspberry Pi Pico est comme son nom l'indique, une carte miniature pensée pour servir de microcontrôleur, Utilisant un microprocesseur maison le RP2040, ce Raspberry Pi Pico n'est pas un SBC (Single Board Computer) mais un microcontrôleur qui n'intègre pas nativement de connecteurs (il faudra les souder vous-même).

- Sortie : 2021
- Ports : 1X USB C
- CPU : 133 MHz
- RAM : 264KB

- Connectivité sans-fil : Bluetooth



FigureI.12. Le Raspberry pi pico

I.3.4.Caractéristique de la Raspberry Pi [3]

Tableau I.1 : Caractéristique de la Raspberry Pi

modèle	A	A+	B	B+
CPU		Monocoeur ARM	700MHz	
GPU	Décodeur	Vidéo	Broadcam	VideoCore IV
RAM	256 MO		512 MO	
USB	1* USB2.0		2*USB2.0	4* USB2 .0
Ethernet	0	10/100	0	10/100
Entrées/sorties	GPIO 26 pts	GPIO 40 pts	GPIO 26 pts	GPIO 40 pts
OS	Officiel : Raspbian	Tiers : Fedora, XBMC /Kodi,OSMC		
stockage	SD	Micro SD	SD	Micro SD
Dimensions	86*54*17	65*54*17	86*54*17	
Poids	45 g	23g	45 g	
Consommation	1.5 W	1 W	3 .5 W	3 W

I.4. Langage Python

I.4.1. Historique

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom Python vient d'un hommage à la série télévisée Monty Python's Flying Circus dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991. **[7]**

La version de Python la plus courante est la version 3. Plus précisément, la version 3.11 a été publiée en Décembre 2022.

I.4.2.Definition de Langage Python

Python est un langage peut être considéré comme un langage émergent car il est encore au cours de développement et de recherche. C'est un excellent langage pour développer une application d'ingénierie.

Python présente de nombreux avantages par rapport au langage traditionnel, ce qui est important dans un environnement d'apprentissage.

Les programmes Python ne sont pas compilés en code machine, mais sont exécutés par un interrupteur, ce qui signifie que les programmes peuvent être testés et débogués rapidement, permettant à l'utilisateur de se concentrer davantage sur les principes du programme et moins sur la programmation elle-même. Car il n'est pas nécessaire de compiler, de lier et d'exécuter après chaque correction.

Python peut être développé en un temps beaucoup plus court que d'autres programmes tels que C++. Python est un logiciel source de stylo, ce qui signifie que sans identifiant il est inclus dans la plupart des distributions LINUX. **[3]**

FigureI.13. Le logo du python

I.4.3.Variables [5]

I.4.3.1. Définition

Une variable est une zone de la mémoire de l'ordinateur dans laquelle une valeur est stockée. Aux yeux du programmeur, cette variable est définie par un nom, alors que pour l'ordinateur, il s'agit en fait d'une adresse, c'est-à-dire d'une zone particulière de la mémoire.

En Python, la déclaration d'une variable et son initialisation (c'est-à-dire la première valeur que l'on va stocker dedans) se font en même temps.

I.4.3.2. Les types de variables

Les trois principaux types dont nous aurons besoin dans un premier temps sont les entiers (integer ou int), les nombres décimaux que nous appellerons floats et les chaînes de caractères (string ou str). Bien sûr, il existe de nombreux autres types (par exemple, les booléens, les nombres complexes, etc.).

I.4.4.Affichage [5]

I.4.4.1. La fonction print ()

La fonction print () qui affiche une chaîne de caractères, En fait, la fonction print () affiche l'argument qu'on lui passe entre parenthèses et un retour à ligne. Ce retour à ligne supplémentaire est ajouté par défaut.

```
>>> print("hello paython")
hello paython
>>>
```

La fonction print () peut également afficher le contenu d'une variable quel que soit son type. Par exemple, pour un entier :

```
>>> var=3
>>> print(var)
3
```

Il est également possible d'afficher le contenu de plusieurs variables (quel que soit, leur type) en les séparant par des virgules :

```
>>> x=22
>>> nom="sarah"
>>> print(nom ,"a" ,x, "ans")
sarah a 22 ans
>>>
```

I.4.4.2. Écriture formatée

L'écriture formatée est un mécanisme permettant d'afficher des variables avec un certain format, par exemple justifiées à gauche ou à droite, ou encore avec un certain nombre de décimales pour les floats.

I.4.4.3. Écriture scientifique

Pour les nombres très grands ou très petits, l'écriture formatée permet d'afficher un nombre en notation scientifique (sous forme de puissance de 10) avec la lettre e :

```
>>> print(f"{1_000_000_000:e}")
1.000000e+09
```

I.4.5. Bibliothèques utilisées [1]

Python est également connu pour sa bibliothèque standard complète, qui fournit un large éventail de modules pour diverses tâches de programmation, notamment la manipulation de fichiers, les opérations mathématiques, la gestion de réseau et les interfaces utilisateur graphiques.

I.4.5.1. Numpy

Numpy est une bibliothèque Python populaire utilisée pour effectuer des calculs numériques basés sur des tableaux. L'implémentation canonique de numpy utilisée par la plupart des programmeurs s'exécute sur un seul cœur de processeur et est parallélisée pour utiliser plusieurs cœurs pour certaines opérations. Cette restriction à une exécution à processeur unique à nœud unique limite à la fois, la taille des données pouvant être traitées et la vitesse potentielle du code numpy.

I.4.5.2.Tkinter

Tkinter est la bibliothèque d'interface graphique standard pour Python. Lorsqu'il est combiné à Tkinter, fournit un moyen rapide et facile pour créer des applications graphiques.

Tkinter fournit une puissante interface orientée objet simple et conviviale. La création d'une application graphique à l'aide de Tkinter est une tâche assez facile. Tout ce que nous avons à faire est de suivre les étapes suivantes :

1. Importé le module Tkinter.
2. Création de la fenêtre principale de l'application graphique.
3. Ajouté un ou plusieurs des widgets graphiques.

4. Entré la boucle d'évènements principale pour agir contre chaque évènement déclenché par l'utilisateur.

I.4.5.3. Open Cv

Est une bibliothèque graphique. Elle est spécialisée dans le traitement d'images, que ce soit pour de la photo ou de la vidéo**. [8]**

Cette bibliothèque est indispensable dans les applications en traitement d'images. La bibliothèque Open Cv sera utilisée dans notre projet.

I.4.5.3.1.Historique

Lancé en 1999, le projet Open CV est développé initialement par Intel pour optimiser les applications gourmandes en temps processeur. Cela faisait partie d'une série de projets tels que l'affichage d'un mur en 3 dimensions.

Les principaux acteurs du projet sont l'équipe de développement de bibliothèque de chez Intel ainsi qu'un certain nombre d'experts dans l'optimisation de chez Intel Russie. Les objectifs de base du projet étaient :

- Faire des recherches sur la vision par ordinateur en vue de fournir un logiciel libre et optimisé.

- Établir une infrastructure commune s'appuyant sur les développeurs pour obtenir un code plus lisible et transférable.

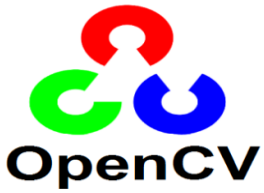
- Continuer à développer en rendant le code portable et permettre des performances optimisées gratuites avec une licence qui est libre de toutes contraintes commerciales.

La première version alpha d'Open CV fut présentée lors de la conférence IEEE sur la vision par ordinateur et la reconnaissance de formes en 2000. Après cela, cinq versions bêta ont été publiées entre 2001 et 2005 et la première version 1.0 a été publiée en 2006. La deuxième version majeure d'Open CV manufacturé en octobre 2009. Il s'agit d'Open CV 2 incluant des changements majeurs au niveau du langage C++ servant à faciliter le développement de nouvelles fonctions et améliorant les performances.

I.4.5.3.2.Definition d'Open CV

Initialement développée par Intel, Open CV (Open Computer Vision) est une bibliothèque graphique. Elle est spécialisée dans le traitement d'images, que ce soit pour de la photo ou de la vidéo. Elle est disponible sur la plupart des systèmes d'exploitation et existe pour les langages Python, C++ et Java ; Sous licence BSD (Berkeley Software Distribution Licence).

Open CV peut être réutilisé librement, en tout ou partie, pour être intégré au sein d'un autre projet. C'est notamment cette notion qui fait qu'Open CV est très populaire et à la base de nombreux logiciels de traitements d'images/vidéos. Elle est aujourd'hui développée, maintenue, documentée et utilisée par une communauté de plus de 40 000 membres actifs**. [7]**



FigureI.14. Le logo de Open CV

I.5. Conclusion

Dans ce chapitre nous avons vu la catre raspberry pi et les déférents types, et la carte Raspberry pi 4 principalement ; nous avons aussi expliqué le langage python qui est également connu pour sa simplicité et sa facilité d'apprentissage, ce qui en fait un choix populaire pour les débutants en programmation ; ainsi que la Bibliothèque Open CV qui est spécialisée dans le traitement d'images.

Le prochain chapitre sera consacré à la programmation avec python, appliqué en traitement d'images.

Chapitre II

Traitement d’images avec Python

II.1. Introduction

Le traitement d'images est une discipline de l'information et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information.

Python devient un bon choix pour de telles tâches de traitement d'images. Cela est dû à sa popularité croissante en tant que langage de programmation scientifique et à la disponibilité gratuite de nombreux outils avancés de traitement d'images dans son écosystème.

Ce chapitre présente d'abord les détails du traitement d'images et comment appliquer différentes modifications aux images en utilisant différents filtres, puis à introduire quelque exemple basés sur le langage Python et la bibliothèque Open CV.

II.2. Historique

Le traitement d'images commence à être étudié dans les années 1920 pour la transmission d'images par le câble sous-marin allant de New York à Londres. Harry G. Bartholomew et Maynard D. McFarlane effectuent la première numérisation d'image avec compression de données pour envoyer des fax de Londres à New York. Le temps de transfert passe ainsi de plus d'une semaine à moins de trois heures.

Le véritable essor du traitement d'images n'a lieu que dans les années 1960 quand les ordinateurs commencent à être suffisamment puissants pour travailler sur des images.

On peut dire que le traitement d'images est né de l'idée et de la nécessité de remplacer l'observateur humain par la machine. Les images ou les signaux provenant des capteurs doivent être numérisés pour pouvoir être traités par un ordinateur. A partir d'une image numérique, il convient d'extraire les informations pertinentes en regard de l'application concernée, les traiter puis les interpréter.

D'une façon plus simple on peut dire que le traitement d'images désigne l'ensemble de techniques qui permettent de calculer à partir d'une image d'entrée, une nouvelle image de sortie où les informations recherchées sont mises en évidence**. [9]**

II.3. Généralités sur le traitement d'images

II.3.1.Definition du traitement d'images

Le traitement d'images est une branche du traitement de signal dédiée aux images et vidéo et l'ensemble des opérations effectuées sur l'image, afin d'en améliorer la lisibilité et d'en faciliter l'interprétation. C'est, par exemple, le cas des opérations de rehaussement de contraste, élimination du bruit et correction d'un flou.

C'est aussi l'ensemble d'opérations effectuées pour extraire des "informations" de l'image comme la segmentation et l'extraction de contours.

Avant le traitement d'images, on peut aussi effectuer des opérations de prétraitement qui sont toutes les techniques visant à améliorer la qualité d'une image. Les opérations de traitement peuvent être divisées en deux niveaux :

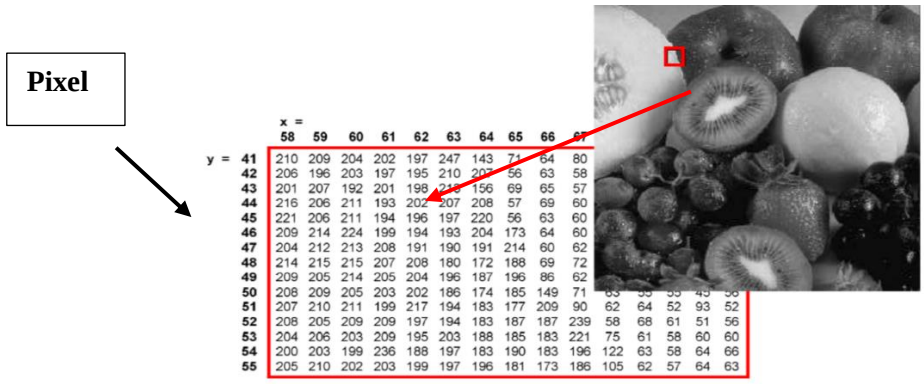
Le traitement bas-niveau qui se construit autour des méthodes d'analyses d'image ayant pour but d'extraire des caractéristiques des images et d'analyser sans les interpréter (contours, texture, par exemple). C'est des données de nature numérique

Le traitement haut-niveau intègre l'ensemble des méthodes permettant d'interpréter les caractéristiques issues du bas-niveau (prise de décision, classification, Intelligence artificielle).

C'est des entités de nature symbolique associées à une représentation de la réalité extraite de l'image**. [10]**

II.3.2. Définition d'une image

Une image est un signal 2D(x, y), D'un point de vue mathématique : Une image est une matrice de nombres représentant un signal, et plusieurs outils permettent de manipuler ce signal, D'un point de vue humain : Une image contient plusieurs informations sémantiques, et il faut interpréter le contenu au-delà de la valeur des nombres. **[11]**



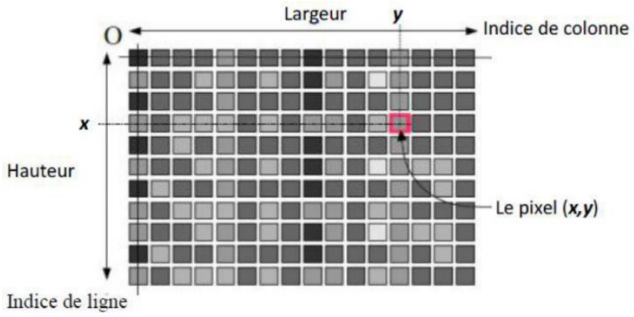
FigureII.1. Exemples d'une image en niveaux de gris

II.3.3.Image numérique

L'appellation d'image numérique désigne toute image (dessin, icône, photographie …) acquise, créée, traitée et stockée sous forme binaire.

L'image numérique est l'image dont la surface est divisée en éléments de taille fixe appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs.

La numérisation d'une image est la conversion de celle-ci de son état analogique en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques f(x, y), comme la montre la figure ci-dessous où :

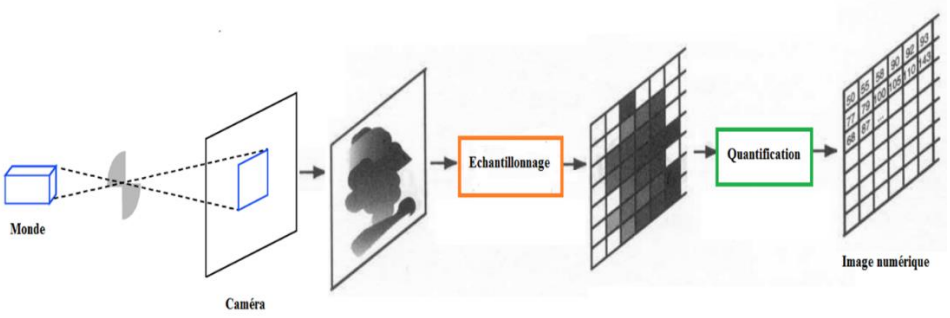


FigureII.2. Représentation d'image numérique

Procédés sont impliqués pour numériser une image :

Numérisation = Echantillonnage + Quantification

L'échantillonnage est limité par la capacité du capteur, donc le nombre de pixels disponible, La quantification est limitée par la quantité de tons de gris définie dans l'intervalle, n bits pour des valeurs entre 0 et 2^n–1.



FigureII.3. Echantillonnage et Quantification

II.3.3.1.Qualité de l'image numérique

Elle dépend, d'une part, de la qualité des images d'origine et, d'autre part, des moyens mis en œuvre pour convertir un signal analogique en signal numérique. Elle dépend aussi de :

- La qualité des périphériques de numérisation de l'image, du nombre de niveaux de gris ou de couleurs enregistrées, etc.

- La qualité de l'affichage à l'écran : définition de l'écran, nombre de teintes disponibles.

Les critères d'appréciation de la qualité d'une image, tels que cités succinctement ci-dessus, dépendent largement de la structure même de l'image réaliste ou conceptuelle et de son mode de représentation (bitmap ou vectorielle).

II.3.3.2.Images bitmap et images vectorielles

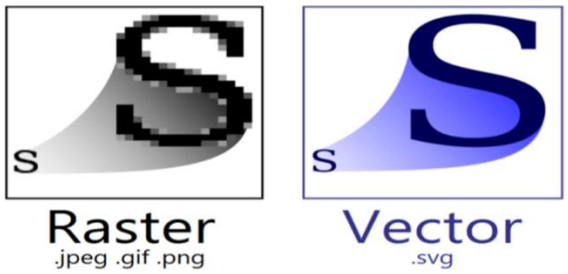
Les images appartiennent à deux grandes familles :

II.3.3.2.1.Images matricielles

Dans la description que nous avons faite jusqu'à présent des images nous avons utilisé une matrice. On dit alors que l'image est matricielle ou en anglais bitmap. Ce type d'image est adapté à l'affichage sur écran mais peu adapté pour l'impression car bien souvent la résolution est faible.

II.3.3.2.2.Images vectorielles

Le principe des images vectorielles est de représenter les données de l'image à l'aide de formules mathématiques. Cela permet alors d'agrandir l'image indéfiniment sans perte de qualité et d'obtenir un faible encombrement.

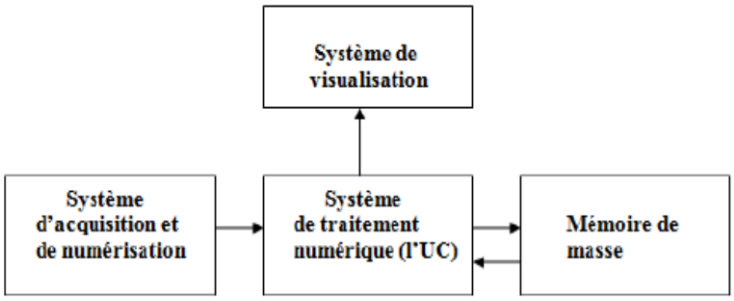


FigureII.4. Différence entre l'image vectorielle et l'image matricielle

II.3.4. Système de traitement d'image

Un système de traitement d'images est généralement composé des unités suivantes :

- Un système d'acquisition et de numérisation qui permet d'effectuer l'échantillonnage et la quantification d'une image.
- Une mémoire de masse pour stocker les images numérisées.
- Un système de visualisation.
- Une unité centrale permettant d'effectuer les différentes opérations de traitement d'images. **[12]**



FigureII.5. Composition d'un système de traitement numérique.

- **Acquisition et numérisation**

L'acquisition d'images constitue un des maillons essentiels de toute chaîne de conception et de production d'images. Pour pouvoir manipuler une image sur un système informatique, il est avant tout nécessaire de lui faire subir une transformation qui la rendra lisible et manipulable par ce système. Le passage de cet objet externe (l'image d'origine) à sa représentation interne (dans l'unité de traitement) se fait grâce à une procédure de numérisation. Ces systèmes de saisie, dénommés optiques, peuvent être classés en deux catégories principales : les caméras numériques et les scanners. **[12]**

- **Visualisation**

Tout système de traitement d'image est doté d'un dispositif de visualisation qui permet l'affichage des images.

L'utilisation de différents types de restituteurs permet de transformer le signal numérique qu'est la matrice image en un signal analogique visible par l'œil de l'observateur. Pour cela, différents types de supports peuvent être employés : moniteur vidéo, clichés photographiques, impression sur papier. Dans tous les cas et pour chaque échantillon de l'image numérique, on recrée un nouvel élément d'image ou un nouveau pixel dont on choisit la forme de façon à reconstituer une image analogique qui soit la plus proche possible de l'image avant numérisation compte tenu des erreurs introduites lors de l'acquisition, de la numérisation et de la transmission **.[12]**

II.3.5.Représentation des images

Une matrice de dimension M x N, Chaque élément (Pixel) à une valeur entière dans l'intervalle L= [Lmin Lmax], Le nombre de bits requis pour représenter les niveaux de gris dans l'intervalle L est b ; la relation entre b et L est : $L = 2^b$, Le nombre de bits pour entreposer une image (la taille) est donc : k = M x N x b (bits).

II.4.Traitement d'image en langage python

Python est largement utilisé pour le traitement d'images en raison de sa richesse de bibliothèques telles qu'Open CV, Pillow et scikit-image.

II.4.1. Open cv et Pycharm

OpenCV (Open Source Computer Vision) est une bibliothèque de vision par ordinateur qui contient diverses fonctions pour effectuer des opérations sur des images ou des vidéos. Il a été initialement construit par Intel mais plus tard géré par Willow Garag, il est actuellement géré par Itseez. Il s'agit d'une bibliothèque multiplateforme disponible pour les langages de programmation autres que python**. [13]**

II.4.1.1. Pycharm

PyCharm est un IDE multiplateforme utilisé dans la programmation informatique spécifiquement pour Python. La plateforme développée par JetBrains est principalement utilisée dans l'analyse de code, le débogueur graphique etc… Elle supporte le développement web avec Django ainsi que la Data Science avec Anaconda. **[13]**

II.4.1.2. les étapes d'installation d'open cv pycharm

- **1ére étape :** installation de pycharm (distribution de python)

- **2éme étape :** Ouvrir PyCharm et crée un nouveau projet Python ou ouvrir un projet existant.

- **3éme étape :** Accédons à Fichier > Paramètres > Projet : <nom projet> > Interprète de projet.

- **4éme étape :** Clique sur l'icône + pour ajouter un nouveau package à l'interpréteur.

- **5éme étape** : Recherche opencv-python et cliquez sur Installer le package.

- **6éme étape** : Attendre que le processus d'installation se termine. Importé le module cv2 dans votre code Python.

Si nous avons déjà installé OpenCV à l'aide de pip ou de tout autre gestionnaire de packages, nous n'avons pas besoin de le réinstaller dans PyCharm. Nous pouvons simplement sélectionner l'interpréteur existant qui contient OpenCV.

II.4.1.3. Exemples de codes en Python avec Open CV

1. **Lire une image**

```python
import cv2

# Charger une image
img = cv2.imread('E:\sarah\master\memoire\Capture.PNG')
```

2. **Afficher une image**

```python
import cv2

# Charger une image
img = cv2.imread('E:\sarah\master\memoire\Capture.PNG')

# Afficher l'image
cv2.imshow('Image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

II.5. Filtrage

Le principe du filtrage est de modifier la valeur des pixels d'une image, généralement dans le but d'améliorer son apparence. En pratique, cela consiste à créer une nouvelle image en tant que serviteur des valeurs de pixels de l'image d'origine.

Toutes les transformations de l'image d'origine ne rentrent pas dans la catégorie du filtrage : zoom, découpage, projections… **[14]**

II.5.1. filtre linéaire

Un système linéaire continu est un filtre linéaire si et seulement si la relation entre l'entrée E et la sortie S est une convolution **[15]**

II.5.1.1. Filtre de convolution

Le filtre de convolution est une technique de traitement d'image qui consiste à appliquer un noyau de convolution à une image afin de modifier son apparence.

Le noyau de convolution est une matrice de nombres qui représente le comportement du filtre. Il est généralement de petite taille, par exemple 3x3 ou 5x5, et peut être conçu pour effectuer une variété d'opérations, telles que l'accentuation des bords, le flou, la détection des contours, etc.

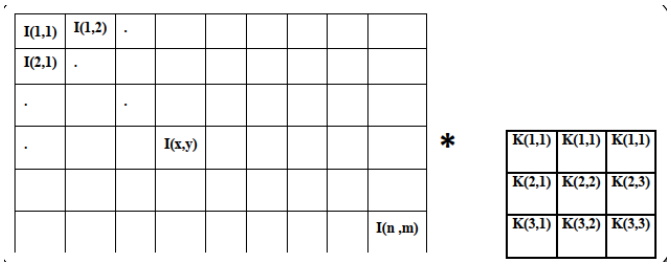
Le processus de convolution implique le déplacement du noyau sur l'image pixel par pixel, en effectuant une multiplication élément par élément entre les valeurs de l'image et du noyau à chaque position, puis en additionnant les produits pour obtenir la valeur du pixel de sortie.

Le filtre de convolution est largement utilisé dans le traitement d'image, notamment pour améliorer la qualité de l'image, réduire le bruit, extraire des caractéristiques, etc.

Des exemples de noyaux de filtres typiques sont présentés ici :

• Filtres passe-bas (Lissage) : un « Moyen » qui lisse l'image, le « binôme-gaussien » lisse également l'image de manière moins marquée mais plus régulière

• filtres passe-haut (Accentuation) : un "amplificateur de contraste", des filtres différenciateurs orientés (Sobel horizontal, gradient oblique), un différenciateur non orienté (Laplacien) **[16].**



FigureII.6. Filtre de convolution

II.5.1.2. Filtre moyen

C'est un cas particulier de filtre de convolution passe-bas, qui remplace chaque pixel par la moyenne des valeurs des pixels voisins et du pixel central. La taille du masque dépend de l'intensité du bruit et de la taille des détails significatifs de l'image traitée**. [15]**

Son masque pour un filtre **3*3** est : $\frac{1}{9}\begin{matrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{matrix}$

Cela se fait en convolant une image avec un filtre de boîte normalisé. Il prend simplement la moyenne de tous les pixels sous la zone du noyau et remplace l'élément central. Ceci est fait par la fonction ***cv.blur ()*** ou ***cv.boxFilter ()***.

Nous devons spécifier la largeur et la hauteur du noyau. Un filtre de boîte normalisé 3x3 ressemblerait à ce qui suit :

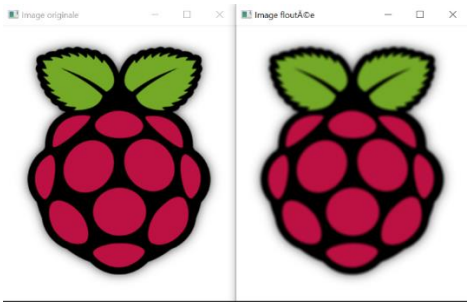
```python
import cv2

# Charger l'image
img = cv2.imread("E:\sarah\master\memoire\Capture.PNG")

# Appliquer un filtre moyen avec un noyau de 3x3 pixels
blur = cv2.blur(img,(3,3))

# Afficher l'image originale et l'image filtrée
cv2.imshow('Image originale',img)
cv2.imshow('Image filtree',blur)

# Attendre que l'utilisateur appuie sur une touche pour fermer les fenêtres d'affichage
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Dans ce code, nous avons chargé une image en utilisant ***cv2.imread ()***, puis nous avons appliqué un filtre moyen à l'image en utilisant ***cv2.blur ()***. Le filtre moyen utilise un noyau de convolution pour prendre la moyenne des pixels voisins et lisser l'image.

Ensuite, nous avons affiché l'image originale et l'image floutée côte à côte en utilisant ***cv2.imshow ()***, et attendu que l'utilisateur appuie sur une touche en utilisant ***cv2.waitKey ()***.

Enfin, nous avons détruit toutes les fenêtres avec ***cv2.destroyAllWindows ()***

L'affichage :



FigureII.7. Image floue avec filtre moyen

II.5.1.3. Filtre gaussien

Ce filtre très populaire utilisait la loi de probabilité gaussienne. Soient U(x) le niveau de gris en un point x de l'image à traiter et Gσ l'écart-type gaussien σ donné par la formule suivante :

$$G_\sigma = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{|X||X|}{2\sigma\sigma}\right) \quad \text{(II.1)}$$

Le filtrage gaussien de l'image résulte de la convolution de cette fonction avec des gaussiennes en chaque point de l'image :

$$U(x,\sigma) = (G_\sigma * U_0)(X) = \int_{R^2} G_\sigma(x-y)\, U_0(y)\, dy \quad \text{(II.2)}$$

Dans ce cas aussi, l'effet du filtre augmente avec la taille de son masque. Les contours et les détails fins sont mieux conservés qu'avec la moyenne, car en pondération gaussienne, le filtre gaussien ou lissé tient mieux compte des corrélations entre pixels, notamment pour une texture d'image (la fonction de corrélation des niveaux de gris pour une texture est fréquemment modélisée par un gaussien). Le filtre gaussien est un bon exemple des performances que l'on peut obtenir avec un filtre linéaire à réponse impulsionnelle finie.

Dans cette méthode, nous utilisons le noyau gaussien. Cela se fait avec la fonction ***cv.GaussianBlur ()***.

Nous devons spécifier la largeur et la hauteur du noyau, qui doivent être positives et impaires.

Nous devons également spécifier l'écart type dans les directions X et Y, sigmaX et sigmaY respectivement. Si seul sigmaX est spécifié, sigmaY est considéré comme identique à sigmaX. Si les deux sont donnés sous forme de zéros, ils sont calculés à partir de la taille du noyau. Le flou gaussien est très efficace pour supprimer le bruit gaussien d'une image. [**17]**

Voici un exemple de code Python qui applique un filtre gaussien à une image en utilisant la bibliothèque OpenCV dans PyCharm :

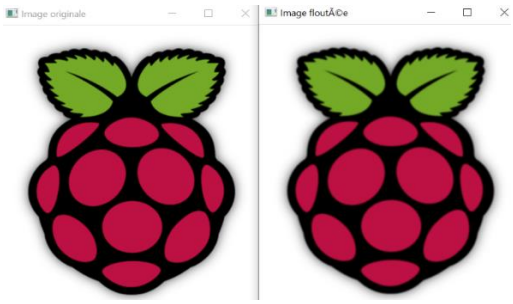
```
1
2   import cv2
3
4   # Charger l'image
5   img = cv2.imread('E:\sarah\master\memoire\Capture.PNG')
6
7   # Appliquer un filtre gaussien avec un noyau 5x5 et un écart-type de 0
8   blur = cv2.GaussianBlur(img,(3,3),0)
9
10  # Afficher l'image originale et l'image floutée côte à côte
11  cv2.imshow('Image originale', img)
12  cv2.imshow('Image floutée', blur)
13  cv2.waitKey(0)
14  cv2.destroyAllWindows()
```

Dans ce code, nous avons chargé une image en utilisant cv2.imread (), puis nous avons appliqué un filtre gaussien à l'image en utilisant cv2.GaussianBlur (). Le filtre gaussien utilise une distribution gaussienne pour prendre une moyenne pondérée des pixels voisins et lisser l'image.

Nous avons utilisé un noyau de 5x5 et un écart-type de 0 pour le filtre gaussien. Vous pouvez ajuster la taille du noyau et l'écart-type en fonction de vos besoins.

Ensuite, nous avons affiché l'image originale et l'image floutée côte à côte en utilisant cv2.imshow (), et attendu que l'utilisateur appuie sur une touche en utilisant cv2.waitKey (). Enfin, nous avons détruit toutes les fenêtres avec cv2.destroyAllWindows ().

L'affichage :



FigureII.8.Image floue avec filtre gaussien

II.5.2.Filtre non linéaire

II.5.2.1.filtre médian

Le filtre médian est un filtre numérique non linéaire, souvent utilisé pour la réduction de bruit. La réduction de bruit est une étape de prétraitement classique visant à améliorer les résultats de traitements futurs (détection de bords par exemple). La technique de filtre médian est largement utilisée en traitement d'images numériques car il permet sous certaines conditions de réduire le bruit tout en conservant les contours de l'image**. [18]**

Voici un exemple de code Python qui applique un filtre médian à une image

```python
import cv2

# Charger l'image
img = cv2.imread('E:\sarah\master\memoire\Capture1.PNG')

# Appliquer un filtre médian avec une taille de noyau de 3x3
median = cv2.medianBlur(img, 3)

# Afficher l'image originale et l'image filtrée côte à côte
cv2.imshow('Image originale', img)
cv2.imshow('Image filtrée', median)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Dans ce code, nous avons chargé une image en utilisant cv2.imread (), puis nous avons appliqué un filtre médian à l'image en utilisant ***cv2.medianBlur ()***. Le filtre médian calcule la valeur médiane des pixels voisins pour chaque pixel de l'image et supprime les valeurs extrêmes.

Nous avons utilisé une taille de noyau de 3x3 pour le filtre médian. Vous pouvez ajuster la taille du noyau en fonction de vos besoins.

Ensuite, nous avons affiché l'image originale et l'image filtrée côte à côte en utilisant cv2.imshow (), et attendu que l'utilisateur appuie sur une touche en utilisant cv2.waitKey (). Enfin, nous avons détruit toutes les fenêtres avec cv2.destroyAllWindows ().

L'affichage :

FigureII.9. Image avec filtre médian

Il existe d'autres filtres non linéaires comme le filtre max, le filtre min, ...

II.5.3. Détection des contours

La détection de contours est un processus en traitement d'image qui consiste à identifier les zones où il y a une forte variation de luminosité ou de couleur dans une image, et à les représenter sous forme de courbes ou de lignes qui délimitent les frontières des objets présents dans l'image.

Cela peut être réalisé en utilisant des algorithmes tels que le filtre de Sobel, le filtre de prewitt et Canny ou le filtre de Laplace, qui analysent les pixels voisins pour détecter les changements de couleur ou de luminosité brusques. La détection de contours est une étape importante dans de nombreuses applications de traitement d'image, telles que la segmentation d'objets, la reconnaissance de formes et la reconstruction en 3D**. [19]**

II.5.3.1. Filtre Prewitt

Le filtre de Prewitt est un filtre linéaire utilisé en traitement d'image pour la détection de contours. Il est souvent utilisé en conjonction avec d'autres filtres pour améliorer la qualité de détection des contours dans une image.

Le filtre de Prewitt est composé de deux masques : un pour la détection des contours horizontaux et un autre pour la détection des contours verticaux. Ces masques sont de taille 3x3 et ont des valeurs prédéfinies. Les pixels de l'image sont multipliés par les valeurs des masques et les résultats sont sommés pour donner une nouvelle valeur au pixel de sortie.

Le masque de détection de contours horizontaux ressemble à ceci :

$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Le masque de détection de contours verticaux ressemble à ceci :

$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

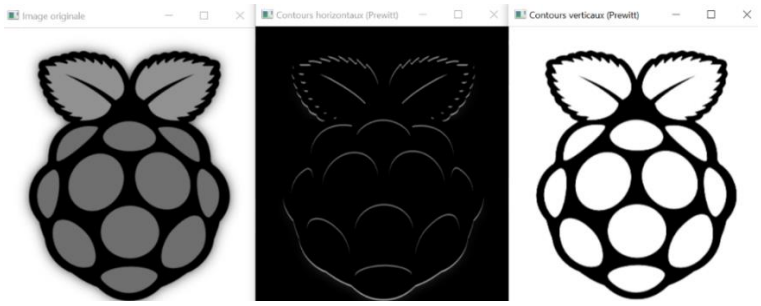
En appliquant ces masques à une image, les contours horizontaux et verticaux peuvent être détectés séparément, ce qui permet d'obtenir une détection plus précise de contours dans l'image**. [19]**

Voici un exemple de code pour appliquer le filtre de Prewitt à une image en utilisant PyCharm et la bibliothèque OpenCV :

```python
import cv2

# Chargement de l'image en niveaux de gris
img = cv2.imread("E:\sarah\master\memoire\Capture.PNG", cv2.IMREAD_GRAYSCALE)

# Création des masques de Prewitt
kernel_x = cv2.getDerivKernels(1, 0, 3)[0]
kernel_y = cv2.getDerivKernels(0, 1, 3)[0]

# Application des masques de Prewitt pour la détection de contours
dx = cv2.filter2D(img, -1, kernel_x)
dy = cv2.filter2D(img, -1, kernel_y)

# Affichage des résultats
cv2.imshow("Image originale", img)
cv2.imshow("Contours horizontaux (Prewitt)", dx)
cv2.imshow("Contours verticaux (Prewitt)", dy)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Dans ce code, nous chargeons une image en niveaux de gris à partir d'un fichier et créons deux masques de Prewitt pour la détection de contours horizontaux et verticaux. Nous appliquons ensuite ces masques à l'image à l'aide de la fonction ***filter2D*** de la bibliothèque OpenCV. Enfin, nous affichons les résultats à l'aide de la fonction ***imshow***.

L'affichage :



FigureII.10. Détection des bords avec Prewitt

II.5.3.2. Filtre Sobel

Le filtre de Sobel est un filtre linéaire utilisé en traitement d'image pour la détection de contours. Il permet de calculer le gradient d'une image en utilisant deux masques de convolution, un pour la détection des contours horizontaux et un autre pour la détection des contours verticaux.

Le masque de détection de contours horizontaux ressemble à ceci :

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Le masque de détection de contours verticaux ressemble à ceci :

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Le filtre de Sobel combine à la fois le lissage par le filtre monodimensionnel [1 2 1] et la dérivée selon une direction perpendiculaire au lissage, obtenue par le filtre [-1 0 1]

Le renforcement de poids des points sur la normale au contour, au point considéré pour réduire le bruit, pour cela le filtre de Sobel donne une meilleure estimation que celle de Prewitt. **[11]**

Le filtre de Sobel calcule la magnitude et la direction du gradient pour chaque pixel de l'image en appliquant ces deux masques de convolution. La magnitude du gradient représente l'intensité du changement dans la direction du contour, tandis que la direction du gradient donne l'orientation du contour.

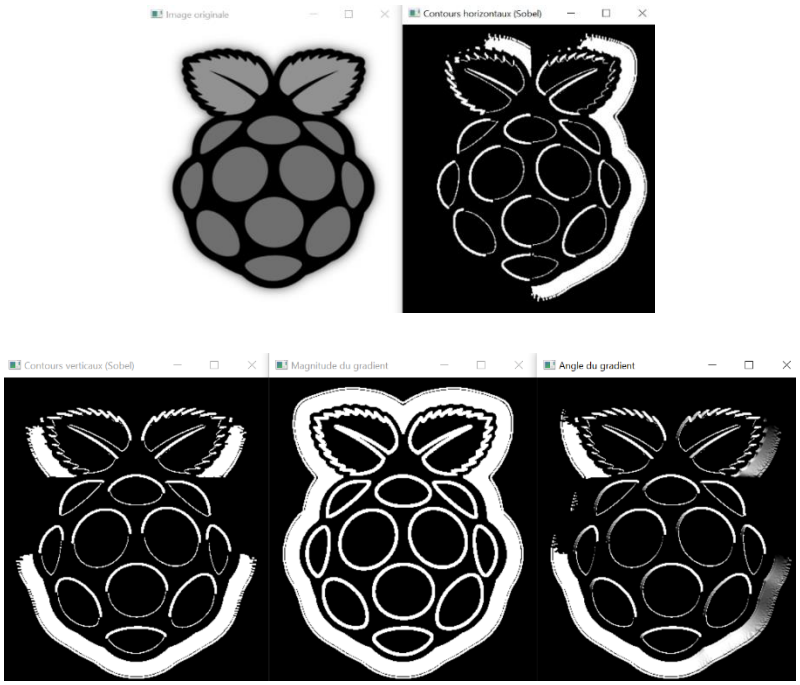
Voici un exemple de code pour appliquer le filtre de Sobel à une image en utilisant Python et la bibliothèque OpenCV :

```python
import cv2
import numpy as np

# Chargement de l'image en niveaux de gris
img = cv2.imread("E:\sarah\master\memoire\Capture.PNG", cv2.IMREAD_GRAYSCALE)

# Application du filtre de Sobel pour la détection de contours
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)

# Calcul de la magnitude et de l'angle du gradient
mag = np.sqrt(sobelx**2 + sobely**2)
ang = np.arctan2(sobely, sobelx)

# Affichage des résultats
cv2.imshow("Image originale", img)
cv2.imshow("Contours horizontaux (Sobel)", sobelx)
cv2.imshow("Contours verticaux (Sobel)", sobely)
cv2.imshow("Magnitude du gradient", mag)
cv2.imshow("Angle du gradient", ang)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Dans ce code, nous chargeons une image en niveaux de gris à partir d'un fichier et appliquons le filtre de Sobel pour la détection des contours horizontaux et verticaux en utilisant la fonction Sobel de la bibliothèque OpenCV. Nous calculons ensuite la magnitude et l'angle

du gradient en utilisant les résultats de la détection de contours et les fonctions sqrt et arctan2 de la bibliothèque Numpy. Enfin, nous affichons les résultats à l'aide de la fonction imshow.

L'affichage :



FigureII.11. Détection des bords avec Sobel

II.5.3.3. Filtre Canny [19]

Le filtre de Canny est un filtre de détection de contours largement utilisé en traitement d'image. Il a été développé par John F. Canny en 1986.

Le filtre de Canny utilise un algorithme en plusieurs étapes pour détecter les contours dans une image. Voici les étapes principales :

1. Filtrage Gaussien : L'image d'entrée est filtrée par un noyau Gaussien pour réduire le bruit dans l'image.

2. Calcul du gradient : Le gradient de l'image est calculé en utilisant les dérivées partielles selon les directions horizontale et verticale. Cela permet de détecter les variations d'intensité dans l'image qui indiquent la présence de contours.

3. Suppression du non-maxima : Les points de contour ne sont conservés que s'ils correspondent à des maxima locaux du gradient dans la direction du gradient.

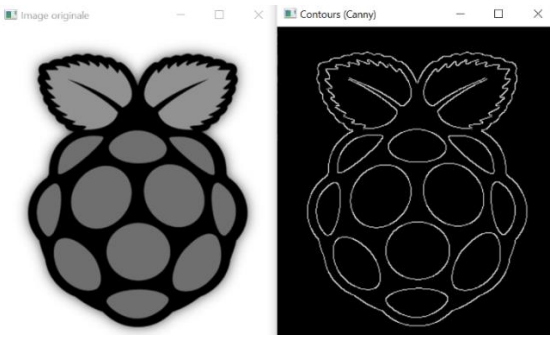
4. Seuillage : Les points de contour sont classés comme forts, faibles ou non-connus en fonction de leur intensité. Les points forts sont ceux qui ont une intensité supérieure à un seuil haut, les points faibles sont ceux qui ont une intensité supérieure à un seuil bas mais inférieure au seuil haut, et les points non-connus sont ceux qui ont une intensité inférieure au seuil bas. Les points faibles peuvent être rejetés ou acceptés en fonction de leur connectivité avec des points forts.

5. Hystérésis : Les points faibles qui sont connectés à des points forts sont considérés comme des points forts et les autres sont supprimés. Cela permet de combler les lacunes entre les segments de contour et de produire des contours lisses et continus.

Le filtre de Canny est efficace pour détecter les contours dans des images avec un faible niveau de bruit et des contours nets. Il est souvent utilisé dans des applications telles que la reconnaissance de forme, la détection d'objet et la vision par ordinateur en général. Voici un exemple de code pour appliquer le filtre de Canny à une image en utilisant Python et la bibliothèque OpenCV :

```python
import cv2

#Chargement de l'image en niveaux de gris
img = cv2.imread("E:\sarah\master\memoire\Capture.PNG", cv2.IMREAD_GRAYSCALE)

# Application du filtre de Canny pour la détection de contours
edges = cv2.Canny(img, 100, 200)

# Affichage des résultats
cv2.imshow("Image originale", img)
cv2.imshow("Contours (Canny)", edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

L'affichage :



FigureII.12. Détection des bords avec Canny

Dans ce code, nous chargeons une image en niveaux de gris à partir d'un fichier et appliquons le filtre de Canny pour la détection de contours en utilisant la fonction Canny de la bibliothèque OpenCV. Nous spécifions les seuils haut et bas pour la classification des points de contour en utilisant les valeurs 100 et 200, respectivement. Enfin, nous affichons les résultats à l'aide de la fonction imshow.

II.5.4. Le contraste [19]

Le contraste est une mesure de la différence entre les niveaux de luminosité les plus clairs et les plus sombres d'une image. Il est souvent utilisé pour décrire la netteté et la clarté d'une image.

Plus précisément, le contraste d'une image est la différence entre les niveaux de gris les plus clairs et les plus sombres de l'image, divisée par la somme de ces niveaux. Cela donne une mesure normalisée de la différence de luminosité entre les parties claires et sombres de l'image.

Une image avec un contraste élevé aura des parties claires très lumineuses et des parties sombres très sombres, avec une forte différence de luminosité entre les deux. Une image avec un contraste faible aura des parties claires et sombres plus proches les unes des autres, avec une faible différence de luminosité entre les deux.

Le contraste peut être ajusté dans une image en utilisant diverses techniques, telles que l'ajustement des niveaux de luminosité dans des zones spécifiques de l'image, ou en utilisant des algorithmes de traitement d'image tels que la transformation de l'histogramme ou la correction gamma.

Un contraste approprié est important pour améliorer la lisibilité et l'interprétation des images, en particulier dans des domaines tels que la photographie, la vidéographie, la radiologie, la microscopie, etc.

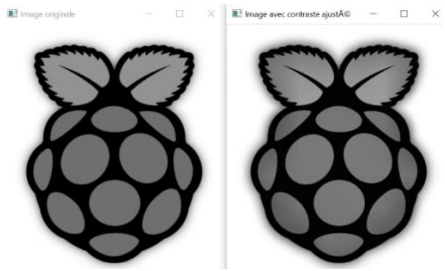
Le contraste est défini en fonction de la luminance de deux zones d'image. Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport :

$$C = \frac{L1 - L2}{L1 + L2} \qquad \text{(II. 3)}$$

Il existe plusieurs méthodes pour ajuster le contraste d'une image en utilisant Python et la bibliothèque OpenCV dans PyCharm. Voici un exemple de code pour appliquer une transformation de l'histogramme à une image pour ajuster son contraste :

```python
import cv2

#Chargement de l'image en niveaux de gris
img = cv2.imread("E:\sarah\master\memoire\Capture.PNG", cv2.IMREAD_GRAYSCALE)

# Application de la transformation de l'histogramme pour ajuster le contraste
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
img_clahe = clahe.apply(img)

# Affichage des résultats
cv2.imshow("Image originale", img)
cv2.imshow("Image avec contraste ajusté", img_clahe)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

L'affichage :



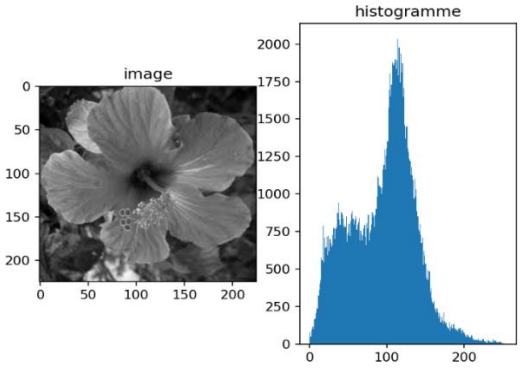
FigureII.13. Ajustement du contraste

II.5.5. Histogramme

Un histogramme est un graphique statistique permettant de représenter la distribution des intensités des pixels d'une image, c'est-à-dire le nombre de pixels pour chaque intensité lumineuse. Par convention un histogramme représente le niveau d'intensité en abscisse en allant du plus foncé (à gauche) au plus clair (à droite). **[20]**

Voici un exemple de code Python utilisant PyCharm pour tracer l'histogramme d'une image en niveaux de gris :

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Charger une image
img = cv2.imread('E:\sarah\master\memoire\img.jfif')
plt.subplot(1,2,1)
plt.imshow(img),plt.title('image')
plt.subplot(1,2,2)
plt.hist(img.ravel(),256,[0,255]),plt.title('histogramme')
plt.show()
```



FigureII.14. Histogramme

II.5.6.Égalisation d'histogramme

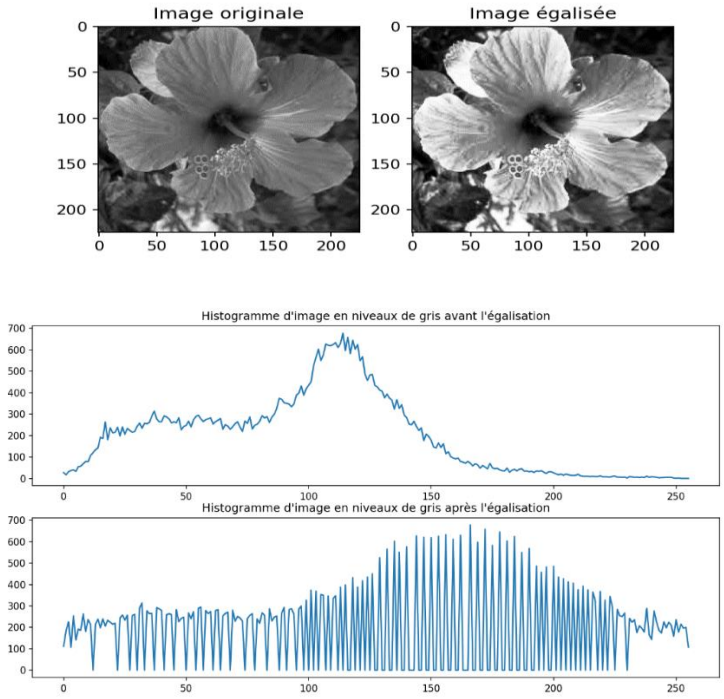
L'égalisation d'histogramme a pour but d'harmoniser la répartition des niveaux de luminosité de l'image, de telle manière à tendre vers un même nombre de pixel pour chacun des niveaux de l'histogramme. Cette opération vise à augmenter les nuances dans l'image. **[20]**

Voici un exemple de code Python utilisant PyCharm pour égaliser l'histogramme d'une image en niveaux de gris à l'aide d'OpenCV :

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

#Charger une image en niveaux de gris
img = cv2.imread('E:\sarah\master\memoire\img.jfif', cv2.IMREAD_GRAYSCALE)

# Calculer l'histogramme de l'image originale
hist_orig, bins = np.histogram(img.flatten(), 256, [0, 256])

# Égaliser l'histogramme de l'image
equalized_img = cv2.equalizeHist(img)

# Calculer l'histogramme de l'image égalisée
hist_eq, bins = np.histogram(equalized_img.flatten(), 256, [0, 256])
```

```
# Tracer l'histogramme de l'image originale
plt.subplot(2, 1, 1)
plt.plot(hist_orig)
plt.title("Histogramme d'image en niveaux de gris avant l'égalisation")

# Tracer l'histogramme de l'image égalisée
plt.subplot(2, 1, 2)
plt.plot(hist_eq)
plt.title("Histogramme d'image en niveaux de gris après l'égalisation")

plt.show()
```

Après l'exécution du programme, le résultat est illustré par la figure ci-dessous :



FigureII.15. Égalisation d'histogramme

Dans ce code, nous avons utilisé la fonction ***np.histogram ()*** de NumPy pour calculer l'histogramme de l'image originale et l'histogramme de l'image égalisée. Nous avons ensuite tracé les deux histogrammes à l'aide de la fonction ***plot ()*** de Matplotlib.

II.5.7. Morphologie mathématique

La morphologie mathématique fournit des outils pour toute la chaîne de traitement de l'image, du prétraitement (filtrage, amélioration du contraste) à la segmentation et à l'interprétation des scènes. L'une de leurs principales caractéristiques est leur nature non linéaire. Ils peuvent transformer des images, extraire des caractéristiques, des objets ou des

mesures (forme, taille, apparence...) grâce à une analyse associant les propriétés des objets eux-mêmes et les propriétés de contexte (voisinage local ou relations avec d'autres objets).

La morphologie mathématique se base sur la théorie des ensembles mathématiques pour décrire et manipuler les formes et les structures dans une image. Elle utilise des opérations telles que l'érosion, la dilatation, l'ouverture et la fermeture pour manipuler les formes et les structures dans une image**. [19]**

II.5.7.1. Erosion

Est l'opération d'érosion consiste à "rincer" les zones les plus sombres ou les plus claires de l'image en utilisant un noyau structurant spécifique. Elle permet de réduire les contours et les zones d'une image.

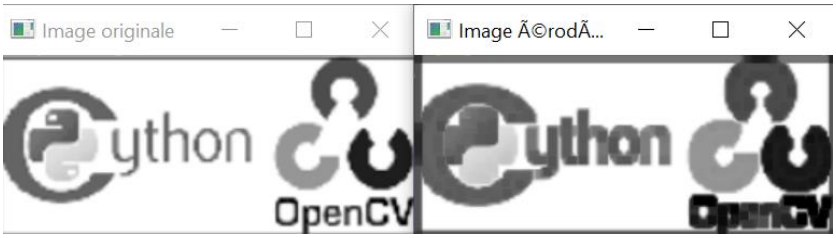
```python
import cv2
import numpy as np

# Charger une image
img = cv2.imread('E:\sarah\master\memoire\Capture.PNG', 0)

# Définir un noyau pour l'érosion
kernel = np.ones((5,5), np.uint8)

# Appliquer l'érosion
erosion = cv2.erode(img, kernel, iterations=1)
```

FigureII.16. Erosion

II.5.7.2. Dilatation

L'opération de dilatation consiste à "remplir" les zones sombres ou claires de l'image en utilisant un noyau structurant spécifique. Elle permet d'élargir les contours et les zones d'une image.

```python
import cv2
import numpy as np

# Charger une image
img = cv2.imread('E:\sarah\master\memoire\Capture.PNG', 0)

# Définir un noyau pour l'érosion
kernel = np.ones((5,5), np.uint8)

# Appliquer la dilatation
dilation = cv2.dilate(img, kernel, iterations=1)

# Afficher l'image d'origine et l'image dilatée
cv2.imshow('Image originale', img)
cv2.imshow('Image dilatée', dilation)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

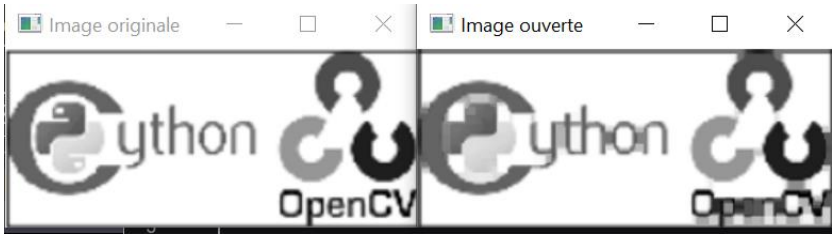


FigureII.17. Dilatation

II.5.7.3.L'ouverture

L'opération d'ouverture consiste à effectuer une érosion suivie d'une dilatation sur une image. Elle permet de supprimer les petites structures de l'image tout en conservant les structures plus grandes.

```python
import cv2
import numpy as np

# Charger une image
img = cv2.imread('E:\sarah\master\memoire\Capture.PNG', 0)

# Définir un noyau pour l'érosion
kernel = np.ones((5,5), np.uint8)

# Appliquer l'ouverture
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)

# Afficher l'image d'origine et l'image ouverte
cv2.imshow('Image originale', img)
cv2.imshow('Image ouverte', opening)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



FigureII.18. Ouverture

II.5.7.4. La fermeture

L'opération de fermeture consiste à effectuer une dilatation suivie d'une érosion sur une image. Elle permet de remplir les petites lacunes dans les structures de l'image tout en conservant les structures plus grandes.

```python
import cv2
import numpy as np

# Charger une image
img = cv2.imread('E:\sarah\master\memoire\Capture.PNG', 0)

# Définir un noyau pour l'érosion
kernel = np.ones((5,5), np.uint8)

# Appliquer la fermeture
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)

# Afficher l'image d'origine et l'image après la fermeture
cv2.imshow('Image originale', img)
cv2.imshow('Image après fermeture', closing)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

FigureII.19. Fermeture

II.6. Conclusion

Dans ce chapitre, nous avons présenté une brève introduction aux concepts liés au domaine du traitement d'images. Nous avons vu quelques modifications intéressantes que nous pouvons appliquer sur une image en appliquant une variété de filtres qui utilisent des algorithmes mathématiques, et nous en avons compris les concepts de base.

On peut dire que l'objectif du traitement d'images est de convertir l'image brute en une image traitée de haute qualité, qui peut être archivée ou utilisée dans diverses applications.

Dans cet objectif, nous avons donné une idée sur le traitement d'images et son processus en utilisant Python avec la bibliothèque Open CV.

Nous pouvons, à présent, aborder au prochain chapitre, la détection et la classification des objets par l'apprentissage approfondi implémenté dans la carte Raspberry pi 4.

Chapitre III

Résultats et interprétations

III.1. Introduction

La carte Raspberry Pi 4 est un mini-ordinateur très performant qui peut être utilisée pour le traitement d'images en raison de ses capacités de calcul et de ses interfaces de connexion pour des caméras. Cette carte peut être utilisée dans diverses applications de traitement d'images telles que la surveillance vidéo, l'analyse d'image, la vision par ordinateur, la reconnaissance faciale, la reconnaissance d'objet, la robotique, etc.

Dans ce chapitre nous allons présenter les applications de Raspberry pi 4 en traitement d'images, et nous allons présenter la préparation de système d'exploitation, l'installation du Raspbian.

III.2. Connexion des composants d'un Raspberry Pi

Les éléments essentiels (accessoires) dont nous aurons besoin pour notre Raspberry Pi et comment les connecter tous pour le configurer et le faire fonctionner.

- Un Raspberry pi 4
- Une alimentation (USB power supply)
- Une carte micro SD
- Un clavier et une souris
- Un câble micro HDMI
- Un Ecran

FigureIII.1. Le matériel

III.2.1. Une alimentation (USB power supply)

Une alimentation de courant de sortie maximum 2,5 A et une tension de 5 V avec un connecteur micro USB. L'alimentation officielle du Raspberry Pi est le choix recommandé, car elle peut faire face aux demandes d'alimentation à commutation rapide du Raspberry Pi.

III.2.2. Une Carte micro SD

Une carte microSD est une carte mémoire spécifiquement conçue pour être utilisée avec les ordinateurs monocartes Raspberry Pi.

La carte microSD est utilisée comme support de stockage principal pour le système d'exploitation et les fichiers du Raspberry Pi. Elle est recommandée d'utiliser une carte microSD au moins de 16 Go.

III.2.3. Un câble HDMI

Un câble HDMI (High-Definition Multimedia Interface) est un type de câble utilisé pour transmettre des signaux audio et vidéo numériques de haute qualité entre différents appareils électroniques. Il est largement utilisé pour connecter des appareils tels que des téléviseurs, des lecteurs Blu-ray, des consoles de jeu, des ordinateurs et des récepteurs audio/vidéo.

III.3.Installation du système d'exploitation [21]

Raspbian est un système d'exploitation libre basé sur la distribution GNU/Linux Debian, et optimisé pour le plus petit ordinateur du monde, la Raspberry Pi.

La Raspberry Pi est une framboise merveilleuse, mais elle reste dotée d'une puissance inférieure à celle d'un ordinateur moderne. Il est donc préférable d'utiliser un système optimisé pour elle, ce qui est le cas de Raspbian.

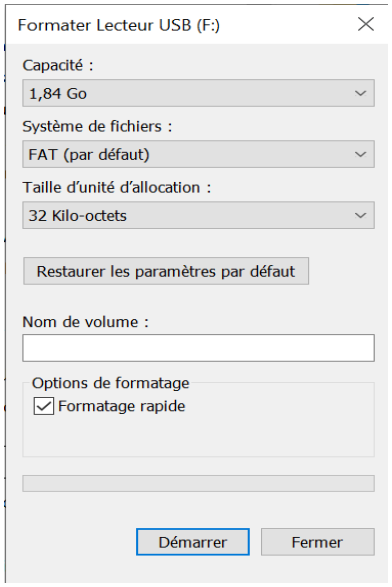
Raspbian est sans doute la distribution la plus utilisée pour les Raspberry, et bénéficie donc d'une communauté large et active.

III.3.1. Préparation du système d'exploitation sur micro SD

III.3.1.1. Formater la carte SD

La carte SD est ce qui fera office de mémoire morte sur le Raspberry Pi, tout comme un disque dur d'ordinateur. C'est donc là-dessus que sera stocké le système d'exploitation, et à priori, nos documents, photos, musiques, vidéos...

Même si la carte SD est neuve, le mieux est de la formater avant de copier les fichiers d'installation **[22].**



FigureIII.2. Préparation de la carte SD

III.3.1.2. Installer Raspbian sur la carte SD

Pour télécharger le système d'exploitation, qui est en fait une distribution de Linux qui s'appelle Raspbian. Cette distribution est une version modifiée de Debian, qui est utilisée sur la grande majorité des serveurs de sites Internet. **[22]**

1. Pour obtenir le système d'exploitation Raspbian imager il faut accéder à la page du site officiel du Raspberry Pi :

https://www.raspberrypi.org/downloads/raspbian/

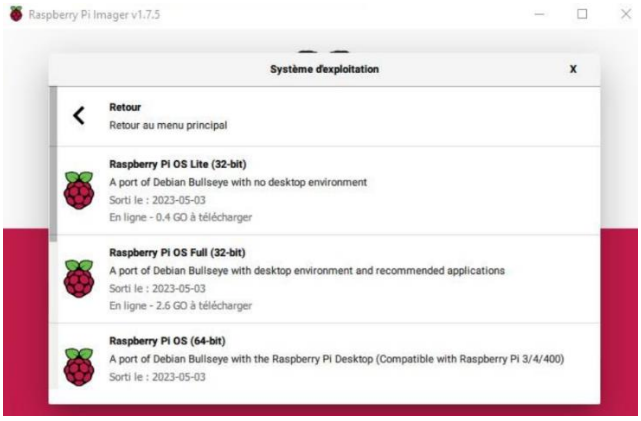
2. Insérer la carte SD dans le lecteur d'ordinateur. Utiliser un outil comme Etcher (disponible sur www.balena.io/etcher) pour flasher l'image du système d'exploitation sur la carte micro SD.

3. Connecter les périphériques : le clavier, la souris, l'écran et l'adaptateur d'alimentation au Raspberry Pi.

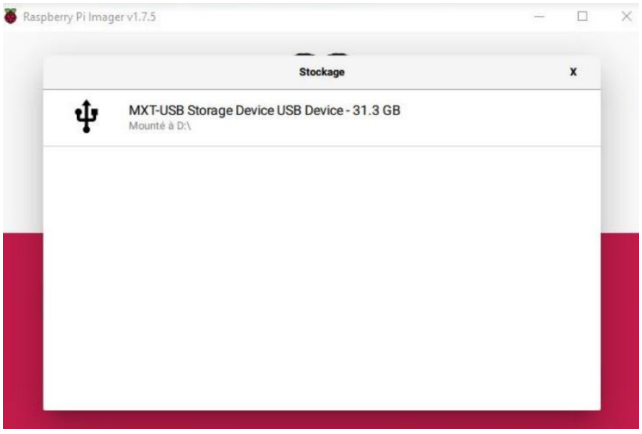
4. Démarrer le Raspberry Pi

FigureIII.3. Démarrage de Raspberry pi

5. Choiser Raspbian OS et la carte SD.



FigureIII.4. Choix de système d'exploitation

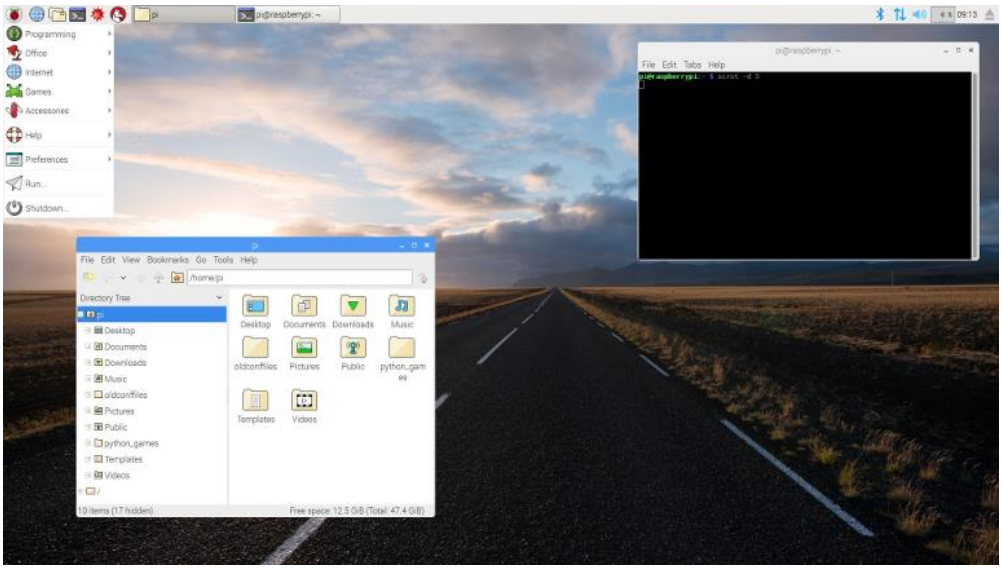


FigureIII.5. Sélection de carte SD

III.3.2. Première configuration

III.3.2.1. Configuration initiale

Une fois le Raspberry Pi démarré, nous serons invité à effectuer une configuration initiale. On peut choisir de modifier le mot de passe par défaut, d'ajuster les paramètres régionaux, de configurer le Wi-Fi et d'autres options.



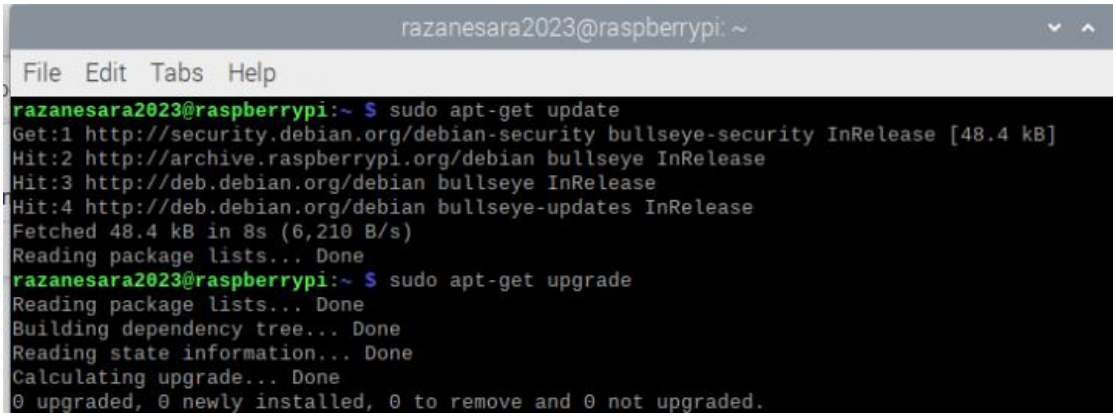
FigureIII.6.Configuration initiale

III.3.2.2. Mise à jour du système d'exploitation

Après avoir terminé la configuration initiale, il faut ouvrir un terminal sur le Raspberry Pi et exécutez les commandes suivantes pour mettre à jour le système d'exploitation :

sudo apt update : pour mettre à jour les listes de paquets disponibles à partir des dépôts (repositories) configurés sur le système.

sudo apt upgrade : pour télécharger et installer les mises à jour



FigureIII.7. Mise à jour de système

Une fois la mise à jour terminée, nous pouvons donc commencer à explorer les fonctionnalités de notre Raspberry Pi. On peut désormais installer des logiciels supplémentaires, configurer des serveurs, programmer des projets, etc…

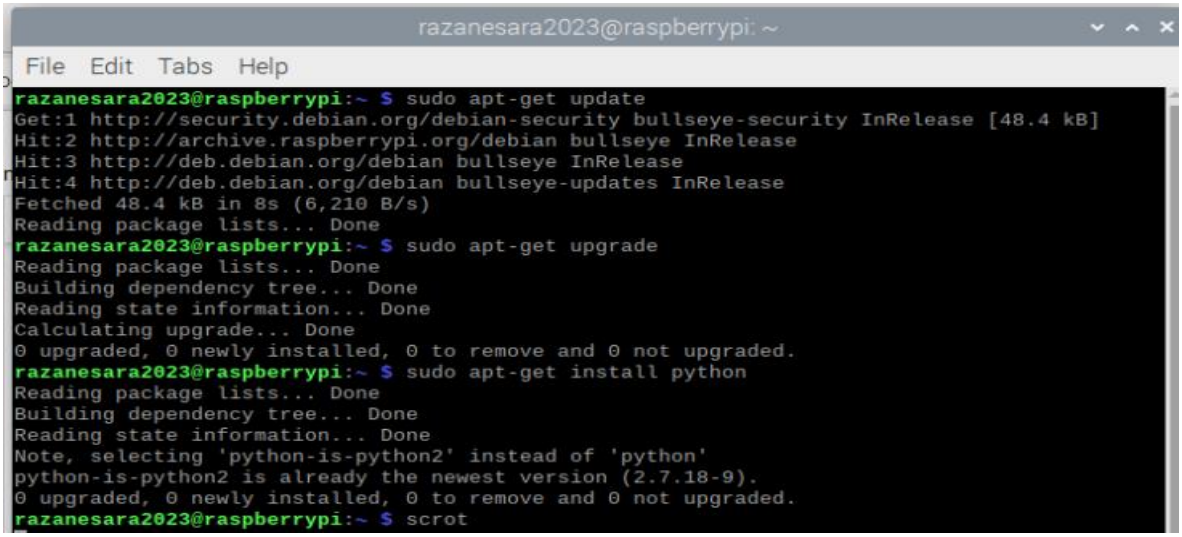
III.3.2.3. Installation Python sur le Raspberry Pi

Le Raspberry Pi dispose généralement de Python préinstallé. Pour vérifier, on doit taper la commande suivante dans le terminal :

python --version

Si Python est déjà installé, donc sa version s'affichée. Sinon, pour installer Python, on exécute la commande suivante dans le terminal :

sudo apt-get install python3



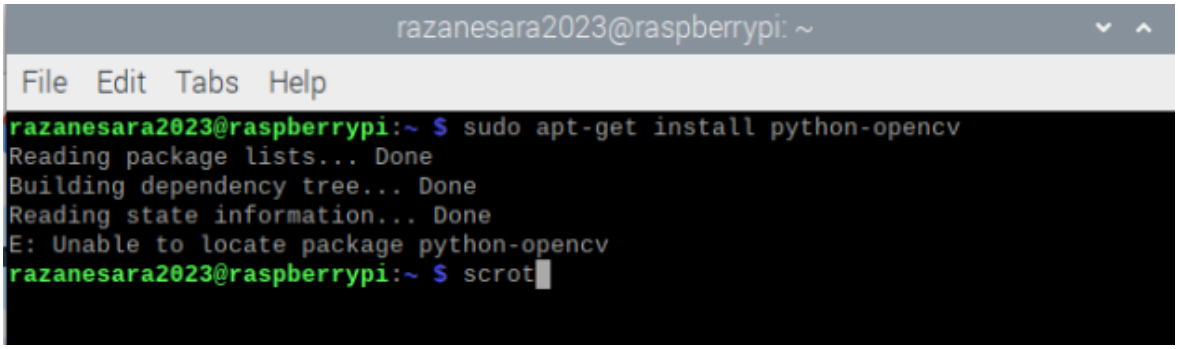
FigureIII.8. Installation de python

III.3.2.4. Installation OpenCV

OpenCV est une bibliothèque très complète et incroyablement puissante pour la vision par ordinateur (en temps réel), y compris la détection d'objets, le suivi de mouvement et l'étalonnage de la caméra. **[23]**

L'installation de opencv ce fait par cette commande :

Sudo apt-get install python3-opencv



FigureIII.9.Installaion d'Opencv

Ou bien : ***Pip3 install opencv-python***

III.4. Les applications de traitement d'images

III.4.1. La détection des objets

La détection d'objets est une technique de vision par ordinateur qui a connu un changement révolutionnaire rapide, cette technique fait la combinaison de la classification et de la localisation d'objets. **[23]**

III.4.1.1. Détection d'objet YOLOv5

III.4.1.1.1.YOLO

YOLO est un algorithme de détection d'objets en temps réel à la pointe de la technologie créé en 2015 et a été pré-formé sur l'ensemble de données COCO. Il utilise un seul réseau de neurones pour traiter une image entière. L'image est divisée en régions et l'algorithme prédit les probabilités et les boîtes englobantes pour chaque région. **[24]**

YOLO est bien connu pour sa rapidité et sa précision et il a été utilisé dans des projets d'intelligence artificielle sur Raspberry Pi, tels que la classification d'images, la détection de visages, la détection de gestes, etc.

III.4.1.1.2. YOLOV 5

YOLOv5 est un algorithme de détection d'objet qui s'appuie sur les versions précédentes de la famille de modèles YOLO (You Only Look Once). Il a été développé pour améliorer les performances des modèles de détection d'objets en termes d'exactitude et de vitesse.

III.4.1.1.3. Les modèles YOLOV5

YOLOv5 est sorti avec cinq tailles différentes : **[24]**

n pour les modèles de très petite taille (nano).

s pour modèle de petite taille.

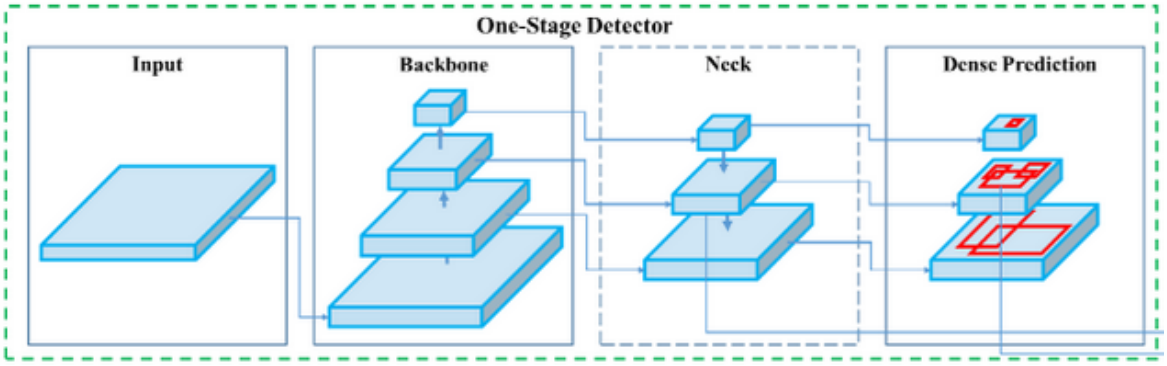
m pour modèle de taille moyenne.

l pour modèle de grande taille

x pour le modèle de très grande taille

III.4.1.1.4. L'architecture de YOLOv5

L'architecture de YOLOv5 est basée sur un réseau de neurones convolutifs appelé Darknet. Cependant, par rapport aux versions précédentes de YOLO, YOLOv5 a été repensé pour être plus léger, plus rapide et plus précis. Voici une description de l'architecture de YOLOv5 :



FigureIII.10. Architecture de YOLOV5

1. **Backbone :**

YOLOv5 utilise un backbone composé de blocs de convolution pour extraire des caractéristiques d'une image. Le backbone est basé sur l'architecture CSPDarknet53, qui utilise des blocs CSP (Cross-Stage Partial connections) pour améliorer la représentation des caractéristiques.

2. **Neck :**

Le "neck" de YOLOv5 est composé de plusieurs blocs de convolution qui permettent de fusionner les caractéristiques de différentes échelles spatiales. Cela aide à détecter des objets de différentes tailles dans une image.

3. **Tête (Head) :**

La tête du réseau est responsable de la prédiction des coordonnées et des classes des objets détectés. Yolov5 utilise une tête de réseau appelée YOLOv5Head, qui est composée de couches convolutives et de couches linéaires (fully connected) pour prédire les bounding boxes (boîtes englobantes) et les classes des objets.

III.4.1.5. Installation de yolov5

Etape1 : Clonage du dépôt YOLOv5

Dans le terminal, on doit exécuter les commandes suivantes pour cloner le dépôt YOLOv5 depuis GitHub :

git clone https://github.com/ultralytics/yolov5

cd yolov5

Etape2 : Installation des dépendances Python

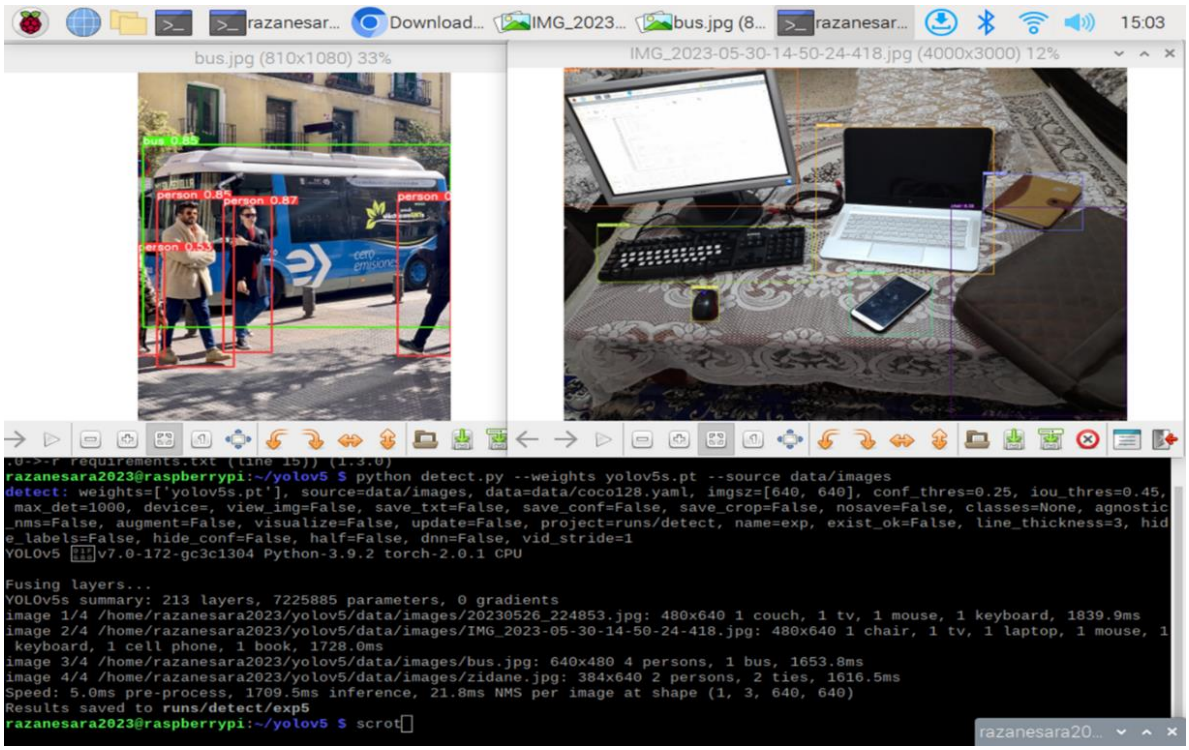
Installez les dépendances Python nécessaires en exécutant la commande suivante :

pip install -r requirements.txt

Etape3 : Utilisation de YOLOv5

On est maintenant prêt à utiliser YOLOv5 sur notre Raspberry Pi. Nous pouvons exécuter un script Python pour détecter des objets dans des images ou des vidéos. Par exemple, pour détecter des objets (ou des classes) dans une image, on doit exécuter la commande suivante :

python3 detect.py --source path/to/our/image.jpg



FigureIII.11. Détection d'objet par YOLOV5

D'après les résultats obtenus par **(**FigureIII.11) nous avons vu que la performance de model yolov5 on détection de diffèrent objet dans les images est plus précis, et la vitesse de détection de YOLOv5 est également impressionnante. Grâce à des optimisations et à l'utilisation d'architectures de réseaux neuronaux plus légères, YOLOv5 est capable d'exécuter des détections en temps réel.

D'après la figure ci-dessus, nous voyons que YoloV5 est capable de détecter plusieurs objets de différentes classes dans une seule image.

Pour effectuer la détection d'objets en temps réel sur une vidéo, on doit s'assurer que nous avons déjà suivi les étapes d'installation de YOLOv5 sur notre Raspberry Pi, et Placez la vidéo sur notre Raspberry Pi.

La détection d'objets sera effectuée sur chaque image de la vidéo, et les résultats seront affichés à l'écran en temps réel.

III.4.2. Entraînement

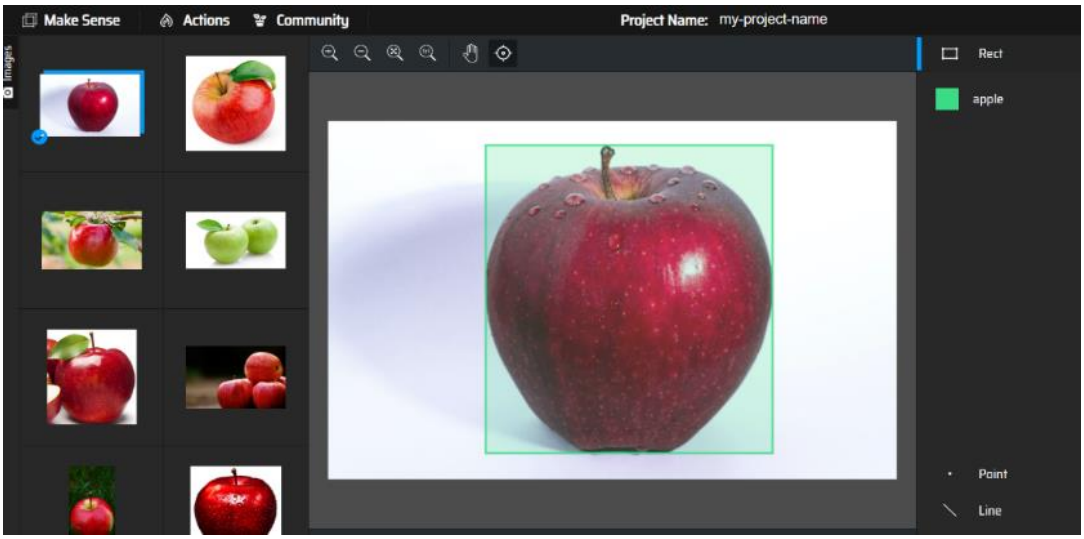
Pour un nouvel entraînement de YOLO, nous utilisons une petite base de données des fruits (8 classes), L'entraînement de YOLOv5 implique plusieurs étapes clés et voici un aperçu du processus d'entraînement :

III.4.2.1. Préparation des données d'entrainement

Tout d'abord, nous devons préparer notre ensemble de données pour l'entraînement. Nous avons besoin des images et des étiquettes (labels), on utilise 20 images pour L'entraînement et 10 images pour validation pour chaque classe. C'est-à-dire 240 images en total.

FigureIII.12. Préparation des données d'entrainement

Nous utilisons le site web ***makesense.ai*** pour la détection d'objet. En premier lieu on fait appel aux images d'entraînement, ensuite nous devons donner les noms des classes et démarrer notre projet.



FigureIII.13. Etiquetage des données (labelling)

Les performances de détection d'objets dépendent fortement de la qualité des données d'entraînement. Des ensembles de données bien étiquetés et représentatifs sont nécessaires pour obtenir les meilleurs résultats.

Dans l'étiquetage (labelling) on doit être prudent dans l'exécution de ce processus, car il influe sur les résultats obtenus par le processus d'entraînement afin d'obtenir une meilleure classification.

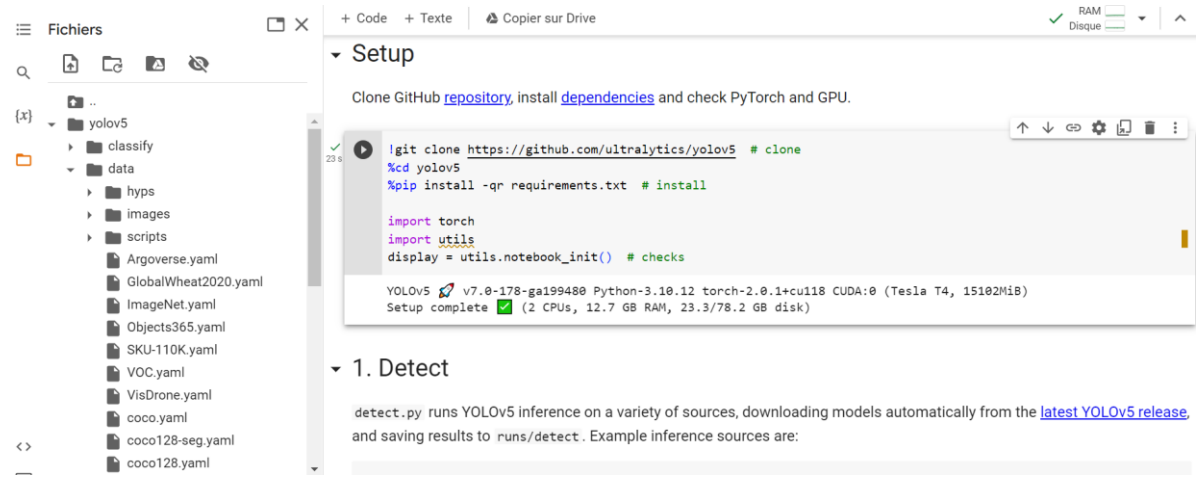
III.4.2.2. Configuration du modèle

III.4.2.2.1. L'Environnement Google Colab

Google Colab est un produit de Google, c'est une plateforme en ligne qui offre un environnement de développement gratuit basé sur Jupyter Notebook et qui prend en charge l'exécution de code Python. Il prend en charge de nombreuses bibliothèques d'apprentissage automatique populaires et de haut niveau qui peuvent être facilement chargées dans notre notebook. **[23]**

III.4.2.2.2. Configuration de Google Colab

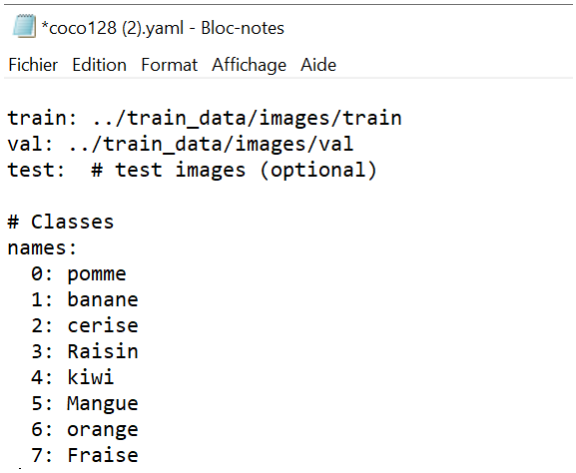
La première instruction à exécuter est d'appelé yolov5, cette étape est illustrée par la figure ci-dessous :



FigureIII.14. Configuration de Google Colab

Ensuite on doit ajouter notre base de données déjà étiquetée sous forme d'un fichier compresser zip et la télécharger dans le Colab.

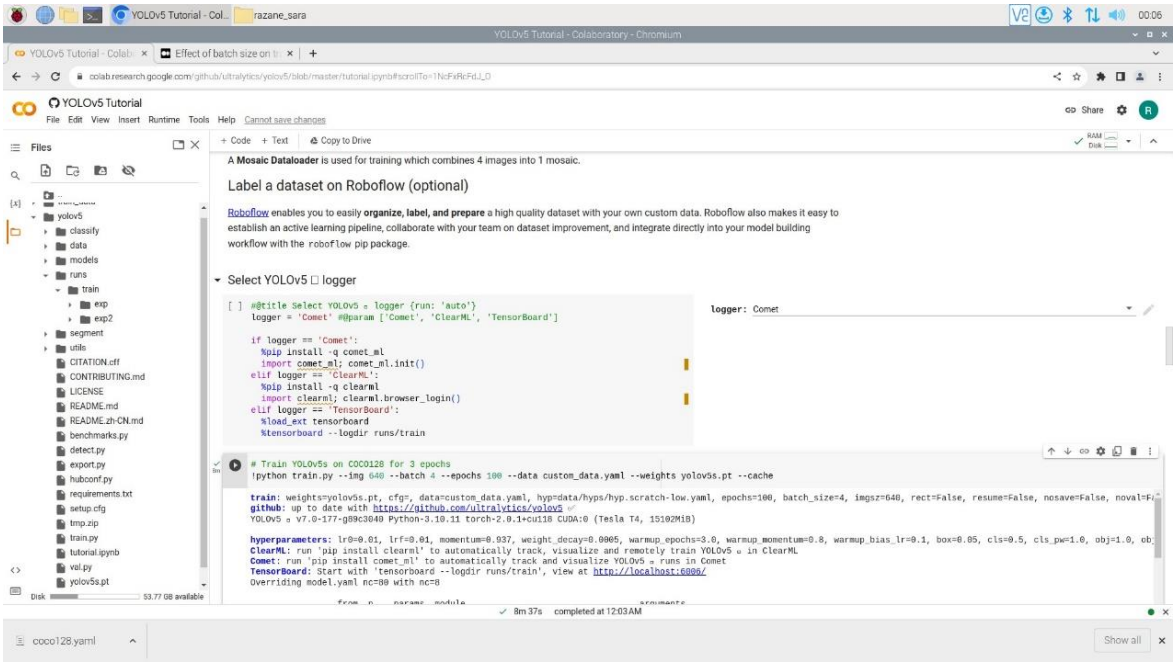
Finalement on doit ainsi ajouter un fichier de type ***cutom_data.yaml*** qui contient le chemin des deux dossiers d'apprentissage (entrainement et validation), et les noms des huit classes à détecter. Voir la figure ci-dessous :



FigureIII.15. Fichier yaml pour les classes

III.4.2.3. L'entraînement sur Google colab

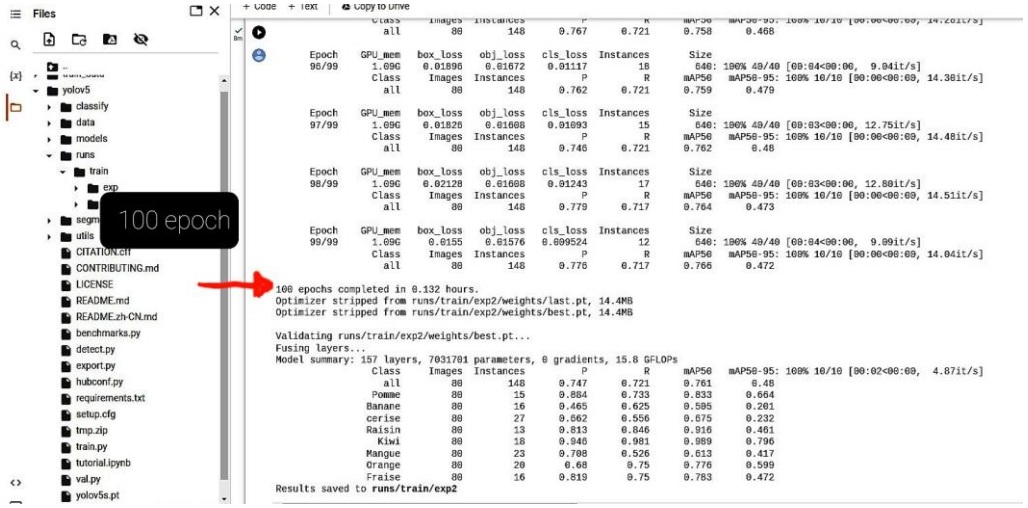
Une fois que nous avons effectué toutes les étapes précédentes, nous sommes prêts à lancer l'entraînement. La figure III.16 illustre la commande python nécessaire pour commencer l'entraînement.

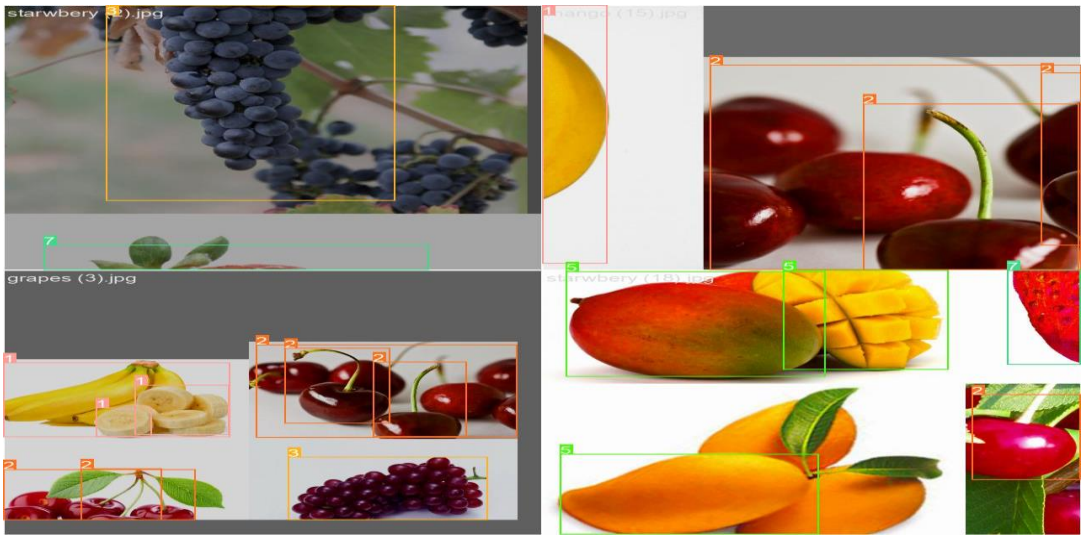


FigureIII.16. L'entraînement sur Google colab

Nous avons effectué plusieurs essais on changeant les paramètres d'entrainement et notre choix s'oriente 100 époch et un batch de 4, afin d'avoir plus de précision.

Les figures suivantes illustrent ce procédé et les résultats obtenus :





FigureIII.17. Résultats de l'entraînement

D'après les conséquences obtenues par **(**FigureIII.17) nous avons vu un bon résultat parce qu'il a reconnu les fruits dans des pourcentages satisfaisants, mais pour de préférables résultats les conditions suivantes doivent être fournies :

- Introduire de nombreuse probabilité pour assurer la qualité de la reconnaissance des objets.

- Réduire la taille de batch, nous permettra d'obtenir une précision élevée.

III.4.2.4.Validation

C'est un processus essentiel pour garantir la qualité et la fiabilité des données.

FigureIII.18. Résultats de validation

Le modèle à bien fonctionné sur la majorité des images de **(**FigureIII.18), il y a eu quelques cas d'erreur de classement, comme la 2^{éme} image de cerise qui montre une erreur de classification.

Pour éviter une telle erreur de classification on doit augmenter le nombre d'images d'entraînement en tenant compte de plusieurs d'utiliser des images qui contient plusieurs classes dans une seule image.

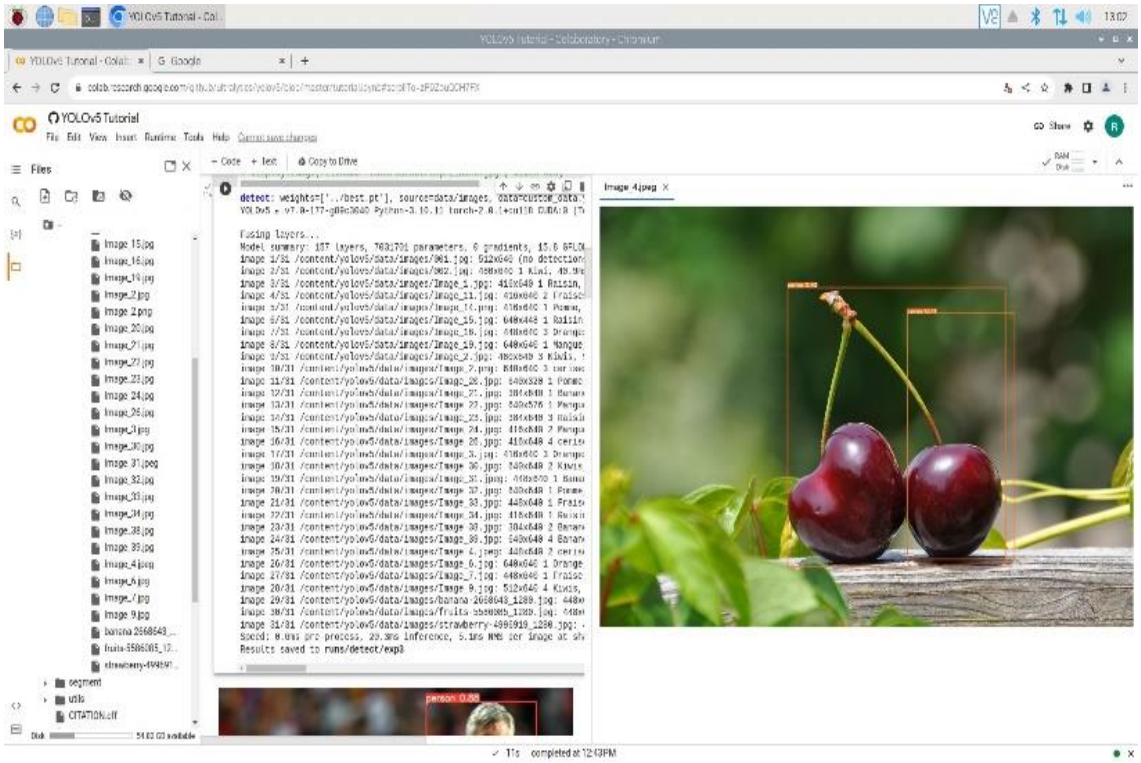
III.4.2.5. Détection

Une fois le modèle entraîné, il peut être utilisé pour détecter des objets (classes) dans de nouvelles images. Pour cela, on doit utiliser le fichier nommé ***best.pt*** qui contient les poids après l'entrainement déjà terminé. Voir la figure III.19 :

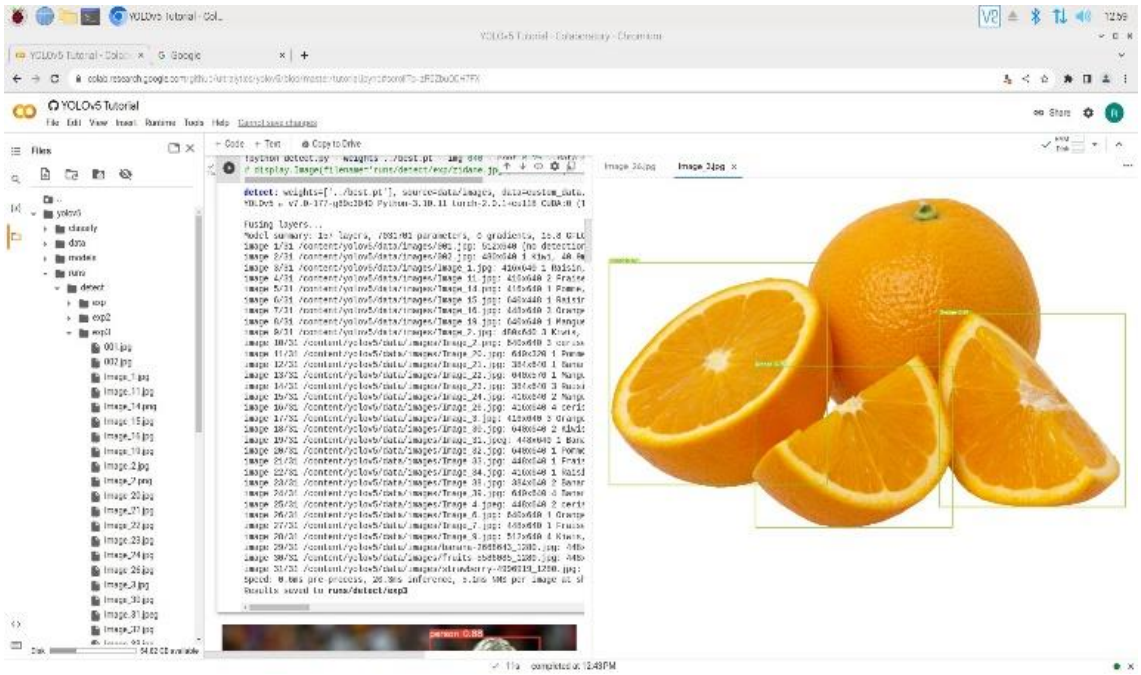
FigureIII.19. Instruction python pour la détection

III.4.2.5.1.Les résultats de détection

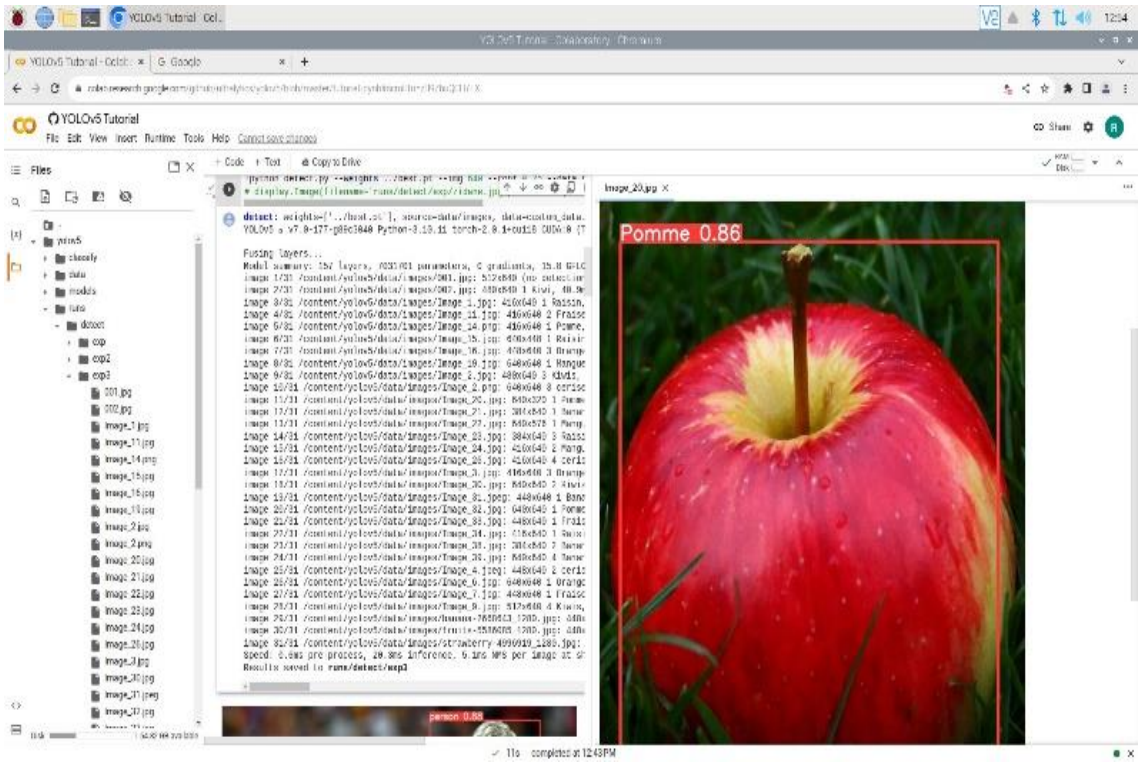
Les figures ci-dessous représentent quelques résultats de détection et de classification pour un ensemble de nouvelles images :



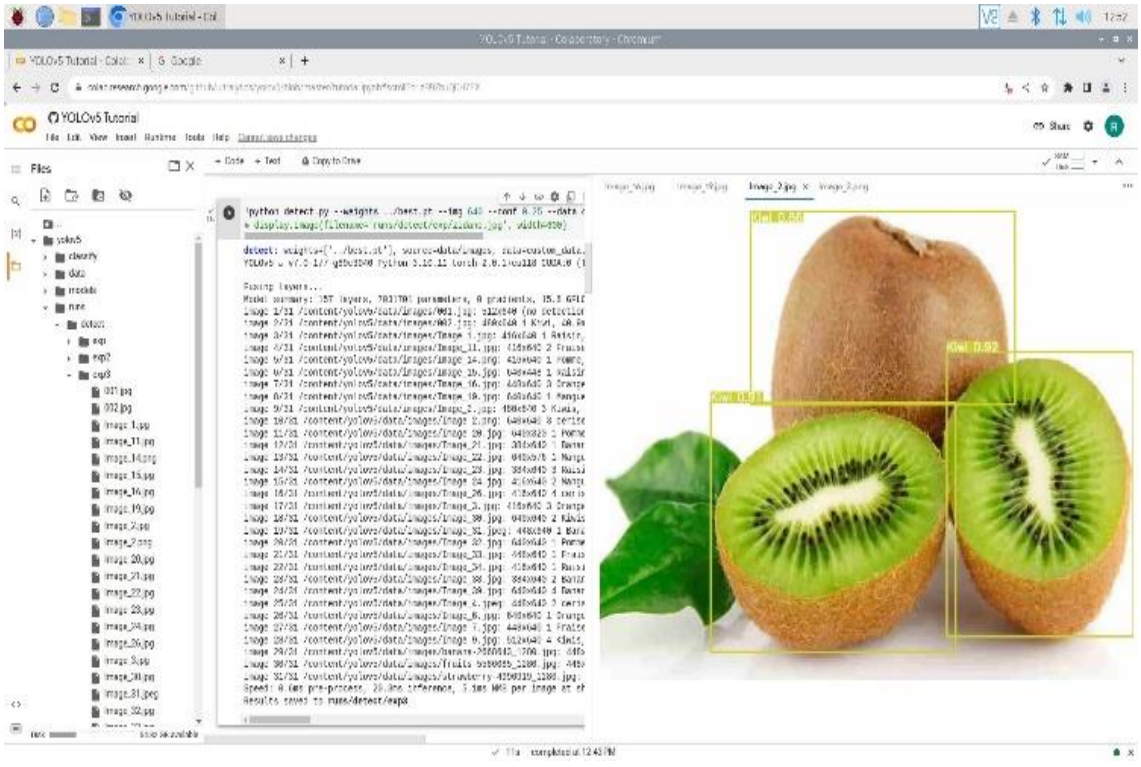
FigureIII.20. Détection de la classe cerise



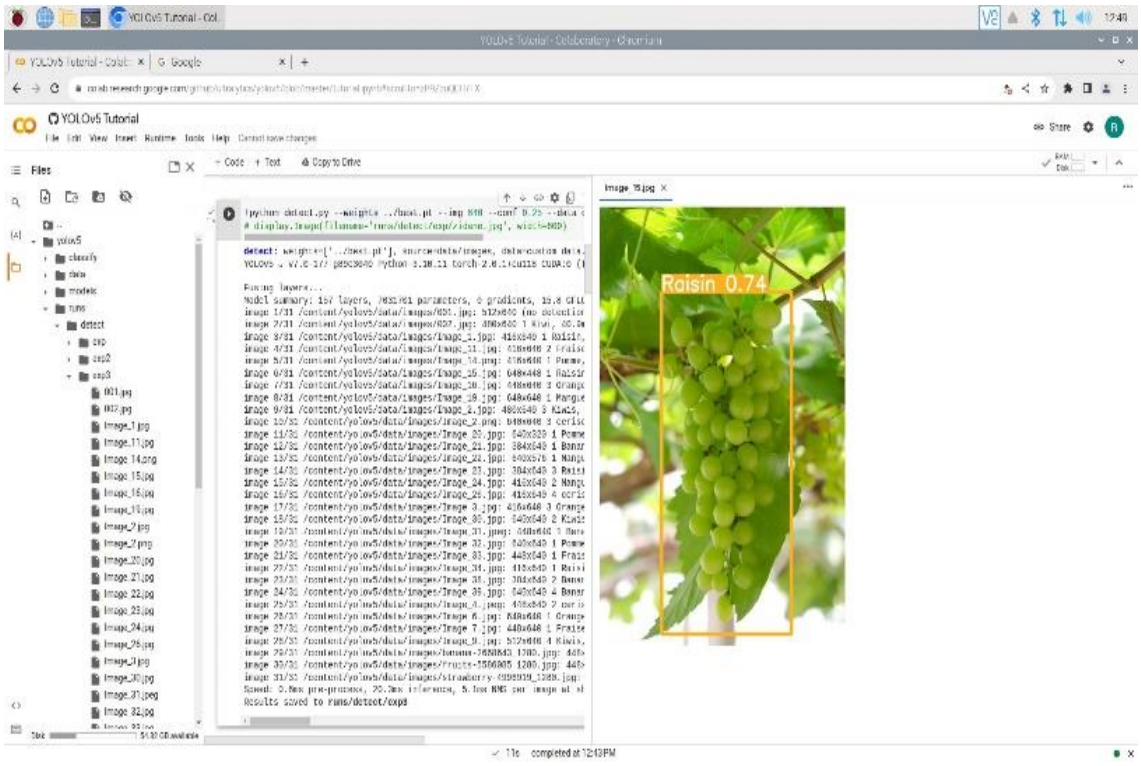
FigureIII.21. Détection de la classe orange



FigureIII.22. Détection de la classe pomme



FigureIII.23. Détection de la classe kiwi



FigureIII.24. Détection de la classe raisin

D'après les résultats obtenus par les figures ci-dessus, nous voyons clairement que notre modèle est bien entrainer afin de connaître des fruits (c-à-d détection et classification).

En terme de précision de classification on a pu obtenu les résultats suivants :

Le kiwi 92 %, orange 74 %, cerise 93 %, raisin 74 %, fraise 89 % et pomme 86 %.

III.5. Conclusion

Dans ce chapitre, nous avons présenté les différentes étapes à suivre afin de configurer la carte Raspberry Pi4 pour être prêt à utiliser.

Ensuite, nous avons définis les principaux concepts des réseaux de neurones convolutifs (CNN) en termes simples, en se basant sur l'architecture Yolov5 qui sera bien exploité pour notre application.

Egalement, on a présenté les différentes étapes à suivre pour la préparation de la base de données qui contient 240 images.

On a pu conclure qu'une bonne base de donnés bien étiquetés et un bon choix des paramètres d'entrainement, nous conduira à obtenir des taux de précision intéressants.

Aussi, le Raspberry Pi 4 offre une plateforme flexible et abordable pour le traitement d'images, et il est possible d'apprendre et de créer des projets de traitement d'images de manière accessible et passionnante.

Conclusion générale

Conclusion Générale

Le Raspberry Pi est un mini-ordinateur mono carte à processeur ARM. Cet ordinateur, de la taille d'une carte de crédit, est destiné à encourager l'apprentissage de la programmation informatique ; il permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux, notamment Debian, et des logiciels compatibles. Mais il fonction également avec le système d'exploitation Microsoft Windows.

Le traitement d'images est une discipline de l'information et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information.

Le langage Python devient un bon choix pour de telles tâches de traitement d'images. Cela est dû à sa popularité croissante en tant que langage de programmation scientifique et à la disponibilité gratuite de nombreux outils avancés de traitement d'images dans son écosystème.

La carte Raspberry Pi 4 peut être exploitée pour des applications en traitement d'images en raison de ses capacités de calcul et de ses interfaces de connexion pour des caméras. Cette carte peut être utilisée dans diverses applications de traitement d'images telles que la surveillance vidéo, l'analyse d'image, la vision par ordinateur, la reconnaissance faciale, la reconnaissance d'objet, la robotique, etc.

Afin de réalisé notre projet, nous avons pu exploiter cette carte pour l'implémentation d'une chaine de traitement d'images basée sur l'apprentissage approfondi (Deep learning) afin de réaliser en premier lieu une détection des objets d'intérêt puis à la classification de ces objets.

Pour les taches décrites précédemment, nous avons exploité l'algorithme de détection d'objet YOLOv5 appliqué à l'entrainement d'une base de données qui contient 240 images de huit classes des fruits. Le choix de YOLOv5 est basé à son architecture qui nous permet d'exécuter des détections en temps réel.

D'après les résultats obtenus, nous voyons clairement que le modèle est bien entrainer afin de connaître des fruits (détection et classification). Car nous avons obtenu une bonne détection des objets d'intérêt et une précision de classification va de 74% pour la classe Orange à 93% de celle de cerise.

Un autre point à décrire, YOLOv5 est capable de détecter plusieurs objets de différentes classes dans une seule image.

Dans un futur travail nous estimons à implémenter différents algorithmes d'apprentissage approfondi dans d'autres cartes et même de réaliser une chaine complète de traitement d'images en temps réel ; allant de l'acquisition, traitement et puis l'affichage des résultats obtenus dans un écran.

Bibliographie :

[1] Julien Cartailler, Richard Rocca - Polimeni. Le Raspberry Pi : un nano-ordinateur au service de la science et de l'enseignement. Cahier des Techniques de l'INRA, 2016, pp.148-156. ffhal-02629562 .

[2] https://raspberrytips.fr/histoire-du-raspberry-pi/ [consulté : 3/2/2023.]

[3] MEHDAOUI Sara et MAATI Imane, Projet de fin d'études de Master en Instrumentation, Conception d'un système de sécurité à reconnaissance facial à l'aide d'un Raspberry pi, Université Aboubakr Belkaid-Telmcan.2020

[4] SI ABDELKADER Soumia et SOUFI Zineb Meriem, Projet de fin d'études de Master en Instrumentation, Conception et réalisation d'un système de contrôle routier à l'aide d'un Raspberry Pi, Université Aboubakr Belkaid-Telmcan.2021

[5] Koceila DOUKI et Ramdane CHERIK, Mémoire de fin d'études de Master Académique en télécommunication et réseaux, Conception et réalisation d'un système de surveillance d'une serre agricole avec une carte Raspberry PI 2, Université MOULOUD MAMMERI de TIZI-OUZOU.2017

[6] https://framboisepi.fr/comparatif-raspberry-pi/ [consulté : 1/2/2023.]

[7] Patrick Fuchs et Pierre Poulain, cours, Introduction à la programmation Python pour la biologie, Université de paris cite.2022

[8] Mohamadi Redha Badr Eddine et Laati Taha Mohamed Ayoub et Laksi Aymen, Rapport de fin d'études de licence en Electronique industrielle, Détection et lecteur de QR code avec Open CV, Université Mohamed el-Bachir el-Ibrahimi Bordj Bou Arreridj.2021

[9] Mr. Karim MEZZOUG, cours, Traitement et analyse des images numérique, Université Ibn Khaldoun – Tiaret.2020

[10] Mohammed Khamadja et Said Benierbah, cours, traitement d'images, Université des frères Mentouri (Constantine 1)

[11] Mr. Mourad Reguiegue, cours, Généralités sur le traitement d'images, Université Ammar Thelidji Laghouat.2022

[12] CHIKH Mohammed Tahar, Mémoire de fin d'études de Master en Informatique, Amélioration des images par un modèle de réseau de neurones, Université Abou-Bakr Belkaid –Tlemcen. 2011

[13] https://www.geeksforgeeks.org/setup-opencv-with-pycharm-environment/ [consulté : 13/3/2023.]

[14] https://xphilipp.developpez.com/articles/filtres/?page=page_3 [Consulté : 13/3/ 2023.]

[15] https://hal.archives-ouvertes.fr/hal-00512280v1. [Consulté: 13/3/2023.]

[16] BAROUDI ABDERREZEQ, LARABI NOUR EL ISLEM, mémoire de Licence en Informatique, Comparaison entre les différents filtres d'images, Université Abou Bakr Belkaid– Tlemcen.2012

[17] http://perso-telecom-paristech.fr/~maitre/BETI/filtrres_lin_nlin/introduction.html. [Consulté: 13/3/2023.]

[18] https://fr.wikipedia.org/wiki/Filtre_m%C3%A9dian [Consulté: 13/3/2023.]

[19] https://chat.openai.com/chat Consulté [: 15/3/2023.]

[20] https://web.maths.unsw.edu.au/~lafaye/CCM/video/traitimg.htm [Consulté: 17/3/2023.]

[21] Raspberry Pi & Raspbian - Premier site de tutoriels en France (raspberry-pi.fr) Consulté [: 4/4/2023.]

[22] https://raspberrypi-guide.github.io/programming/install-opencv [Consulté : 2/6/ 2023.]

[23] MESBAH Fethia, Mémoire de Fin d'Etudes Master Informatique, Détection d'objets par Deep Neural Network à l'aide du modèle YOLO en temps réel, Université 8 Mai 1945 –Guelma.

[24] https://iq.opengenus.org/yolov5/ [Consulté : 2/6/2023.]