Université de Picardie Jules Verne UFR des sciences

Licence 3 informatique

Rapport de Projet S6 L3 INFORMATIQUE

« Reconnaissance de texte »

Réalisé par :

YAFINE Mohamed & EL BOUAD AYOUB

Projet encadré par :

MME. Soufia Bennai

M. LAZURE Dominique

Année universitaire : 2021/2022

SOMMAIRE

Table des illustrations

I. INTRODUCTION

Dans le cadre de notre troisième année d'études dans la spécialité Informatique à l'université Picardie Jules Verne Amiens nous avons eu l'opportunité de choisir comme projet : la reconnaissance de texte via le traitement d'Image/vidéo. La plupart des activités humaines reposent sur une faculté importante de notre cerveau pour voir distinguer entre les objets. Par exemple, reconnaître une personne dans une image ou de distinguer entre un rectangle qu'un cercle. Ce projet a pour but d'automatiser les tâches habituellement effectuées par l'homme. Et aussi de découvrir des nouvelles technologies et d'algorithmes.

Ce rapport contient l'ensemble des éléments du projet. D'un point de vue technique tout d'abord, nous présenterons le cahier des charges avec les modifications que nous avons fait sur la version ancienne (surtout dans l'environnement de travail), La deuxième partie de ce rapport a pour objectif d'expliquer certaines parties du code, ainsi que les résultats des différents tests effectués

II. Description du projet

1) Description Générale

a) Image

Une **image** est une représentation visuelle, voire mentale, de quelque chose (objet, être vivant ou concept). Elle peut être naturelle (ombre, reflet) ou artificielle (sculpture, peinture, photographie), visuelle ou non, tangible ou conceptuelle (métaphore), elle peut entretenir un rapport de ressemblance directe avec son modèle ou au contraire y être liée par un rapport plus symbolique.[1]

b) Vidéo

Une vidéo est une succession d'images à une certaine cadence. L'œil humain a comme caractéristique d'être capable de distinguer environ 20 images par seconde. Ainsi, en affichant plus de 20 images par seconde, il est possible de tromper l'œil et de lui faire croire à une image animée.[2]

c) Détection d'objets

En vision par ordinateur on désigne par détection d'objet une méthode permettant de détecter la présence d'une instance (reconnaissance d'objet) ou d'une *classe d'objets* dans une image numérique. Une attention particulière est portée à la détection de visage et la détection de personne[3].

→ Exemple :

Figure 1 : exemple de détection des objets

d) Reconnaissance de texte, forme, couleurs

La reconnaissance de texte, forme, couleurs est une technique qui permet d'identifier les différents éléments (formes, texte, couleurs) constituant dans une image donnée en entrée [4].

→ Exemple :

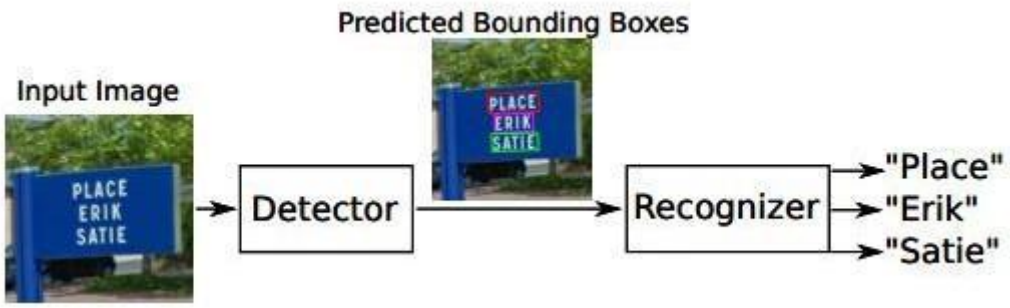


Figure 2 : exemple de détection de texte

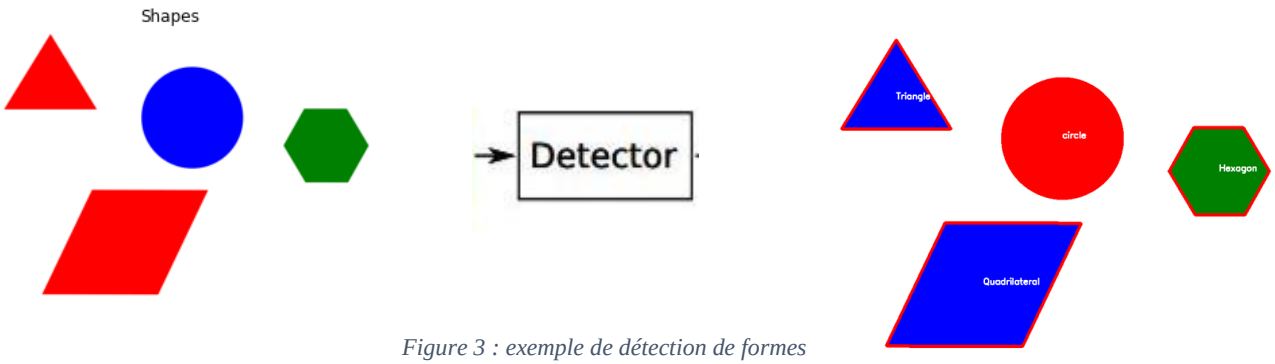


Figure 3 : exemple de détection de formes

2) Description de l'utilisateur

Les utilisateurs qui vont utiliser notre modèle doivent impérativement être francophones (une personne qui parle français) car dans notre projet on va utiliser la langue française.

Cependant, le projet ne sera pas adapté aux personnes malvoyantes, en effet, le temps disponible ne permet pas la mise en place d'une solution adapté.

III. ANALYSE DE L'EXISTANT

Une multitude de projets de ce type qu'ils soient open-source ou non sont disponibles sur internet, nous allons en présenter quelques-uns :

- **Amazon Rekognition** [5] offre des fonctionnalités de reconnaissance d'image (CV) pré-entraînées et personnalisables pour extraire des informations de vos images et vidéos, parmi ces fonctionnalités :
    - Détection des objets
    - Reconnaissance faciale
    - Analyse faciale
    - Texte au sein d'images

→ Les technologies utilisées sont : JAVA, .NET, Kotlin, Python.

- **IBM Image Détection** [6] : est un outil qui permet aux utilisateurs d'identifier automatiquement les sujets et les objets contenus dans l'image, ainsi que d'organiser et de classer ces images en catégories logiques, parmi ces fonctionnalités on cite :
    - Analyse des images de lieux, des objets, des visages, des couleurs et des aliments.
    - Isoler les marques à l'intérieur d'une image

→ Les technologies utilisées sont : Python, Scala…

- **Google Lens** [7] **:** L'objectif principal de Google Lens est, grâce à l'appareil photo d'un smartphone, d'afficher des résultats de recherche et des informations pertinentes lorsqu'un utilisateur prend une photo de quelque chose, d'un objet, d'un bâtiment ou bien d'un texte, parmi ces fonctionnalités :
    - Traduction de mots et prononciation de texte.
    - Transformez les notes manuscrites en texte.
    - Détection des objets (Animal, Plants,).

Notre projet sera une application avec un code source libre, il sera possible pour l'utilisateur de détecter des objets, textes, formes, couleurs dans une image /vidéo.

IV. Les moyens et contraintes

1) Les moyens

a) Moyens humains :

Le projet sera réalisé par un groupe de deux étudiants sous l'encadrement d'un enseignant.

On estime une charge de travail personnelle presque de 42 jours pour chaque membre de groupe, cette charge de travail comprend la conception, le développement et les tests.

b) Moyens financiers :

On n’aura pas besoin des moyens financiers puisque on va développer ce projet en utilisant des langages open sources et des logiciels qui sont gratuits (python, OpenCV, Visual Studio Code, etc…).

c) Moyens matériels :

Durant la réalisation de ce projet, nous disposons des matériels suivants : Ordinateurs, Connexion internet

2) Les contraintes

a) Les Contraintes de délai :

Le cahier des charges doit être rendu le 13 février 2022 et le produit final et le rapport doivent être rendu avant le 01 avril 2022.

b) Les contraintes d’ergonomiques :

Les contraintes ergonomiques sont les contraintes liées à l'adaptation entre les fonctionnalités de projet, leurs interfaces et leur utilisation. Pour notre projet, nous devons obéir aux contraintes ergonomiques suivantes :

- Permettre un accès rapide de l'information.
- Interface simple et compréhensible.

c) Contraintes de qualité :

Lors du développement de l'application, l'équipe devra suivre des normes techniques spécifiques pour assurer une meilleure performance et une facilité de maintenance et de mise à jour.

d) Contraintes de sécurité :

Aucune information personnelle n'est requise pour accéder à notre application, L’utilisateur reste anonyme tout au long d’utilisation de l’application.

V. Fonctionnalités ciblées

Les fonctionnalités de l’application sont les suivantes :

Utilisateur :

- Choisir le mode
- Introduire une image
- Introduire une vidéo
- Lancer la détection de texte
- Lancer la détection des images
- Lancer la détection des couleurs
- Lancer la détection des objets
- Lancer la détection des formes

Ci-dessous un diagramme de cas d'utilisation qui présente les fonctionnalités et les acteurs de notre projet :

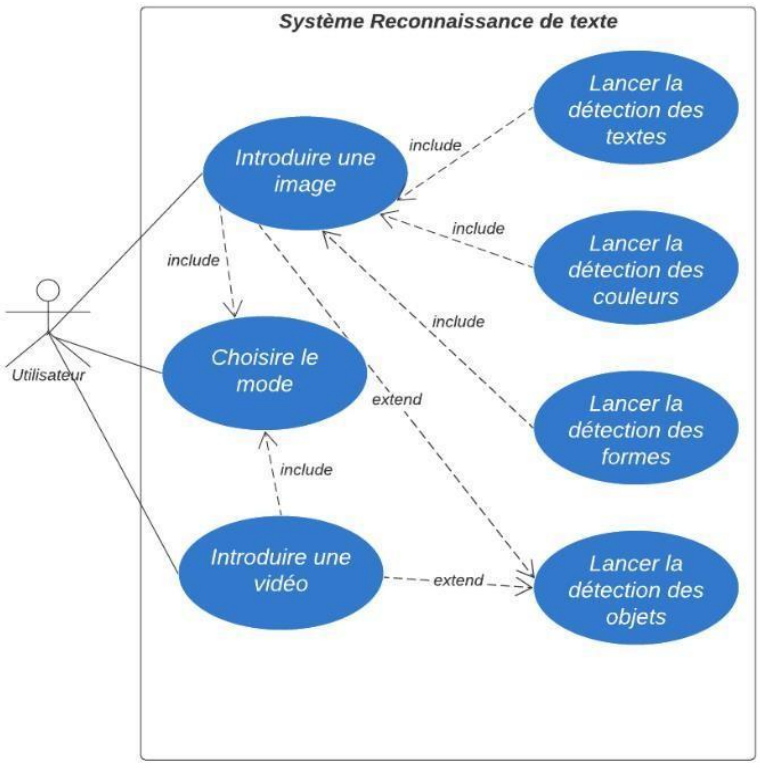


Figure 4 : diagramme de cas d'utilisation

VI. DESCRIPTION DE CHACUNE DES FONCTIONNALITÉS

1) Description des fonctionnalités

Fonctionnalité : Choisir le mode
Utilisateurs : Utilisateur
Description fonctionnalité : L'utilisateur devra choisir le type de mode soit de détecter les couleurs, un texte ou bien les formes d'une image et les objets (image/vidéo).
Règles de gestion : • **Entrée :** Un seul choix à choisir parmi les 5 fonctionnalités proposées par l'application. • **Sortie** : L'interface pour ajouter l'image/vidéo
Priorité : M (must)

Fonctionnalité : Introduire image

Utilisateurs : Utilisateur

Description fonctionnalité : Une fois que l'utilisateur accède à l'application, il trouvera un bouton pour ajouter une image.

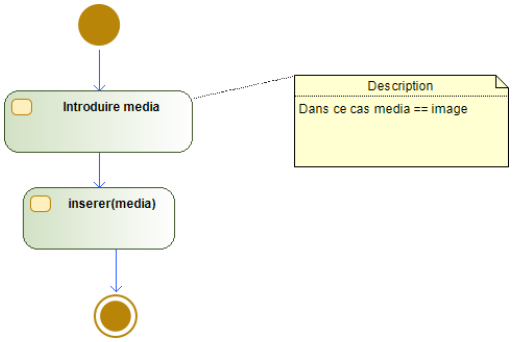


Figure 5 : diagramme d'activité pour la fonctionnalité Introduire image

Règles de gestion :

- **Entrée :** l'image doivent être des extensions JPG, PNG, JPEG
- **Sortie :** Image avec les informations extraites selon le type de reconnaissance choisi par l'utilisateur.

En cas d'erreur :

Un message d'erreur indique que l'utilisateur doit insérer obligatoirement un fichier.

Priorité : M (must)

Fonctionnalité : Introduire une vidéo

Utilisateurs : Utilisateur

Description fonctionnalité : Une fois que l'utilisateur accède à l'application, il trouvera un bouton pour ajouter une vidéo.

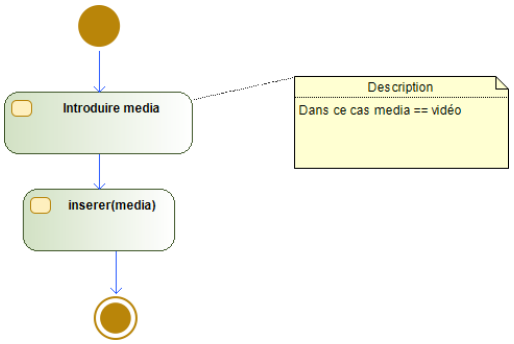


Figure 6 : diagramme d'activité pour la fonctionnalité Introduire vidéo

Règles de gestion :

- **Entrée :** vidéo doivent être des extensions MP4, AVI, FLV
- **Sortie :** Vidéo avec les informations extraites selon le type de reconnaissance choisi par l'utilisateur.

En cas d'erreur :

Un message d'erreur indique que l'utilisateur doit insérer obligatoirement un fichier.

Priorité : M

Fonctionnalité : Lancer la détection du texte

Utilisateurs : Utilisateur

Description fonctionnalité : Une fois que l'utilisateur a ajouté l'image, La détection du texte sera automatiquement lancée.

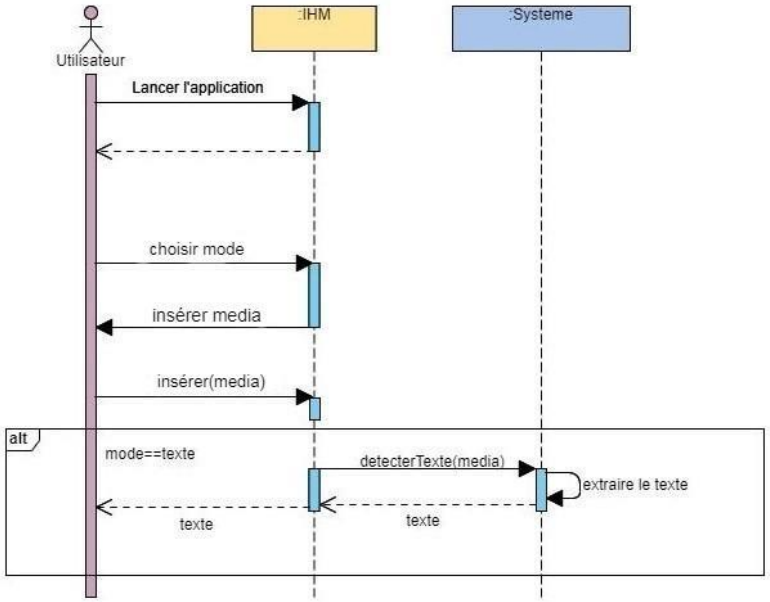


Figure 7 : diagramme de séquence pour la fonction détection du texte (image)

Règles de gestion :

- **Entrée :** Image contient du texte.
- **Sortie :** image affiche le texte origine de l'image et le texte détecté par notre système encadré.

Priorité : M

Fonctionnalité : Lancer la détection des couleurs

Utilisateurs : Utilisateur

Description fonctionnalité : Une fois que l'utilisateur a ajouté l'image, il devra choisi entre 4 propositions le mode voulu dans ce cas l'utilisateur choisi le mode de la détection des couleurs.

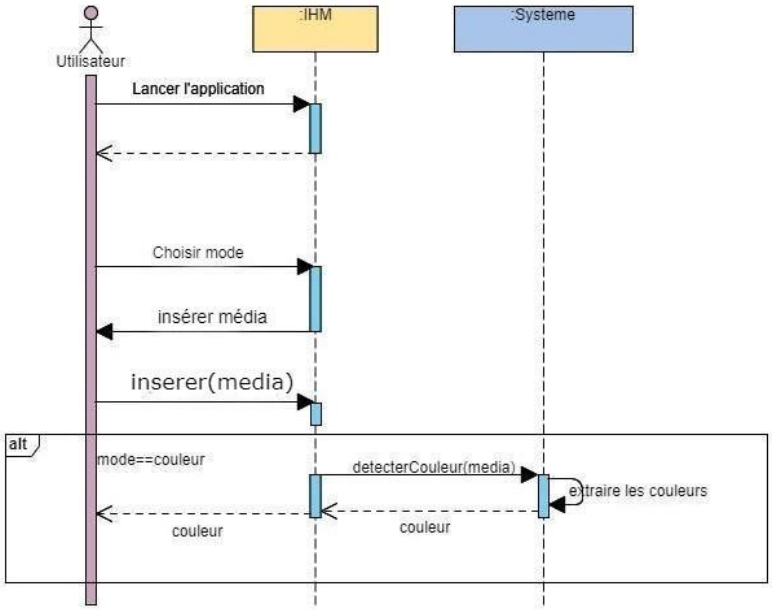


Figure 8 : diagramme de séquence pour la fonction détection de couleurs(image)

Règles de gestion :

- **Entrée :** Image contient des couleurs.
- **Sortie :** Une image sur laquelle l'utilisateur clique sur une zone, le système prend les coordonnées x et y de cette zone et il affiche sa couleur.

Priorité : M

Fonctionnalité : Lancer la détection des formes

Utilisateurs : Utilisateur

Description fonctionnalité : Une fois que l'utilisateur a ajouté l'image, il devra choisi entre 4 propositions le mode voulu dans ce cas l'utilisateur choisi le mode de la détection des formes (Rectangle, Cercle,)

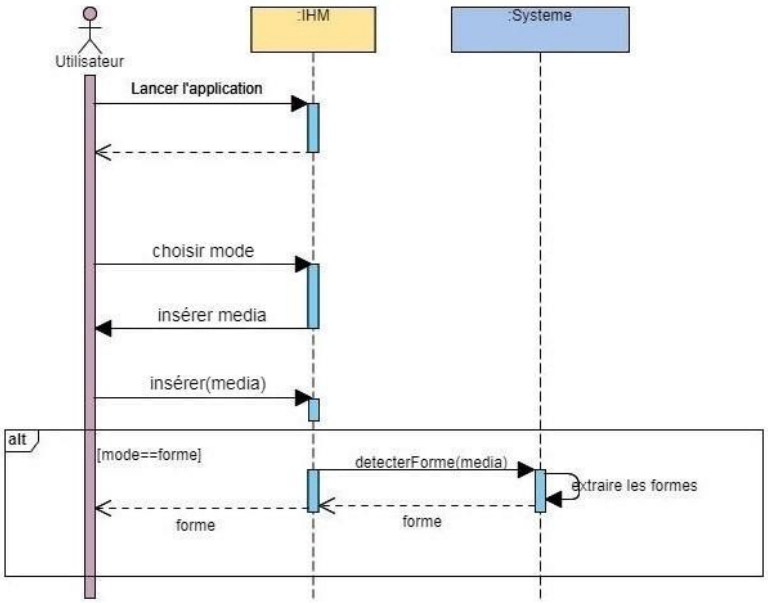
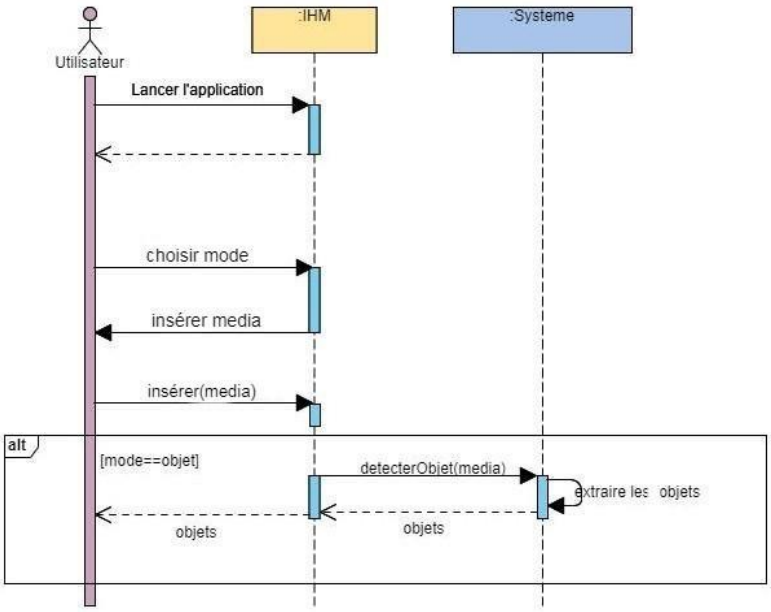


Figure 9: diagramme de séquence pour la fonction détection de formes (image)

Règles de gestion :

- **Entrée :** Image contient des formes
- **Sortie :** Image et chaque forme avec son propre nom.

Priorité : M

Fonctionnalité : Lancer la détection des objets
Utilisateurs : Système
Description fonctionnalité : L'utilisateur a le choix d'ajouter soit une image soit une vidéo si cette opération est bien effectuée, il devra choisi entre 4 propositions le mode voulu dans ce cas l'utilisateur choisi le mode de la détection des objets (Personne, Voiture, ...)    *Figure 9 : diagramme de séquence pour la fonction détection des objets (image/vidéo)*
Règles de gestion : • **Entrée :** Image/Vidéo contient des objets • **Sortie :** Image/Vidéo contient le nom de chaque objet existant dans l'image/vidéo.
Priorité : M

Méthode Moscow : M correspond à « Must »
Ci-dessous un diagramme de séquence qui décrit le fonctionnement global de l'application

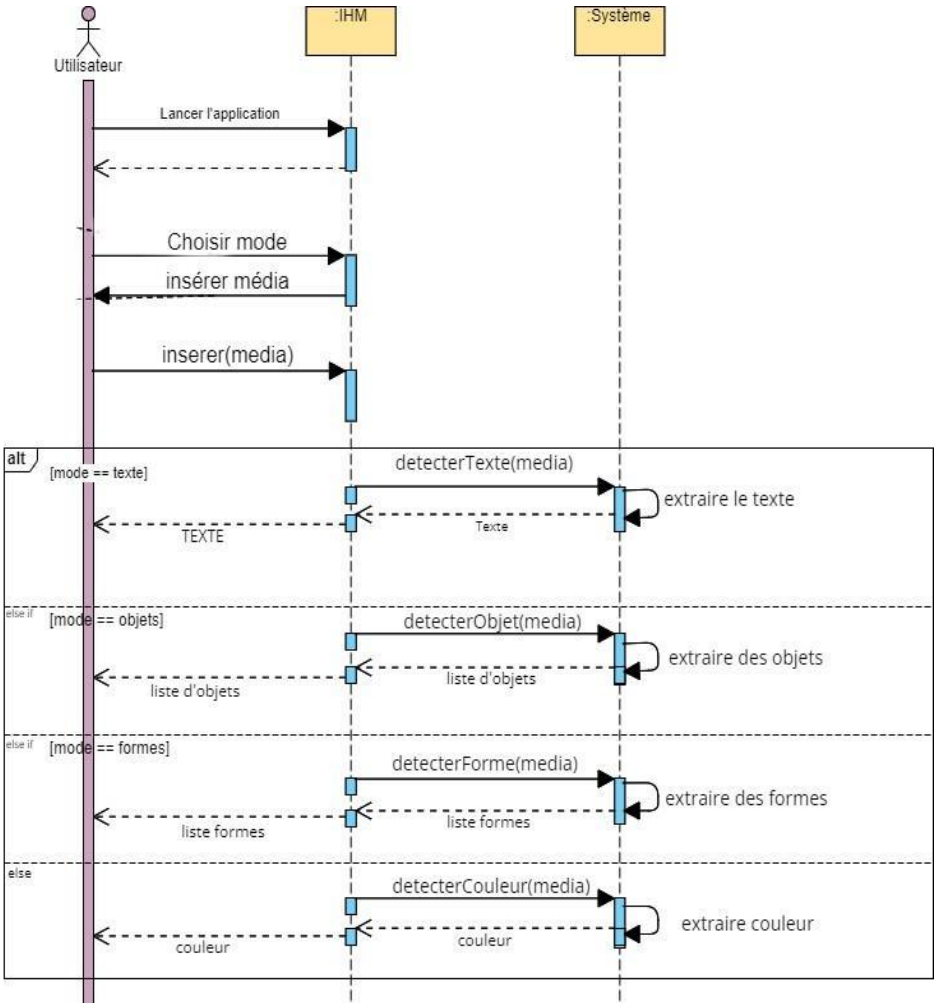


Figure 10 : diagramme de séquence pour un exemple d'usage

2) SCÉNARIO D'USAGE

1. L'utilisateur accède à l'application.
2. Une interface apparaît avec tous les modes de notre projet et l'utilisateur devra en choisir un en cliquant sur le bouton correspondant au mode souhaité.
3. L'utilisateur doit choisir un fichier.
4. Après la vérification du média, Le système lance l'algorithme qui correspond au mode choisi par l'utilisateur.
5. Le système fait la reconnaissance et il retourne les valeurs extraites dans l'image/vidéo à l'utilisateur.
6. L'utilisateur peut répéter cette procédure pour un autre mode ou bien quitter le système.

VII. ÉBAUCHE DE PLANIFICATION - DÉCOMPOSITION EN TÂCHES ET JALONS

1) Liste des tâches

Lettre	Description de la tâche	Durée alloués	Tâches antérieures
A	L'installation des outils de développement	2jour	Aucune
B	Détermination de l'algorithme de détection du texte	6jours	A
C	Détermination de l'algorithme de détection de formes	6jours	A
D	Détermination de l'algorithme de détection de couleurs	6jours	A
E	Détermination de l'algorithme de détection d'objets	6jours	A
F	Test de chacun des algorithmes	8jours	B, C, D, E
G	La création de l'interface	8jours	F
H	L'intégration de nous algorithmes dans l'interface	8jours	G
I	Test du projet	3jours	H

VIII. CONCEPTION

1) Maquette

Dans notre projet on a une seule interface qui représente les 5 fonctionnalités :

- Bouton 1 : détection d'objets à partir d'une image
- Bouton 2 : détection d'objets à partir d'une vidéo
- Bouton 3 : Reconnaissance de texte à partir d'une image
- Bouton 4 : Reconnaissance de formes à partir d'une image
- Bouton 5 : Reconnaissance de couleurs à partir d'une image
- Bouton 6 : Fermeture de la fenêtre

Figure 11 : l'interface graphique du projet

2) Technologies et outils

Nous menons tout d'abord une étude technique où nous décrivons les ressources logicielles utilisées dans le développement de notre projet.

a) Python

Python est le langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels.[6]

b) OpenCV

OpenCV (pour Open Computer Vision) est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel.[7]

c) Pandas

Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Pandas est un logiciel libre sous licence BSD. [10]

d) Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et MacOs [11].

e) Tesseract OCR

Tesseract OCR est un moteur de reconnaissance optique de caractères (acronymie : ROC ou OCR en Anglais) qui a été conçu par les ingénieurs de Hewlett Packard® de 1984 à 1995, avant d'être abandonné [8].

f) Tkinter

Tkinter (de l'anglais Tool kit interface) est la bibliothèque graphique libre d'origine pour le langage Python, permettant la création d'interfaces graphiques. Elle vient d'une adaptation de la bibliothèque graphique Tk écrite pour Tcl.[9]

3) TESTS

L'interface principale de notre projet, qui nous permet de choisir le mode voulu soit de faire la détection d'objets par image ou vidéo, faire la reconnaissance d'un texte, des formes ou bien couleurs.

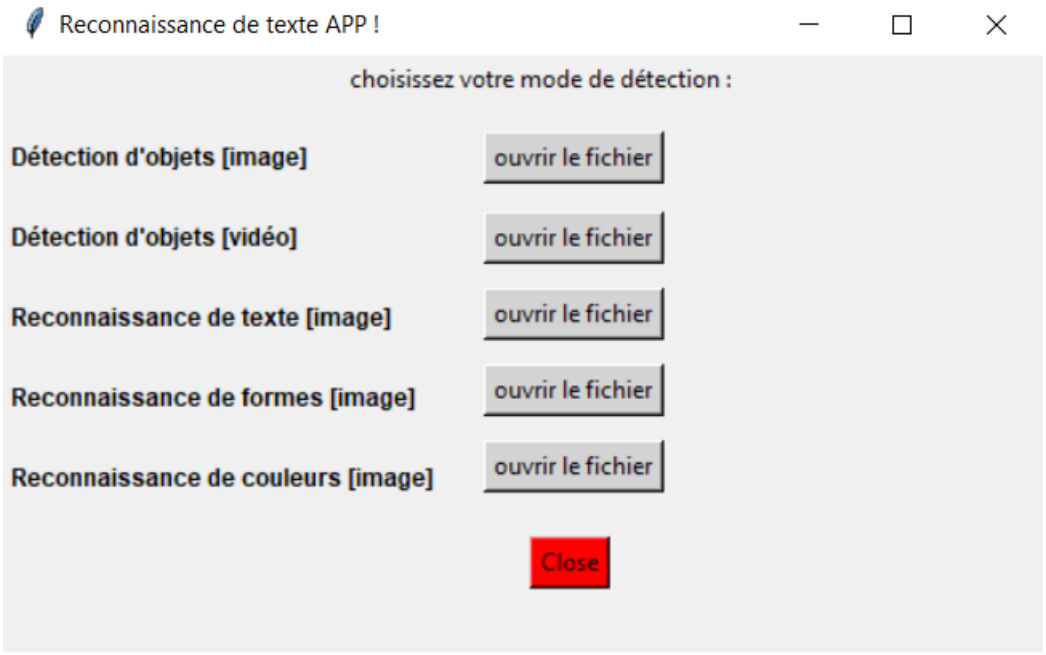


Figure 12 : l'interface graphique du projet

Supposons par exemple que l'utilisateur a choisi le premier choix et donc il va insérer l'image si ce n'est pas le cas le système va afficher un message d'erreur indique l'obligation d'ajouter une image, et si tout va bien le système va faire la détection en affichant l'image entrée par l'utilisateur avec les objets qu'il contient.

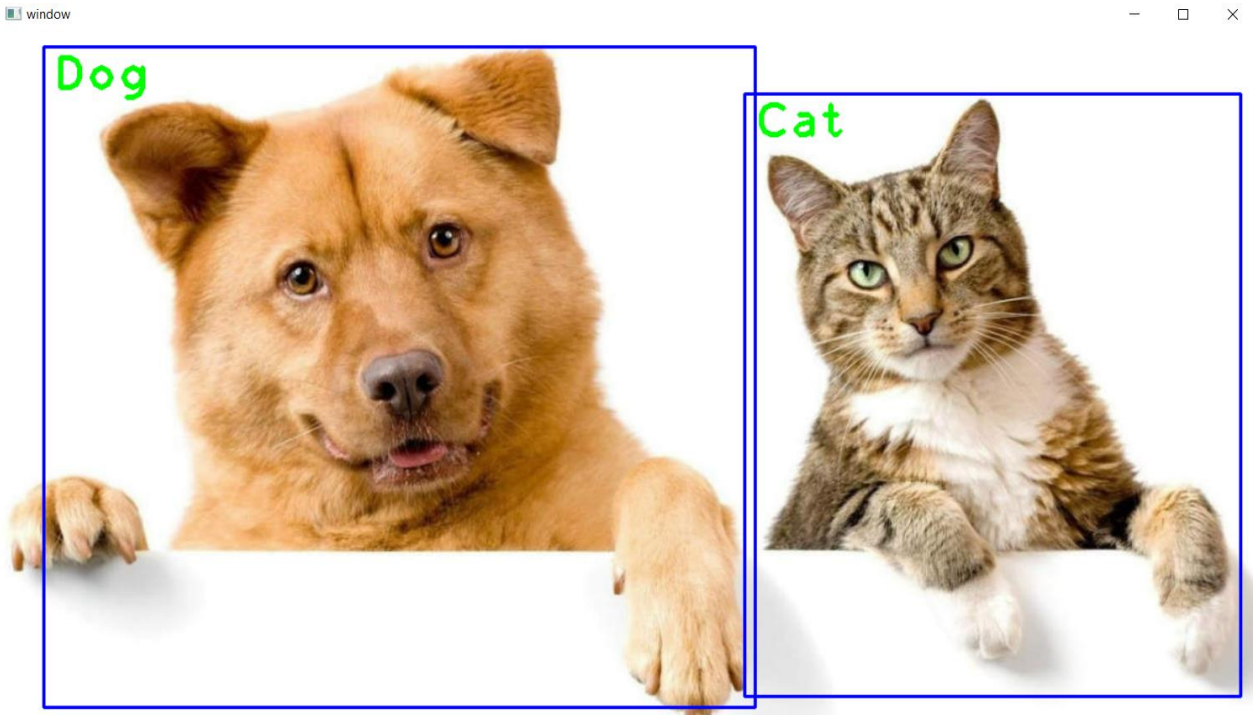


Figure 13 : exemple de la détection d'objets

Le troisième cas consiste à la détection d'un texte dans une image donnée par l'utilisateur, c'est presque le même principe de premier cas juste on a ajouté une autre fonction qui permet de sauvegarder les résultats dans un fichier texte 'result_text'.

Figure 14 : exemple de la détection du texte

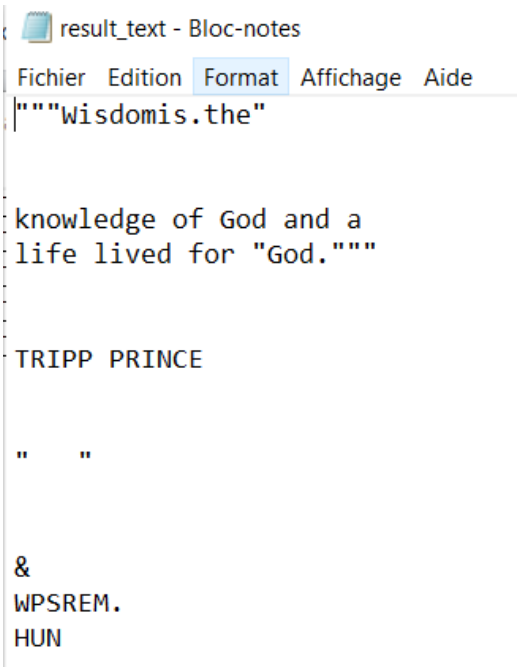


Figure 15 : fichier contient le résultat de la détection du texte

La détection des formes

Résultat après la détection des formes dans une image

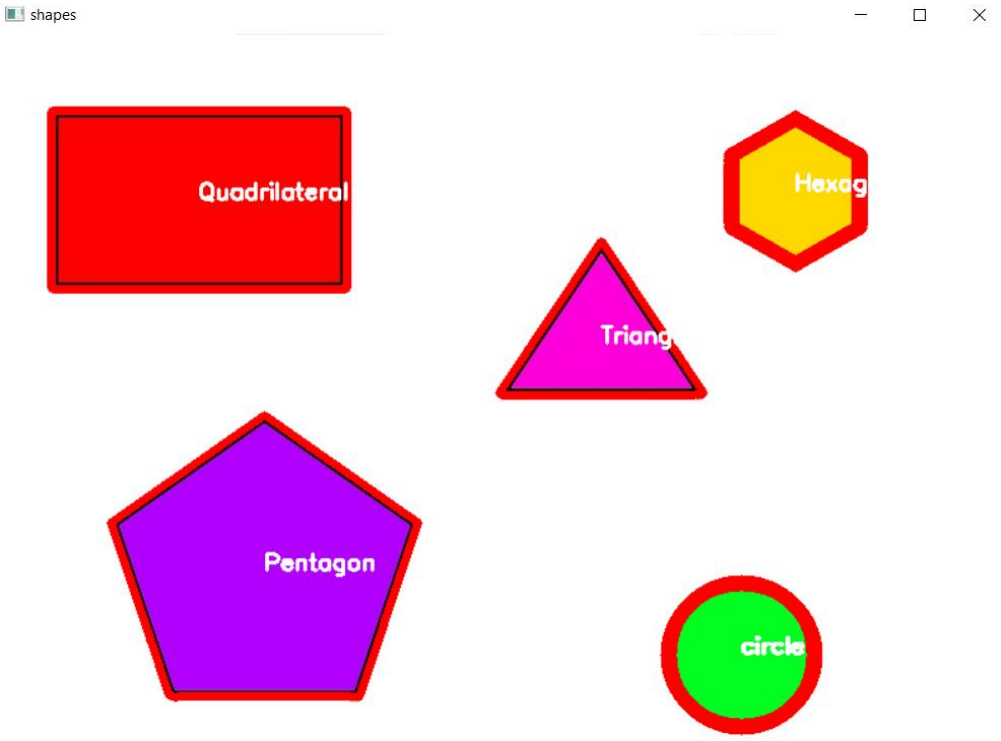


Figure 16 : exemple de la détection de formes

La détection de couleurs

Résultat après la détection de couleurs dans une image

Figure 17 : exemple de la détection de couleurs

4) Guide d'installation et d'utilisation

Avant de lancer le projet, il faut installer les packages suivants :

SUR WINDOWS

Bibliothèque	Instructions ET Command
Opencv2	Https://docs.opencv.org/3.4/d5/de5/tutorial_py_setup_in_windows.html
Tkinter	Pip install tk

	Pre requisite: • Python
Tesseract	Télécharger tesseract .exe : • https://github.com/UB-Mannheim/tesseract/wiki pip install pytesseract
Pandas	Pip install pandas

SUR LINUX

Bibliothèque	Commande
Opencv2	sudo apt install python3-opencv
Tkinter	sudo apt-get install python-tk
Tesseract	sudo apt install tesseract-ocr
Pandas	pip3 install pandas

Après on lance le projet avec la commande **python projet.py** Dans cmd en cas de Windows et en terminal en cas de Linux

IX. Codage

1) Les besoins

Au niveau des technologies, le produit aura besoin de la bibliothèque OpenCV et de Tkinter pour la création de l'interface graphique, et pour les différentes fonctions en aura besoin de :

- Pour la détection d'objets, le fichier **'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'** mobilenet est un réseau Single-Shot multibox Detection (SSD), destiné à effectuer la détection d'objets. En termes simples, ce fichier est un modèle Tensorflow pré-formé et a déjà été formé sur l'ensemble de données Labels, et le fichier **'frozen_inference_graph.pb'** est un fichier de processus pour identifier et enregistrer toutes les choses requises (graphique, poids, etc.) dans un seul fichier que vous pouvez facilement utiliser, et un autre fichier **'Labels.txt'** qui contient le jeu de données. Il existe 91 catégories d'objets d'images étiquetées et segmentées dans le fichier. Ainsi, notre modèle peut détecter et identifier 91 types d'objets.

- Pour la détection de couleurs, le fichier '**color.csv**' comprend 865 noms de couleurs ainsi que leurs valeurs RVB et hexadécimales.

Donc enfin il y aura 5 fonctions dont nous avons parlé et une fonction pour la création de l'interface graphique.

2) Solutions choisies

Principalement on aura 5 fonctions, chaque fonction correspond à un algorithme de détection et une fonction pour la création de l'interface.

La première fonction de détection des objets dans une image.

```
# detection des objects (image)
def detection_objet_image():
    try :
        filepath=filedialog.askopenfilename(filetypes=[
                        ("image", ".jpeg"),
                        ("image", ".png"),
                        ("image", ".jpg"),
                    ])

        config_file = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
        frozen_model = 'frozen_inference_graph.pb'

        model = cv2.dnn_DetectionModel (frozen_model,config_file)

        classLabels=[]
        file_name='Labels.txt'
        with open(file_name,'rt') as fpt:
            classLabels = fpt.read().rstrip('\n').split('\n')

        model.setInputSize(320,320)
        model.setInputScale(1.0/127.5)
        model.setInputMean((127.5,127.5,127.5))
        model.setInputSwapRB(True)

        img=cv2.imread(filepath)



        ClassIndex, confidece, bbox = model.detect(img,confThreshold=0.5)

        font_scale = 3
        font = cv2.FONT_HERSHEY_PLAIN
        for ClassInd , conf, boxes in zip(ClassIndex.flatten(),confidece.flatten(), bbox):
            cv2.rectangle(img,boxes,(255,0,0),2)
            cv2.putText(img,classLabels[ClassInd-1],(boxes[0]+10,boxes[1]+40),font, fontScale=font_scale,color=(0,255,0),thickness=3)

        cv2.imshow("window",img)
        cv2.waitKey(0)

    except Exception:
        pass
        print("vous devez ajouter votre fichier !\n")
```

Figure 18 : fonction pour la détection des objets dans une image

Première chose nous avons défini une fonction **detection_objet_image** qui contient un variable path qui prend le module filedialog en fait c'est un module avec des fonctions de dialogues d'ouverture et de sauvegarde, qui pourra nous aider à améliorer l'interface graphique Tkinter pour

ouvrir ou sauvegarder un fichier, donc comme le code indique l'utilisateur devra entrer une image de type jpeg, png et jpg si la photo inséré ne respecte pas ce critère la détection des objets ne devra pas réalisé.

Après avoir ajouté le fichier de configuration et notre image on utilise la méthode **dnn_DetectionModel**, elle s'agit d'un modèle de réseau neuronal profond qui aide à charger un modèle pré-entraîné (ssd-mobilenet dans notre cas).

Grâce à la fonction cv2 Imshow(), nous affichons l'image entrée avec des rectangles sur chaque objet détecté , après on récupère tous les noms d'objets dans l'image en utilisant les fonctions cv2.rectangle et cv2.putText().

La deuxième fonction de détection des objets dans une vidéo

```
# detection des objects (video)
def detection_objet_video():
    try:
        filepath=filedialog.askopenfilename(filetypes=[
                        ("all video format", ".mp4"),
                        ("all video format", ".flv"),
                        ("all video format", ".avi"),
                    ])

        config_file = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
        frozen_model = 'frozen_inference_graph.pb'

        model = cv2.dnn_DetectionModel (frozen_model,config_file)

        classLabels=[]
        file_name='Labels.txt'
        with open(file_name,'rt') as fpt:
            classLabels = fpt.read().rstrip('\n').split('\n')



        model.setInputSize(320,320)
        model.setInputScale(1.0/127.5)
        model.setInputMean((127.5,127.5,127.5))
        model.setInputSwapRB(True)

        cap=cv2.VideoCapture(filepath)


        #vérifiez si le videoCapture est ouvert correctement
        if not cap.isOpened():
            cap= cv2.videoCapture(0)

        if not cap.isOpened():
            raise IOError("impossible d'ouvrir la vidéo")


        font_scale=3
        font = cv2.FONT_HERSHEY_PLAIN


        while cap.isOpened():
            ret,frame = cap.read()
```

```
        ClassIndex, confidece, bbox = model.detect(frame,confThreshold=0.55)

        if (len(ClassIndex)!=0):
            for ClassInd ,conf, boxes in zip(ClassIndex.flatten(),confidece.flatten(), bbox):
                if(ClassInd<=80):
                    cv2.rectangle(frame,boxes,(255,0,0),2)
                    cv2.putText(frame,classLabels[ClassInd-1],(boxes[0]+10,boxes[1]+40),font, fontScale=font_scale,color=(0,255,0),thickness=2)


        cv2.imshow("détection d'objet",frame)

        if cv2.waitKey(2) & 0xFF == 27 :
            break



    cap.release()

except Exception as e:
    print("la reconnaissance des objets dans la vidéo a été effectuée avec succès\n")
```

Figure 19 : fonction pour la détection des objets dans une vidéo

À partir d'une source vidéo, nous allons utiliser la librairie **Tkinter** pour lire cette vidéo et la découper en différentes frames. C'est à partir de ces frames que l'on va pouvoir détecter des objets sur la vidéo. En effet, les modèles de machine learning pour la détection et la classification d'objet fonctionnent sur des images issues de la vidéo. Chaque frame est alors traité par un modèle de détection développé sur **TensorFlow** qui après entraînement détectera les objets sur chaque frame de la vidéo.

La 3ème fonction de la reconnaissance de texte dans une image

```
# detection de texte (image)
def detection_texte():
    try:
        pytesseract.tesseract_cmd = "C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
        filepath=filedialog.askopenfilename(filetypes=[
                        ("image", ".jpeg"),
                        ("image", ".png"),
                        ("image", ".jpg"),
                    ])



        img = cv2.imread(filepath)

        image_data = pytesseract.image_to_data(img, output_type=Output.DICT)


        for i, word in enumerate(image_data['text']):
            if word !="":
                x,y,w,h = image_data['left'][i],image_data['top'][i],image_data['width'][i],image_data['height'][i]
                cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),1)
                cv2.putText(img, word,(x,y-16),cv2.FONT_HERSHEY_COMPLEX, 0.5,(0,255,0),1)


        cv2.imshow("window",img)
        cv2.waitKey(0)
```

Figure 20 : fonction pour la détection du texte dans une image

Cette fonction permet la reconnaissance de texte dans une image utilisant tesseract OCR.

Après on a ajouté des lignes de code pour ajouter le résultat obtenu dans un fichier nommé ‘result_text’.

```
# ajouter le contenu de l'image dans un fichier .txt
parse_text = []
word_list = []

last_word = ''

for word in image_data['text']:

    if word!='':

        word_list.append(word)
        last_word = word

    if (last_word!='' and word == '') or (word==image_data['text'][-1]):

        parse_text.append(word_list)
        word_list = []

with open('result_text.txt',  'w', newline="") as file:
    csv.writer(file, delimiter=" ").writerows(parse_text)
```

La 4ème fonction de la reconnaissance des formes dans une image

```
def detection_forme():

    try:
        filepath=filedialog.askopenfilename(filetypes=[
                        ("image", ".jpeg"),
                        ("image", ".png"),
                        ("image", ".jpg"),
                    ])

        img = cv2.imread(filepath)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        _, threshold = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
        contours, _ = cv2.findContours(threshold, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        i = 0

        for contour in contours:

            if i == 0:
                i = 1
                continue

            approx = cv2.approxPolyDP(contour, 0.01 * cv2.arcLength(contour, True), True)

            cv2.drawContours(img, [contour], 0, (0, 0, 255), 5)

            M = cv2.moments(contour)
            if M['m00'] != 0.0:
                x = int(M['m10']/M['m00'])
                y = int(M['m01']/M['m00'])

            # mettre le nom de la forme au centre de chaque forme
            if len(approx) == 3:
                cv2.putText(img, 'Triangle', (x, y),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

            elif len(approx) == 4:
                cv2.putText(img, 'Quadrilateral', (x, y),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

            elif len(approx) == 5:
                cv2.putText(img, 'Pentagon', (x, y),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
```

```
        elif len(approx) == 6:
            cv2.putText(img, 'Hexagon', (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

        elif len(approx) == 7:
            cv2.putText(img, 'heptagone', (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

        elif len(approx) == 8:
            cv2.putText(img, 'octogone', (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)


        else:
            cv2.putText(img, 'circle', (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

    cv2.imshow('shapes', img)

    cv2.waitKey(0)
    cv2.destroyAllWindows()

except Exception:
    pass
    print("vous devez ajouter votre fichier !\n")
```

Figure 21 : fonction pour la détection de formes dans une image

Pour trouver des formes présentées dans une image nous avons utilisé la fonction findContours() et approxPolyDP() d'OpenCV. Ces deux fonctions permettent de détecter des formes en fonction du nombre de coins dont il dispose.
Par exemple, un triangle à 3 coins, un carré à 4 coins et un pentagone à 5 coins.

Mais d'abord il faut utiliser une image binaire ou en noir et blanc à l'intérieur de la findContours() fonction donc pour convertir l'image donnée en binaire, c'est le rôle de la fonction cvtColor() et threshold() d'OpenCV. Après on va prendre le résultat retourné par la fonction drawCountours() et si ce résultat = 3 donc il s'agit d'un triangle et s'il est égal à 4 donc il s'agit d'un carré et ainsi de suite.

La 5ème fonction de la reconnaissance de couleurs dans une image

```
try:
    filepath=filedialog.askopenfilename(filetypes=[
                        ("image", ".jpeg"),
                        ("image", ".png"),
                        ("image", ".jpg"),
                    ])


    csv_path = 'colors.csv'

    # lecture du fichier csv
    index = ['color', 'color_name', 'hex', 'R', 'G', 'B']
    df = pd.read_csv(csv_path, names=index, header=None)


    img = cv2.imread(filepath)
    img = cv2.resize(img, (800,600))

    #declaration des variables globales
    clicked = False
    r = g = b = xpos = ypos = 0

    #fonction pour calculer la distance minimale de toutes les couleurs et obtenir la couleur la plus correspondante
    def get_color_name(R,G,B):
        minimum = 1000
        for i in range(len(df)):
            d = abs(R - int(df.loc[i,'R'])) + abs(G - int(df.loc[i,'G'])) + abs(B - int(df.loc[i,'B']))
            if d <= minimum:
                minimum = d
                cname = df.loc[i, 'color_name']

        return cname

    #fonction pour obtenir les coordonnées x,y du double clic de la souris
    def draw_function(event, x, y, flags, params):
        if event == cv2.EVENT_LBUTTONDBLCLK:
            global b, g, r, xpos, ypos, clicked
            clicked = True
            xpos = x
            ypos = y
            b,g,r = img[y,x]
            b = int(b)
            g = int(g)
            r = int(r)
```

```
    cv2.namedWindow('image')
    cv2.setMouseCallback('image', draw_function)

    while True:
        cv2.imshow('image', img)
        if clicked:
            #cv2.rectangle(image, startpoint, endpoint, color, thickness)-1 fills entire rectangle
            cv2.rectangle(img, (20,20), (600,60), (b,g,r), -1)

            #Creating text string to display( Color name and RGB values )
            text = get_color_name(r,g,b) + ' R=' + str(r) + ' G=' + str(g) + ' B=' + str(b)
            #cv2.putText(img,text,start,font(0-7),fontScale,color,thickness,lineType )
            cv2.putText(img, text, (50,50), 2,0.8, (255,255,255),2,cv2.LINE_AA)

            #For very light colours we will display text in black colour
            if r+g+b >=600:
                cv2.putText(img, text, (50,50), 2,0.8, (0,0,0),2,cv2.LINE_AA)

        if cv2.waitKey(20) & 0xFF == 27:
            break

    cv2.destroyAllWindows()


except Exception:
        pass
        print("vous devez ajouter votre fichier !\n")
```

Figure 22 : fonction pour la détection de couleurs dans une image

Cette fonction permet la reconnaissance de couleurs dans une image, donc après que l'utilisateur importe une image on commence par lire le fichier CSV avec pandas, c'est la bibliothèque qui est très utile lorsque nous devons effectuer diverses opérations sur des fichiers de données comme CSV.

La fonction **draw_function** calcule les valeurs RGB du pixel sur lequel nous double-cliquons. Les paramètres de la fonction ont le nom de l'événement, les coordonnées (x,y) de la position de la souris, etc. Dans la fonction, nous vérifions si l'événement est double-cliqué, puis nous calculons et définissons les valeurs r, g, b avec x, y positions de la souris.

La fonction **get_color_name,** après avoir les valeurs r, g et b avec la fonction draw_function. Cette fonction renverra le nom de la couleur à partir des valeurs RGB. Pour obtenir le nom de la couleur, nous calculons une distance (d) qui nous indique à quelle distance nous sommes de la couleur et choisissons celle qui a la distance minimale.

Après l'affichage de l'image, chaque fois qu'un événement de double-clic se produit, il met à jour le nom de la couleur et les valeurs RVB sur la fenêtre.

En utilisant la fonction cv2.imshow(), nous affichons l'image sur la fenêtre. Lorsque l'utilisateur double-clique sur la fenêtre, nous dessinons un rectangle et obtenons le nom de la couleur pour dessiner du texte sur la fenêtre à l'aide des fonctions cv2.rectangle et cv2.putText().

Et la dernière fonction pour la création de l'interface graphique

```
def window1():


    window = tk.Tk()
    window.iconbitmap('icon.ico')
    window.title("Reconnaissance de texte APP !")
    label=Label(window,text="choisissez votre mode de détection :")
    label.pack()
    label.place(x=170,y=1)

    window.geometry('520x300')

    window.minsize(520,300)
    window.maxsize(520,300)

    T1=Label(window,text="Détection d'objets [image] ",font=("Arial Bold",9))
    T1.place(x=1,y=40)
    button = Button(window,text='ouvrir le fichier', command=detection_objet_image,bg='light grey')
    button.pack(side=tk.BOTTOM)
    button.place(x=240,y=38)


    T2=Label(window,text="Détection d'objets [vidéo] ",font=("Arial Bold",9))
    T2.place(x=1,y=80)
    button2= Button(window,text="ouvrir le fichier", command=detection_objet_video,bg='light grey')
    button2.pack(side=tk.BOTTOM)
    button2.place(x=240,y=78)


    T3=Label(window,text="Reconnaisse de texte [image] ",font=("Arial Bold",9))
    T3.place(x=1,y=120)
    button3 = Button(window,text='ouvrir le fichier', command=detection_texte,bg='light grey')
    button3.pack(side=tk.BOTTOM)
    button3.place(x=240,y=116)



    T4=Label(window,text="reconnaissance de forme [image] ",font=("Arial Bold",9))
    T4.place(x=1,y=160)
    button4= Button(window,text="ouvrir le fichier", command=detection_forme,bg='light grey')
    button4.pack(side=tk.BOTTOM)
    button4.place(x=240,y=154)
```

```
T5=Label(window,text="reconnaissance de couleurs [image] ",font=("Arial Bold",9))
T5.place(x=1,y=200)
button5 = Button(window,text='ouvrir le fichier', command=detection_couleur,bg='light grey')
button5.pack(side=tk.BOTTOM)
button5.place(x=240,y=192)



button6= Button(window,text="Close", command=window.destroy, bg='red')
button6.pack(side=tk.BOTTOM)
button6.place(x=263,y=240)

window.mainloop()
```

Figure 23 : fonction pour la création de l'interface graphique

Cette fonction nous permet de créer une interface qui contient du texte et des boutons, chaque bouton est référencé par une fonction utilisant la méthode ***command***.

Conclusion

Tout au long de la préparation de notre projet S6, nous avons essayé de mettre en pratique les connaissances acquises durant nos études universitaires et cela dans le but de réaliser un projet de Reconnaissance de texte avec Python et OpenCV.

Nous sommes satisfaites du rendu final de notre projet, car il répond au besoin de cahier des charges. D'un point de vue technique, ce projet assez complet, nous a beaucoup apporté, notamment grâce à l'apprentissage et la maîtrise OpenCV, qui nous était inconnue au départ. Mais aussi en termes de traitement d'images, où nous n'avions que de faibles connaissances dans ce domaine. Et enfin, en programmation, de l'approfondissement et de l'apprentissage concernant le langage de programmation Python. D'un point de vue gestion de projet, celui-ci nous a permis de travailler en binôme, en apprenant à devoir s'adapter à la façon de travailler de chacun, mais aussi de devoir respecter un certain délai, nous obligeant à travailler en parallèle sur différents domaines du projet.

Nous avons également mis en place une planification de la mise en œuvre d'un bout à l'autre, donner une vision globale du projet et de la façon dont il se déroulera et avoir une visibilité globale avec les contraintes que cela peut se produire, aussi bien que nous détaillons toutes les étapes de l'analyse et pour que notre projet soit couronné de succès.

Références

[1] Image :

https://fr.wikipedia.org/wiki/Image

[2] Vidéo :

https://www.coursinfo.fr/je-programme/apprendre-a-maitriser-video/quest-ce-que- la-video/

[3] Détection d'objets : https://fr.wikipedia.org/wiki/D%C3%A9tection_d%27objet

[4] Reconnaissance de texte, forme, couleurs :

https://fr.wikipedia.org/wiki/Reconnaissance_de_formes

[5] Amazon Rekognition : https://aws.amazon.com/fr/rekognition/

[6] IBM Image Détection :

https://www.ibm.com/fr-fr/cloud/watson-visual-recognition/pricing

[7] Google Lens :

https://www.fnac.com/Mais-c-est-quoi-Google-Lens/cp39093/w-4