République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et Informatique

Département Informatique

Rapport du Projet : Partie 1
Vision Artificielle

Rédigé par

GUELLATI Mehdi Anis

ZAIT Fouad

Enseignant

Lyes Abada

Année universitaire :2022/2023

Introduction :

Dans ce projet nous avons cacher un texte secret dont on a générer une image imgB dans une image imgA en profitant des avantages de l'espace colorimétrique YCbCr qui sépare la luminance et la chrominance.

Objectif du projet :

a) L'émetteur du texte secret :

- Conversion du texte secret en une image imgB BGR sur 8bit.

- Conversion de imgA à l'espace colorimétrique YCbCr sur 16 bits.

- Insertion de l'image imgB dans l'image imgA dans les bits de poids faible de CbCr (Dans notre travail on a insérer dans les 7eme et 8eme bits Cb ensuite une fois qu'on a terminer et qu'il reste encore du des bits de l'imgB on insère dans Cr)

- Conversion de l'image résultante de l'opération précédente et sauvegarde en RGB sur 16bits .

b) Le récepteur du texte secret :

- Le récepteur fait l'opération inverse pour afficher l'image imgB qui contient le texte secret.

Explication de la démarche suivi et du code source :

1-Importation des librairies nécessaires au travail :

import cv2 : Pour le traitement des images en temps réels .

import numpy as np : Pour la manipulation de tableaux multidimensionnels nous en aurons besoin dans plusieurs étapes (conversion de l'image A en BGR , créer l'image B blanche initialement …)

import tkinter as tk : Nous avons utiliser la librairie tkinter pour programmer notre interface graphique .

from PIL import Image, ImageTk : Bibliothèque qui prend en charge l'ouverture, la manipulation et l'enregistrement d'images dans la fenêtre Tkinter .

2-Chargement de l'image et conversion en BGR 16bits :

-Charger l'image image.png en couleur

```
#********l'image imgA doit être converti à l'espace colorimétrique YCbCr sur 16 bits.
#importation de l'image imgA
imgA = cv2.imread("image.png", cv2.IMREAD_COLOR)
```

-Vérification d'erreur de chargement :

```
if (imgA is None):
    print("Erreur de chargment")
    exit(0)
```

-Redimensionnement de l'image :(Nous avons pris les dimensions (largeur,hauteur)=(600, 330) pour l'image A)

```
imgA=cv2.resize(imgA,(600,330))
```

-Conversion de l'image A en BGR :

Chaque pixel de l'image sera en format BGR (Blue Green Red sur 16bits les valeurs iront de 0 à (2^16 - 1) = 65535 selon l'intensité de la couleur de l'image .

```
#convertion de l'image imgA en BGR 16 bits
imgA = np.uint16(imgA)*256
```

-Fonctions utilisées :

-DecToBin8 : cette fonction permet pour un nombre donnée de retourner le nombre en binaire sur 8bits .

```
def DecToBin8 (nombre):
    return format(nombre,'08b')
```

-DecToBin16: cette fonction permet pour un nombre donnée de retourner le nombre en binaire sur 16bits .

```
def DecToBin16 (nombre):
    B = '{0:016b}'.format(nombre)
    return B
```

-AfficheImage : Cette Fonction permet d'afficher l'image A et de fermer la fenêtre des que le bouton soit enfoncée grâce aux fonctions prédéfinis de opencv .

```
def AfficheImage():

    cv2.imshow("imageA", imgA)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

-Emmeteur : On donnera le texte a cacher en entrée a cette fonction .

Definir imgA comme variable globale pour pouvoir lui effectuer des modifications .

Création d'une image blanche Image B qui contiendra le texte par la suite .

on a définit la taille a 150 *250 , les pixels seront sur 8bits pour chaque couleur BGR.

```
global imgA
imgB = np.zeros((150,250,3), np.uint8)
imgB[:,:,:]=255
```

Découpage du texte en ligne (avec 4 espaces dans chaque ligne) , ici on a choisi de sauter la ligne pour l'affichage du texte après chaque 4 mots pour ne pas dépasser la largeur de la fenêtre.

```
    #découpage du texte en ligne (avec 4 espaces dans chaque ligne)
    compteurEspace=0
    for i in range(len(texte)):
        if texte[i]==" ":
            compteurEspace+=1
            if compteurEspace==4:
                texte=texte[:i+1]+"\n"+texte[i+1:]
                compteurEspace=0
    textAffich=texte.split('\n')
```

-Ajout du message dans l'image vide

Ici on a utilise la fonction prédéfini putText de opencv qui permet d'inserer du texte dans une image .

```
#ajout du message dans l'image vide
y0,d=15,20
for i,line in enumerate(textAffich):
    y=y0+i*d
    cv2.putText(img= imgB,text=line,
    org=(0,y),
    fontFace=cv2.FONT_HERSHEY_SIMPLEX,
    fontScale=0.5,
    color=(0,0,0),
    thickness=1,
    lineType=2)
```

-Conversion de l'image B en binaire

Ici nous avons parcouru les pixels de l'image B et nous avons converti chaque valeur en binaire , on aura a la fin de cette boucle l'image B en binaire .

```
#convertion de l'image du message en binaire
hB,wB,cB = imgB.shape
imageDuTextEnBinaire=''
i=0
for y in range(hB):
    for x in range(wB):
        imageDuTextEnBinaire=imageDuTextEnBinaire+str(DecToBin8(imgB[y,x,0]))
```

-Sauvegarde de la taille du texte en binaire sur 32bits

```
tailleMessage=len(imageDuTextEnBinaire)
tailleMessageEnBinaire='{:032b}'.format(tailleMessage)
```

-Conversion de l'image A en YCBCR et division en Y,Cb ,Cr :

Dans cette partie nous avons créer des arrays vides de la taille de l'image A puis on les a remplis par les valeurs des couches Y , Cb ,Cr de l'image A

```
#convertion de l'image imgA en YCbCr 16 bits
imgA = cv2.cvtColor(imgA, cv2.COLOR_BGR2YCrCb)

#division de l'image en Y / Cb et Cr
imgY = np.zeros(imgA.shape, imgA.dtype)
imgCb = np.zeros(imgA.shape, imgA.dtype)
imgCr = np.zeros(imgA.shape, imgA.dtype)
imgY[:,:,0]=imgA[:,:,0]
imgCb[:,:,1]=imgA[:,:,1]
imgCr[:,:,2]=imgA[:,:,2]
```

-Conversion de Cb et Cr en binaire : Nous allons stocker notre message dans Cb Cr donc on doit avoir converti Cb et Cr en binaire , dans cette partie on a parcouru chaque pixel et on a converti en binaire chaque valeur (sur 16bits) ,a la fin de cette boucle on aura Cb et Cr en binaire stocker .

```
#convertion de Cb et Cr en binaire
hA,wA,cA = imgA.shape
CbEnBinaine=""
CrEnBinaine=""
for y in range(hA):
    for x in range(wA):
        CbEnBinaine=CbEnBinaine+str(DecToBin16(imgCb[y,x,1]))
        CrEnBinaine=CrEnBinaine+str(DecToBin16(imgCr[y,x,2]))
```

-Ajout de la taille du message secret a Cb :

Dans cette partie on a ajouter la taille du message (on a parcouru la taille du message et on prend a chaque fois 2 bits et on les insère dans les 7eme et 8eme bits de Cb selon cette ordre) .

```
compteurCb=7
lstCb=list(CbEnBinaine)
for i in range(0, 32,2):
    lstCb[compteurCb]=tailleMessageEnBinaire[i]
    lstCb[compteurCb+1]=tailleMessageEnBinaire[i+1]
    compteurCb=compteurCb+16
```

-Ajout de l'image B dans Cb ensuite Cr :

Une fois qu'on a insérer la taille du message dans Cb on sera a l'indice 263 , on fera l'insertion des bits de l'image B 2 par 2 jusqu'à la fin dans les 7eme et 8eme bits de Cb , ensuite une fois que la taille de Cb sera atteinte on insère dans Cr .

```
#Ajout de imgB dans Cb et Cr
compteurCb=263
compteurCr=7
lstCr=list(CrEnBinaine)
for bit in range(0,tailleMessage,2):
    if compteurCb<len(CbEnBinaine):
        lstCb[compteurCb]=imageDuTextEnBinaire[bit]
        lstCb[compteurCb+1]=imageDuTextEnBinaire[bit+1]
        compteurCb=compteurCb+16
    elif compteurCr<len(CrEnBinaine):
        lstCr[compteurCr]=imageDuTextEnBinaire[bit]
        lstCr[compteurCr+1]=imageDuTextEnBinaire[bit+1]
        compteurCr=compteurCr+16
    else:
        print('DEPASSEMENT  !!!!')
CbEnBinaine=''.join(lstCb)
CrEnBinaine=''.join(lstCr)
```

On récupère a la fin Cb en Binaire et Cr en binaire dans leurs listes modifié après insertion de l'image du texte .

-Reconversion de l'image A en BGR

Dans cette étape on a reconstituer l'image a partir de Cb en binaire et Cr en binaire qu'on a obtenu de l'étape précédente en remplacent les Cb et Cr de l'image A par les nouvelles après insertion du texte .

```
compteur=0
for y in range(hA):
    for x in range(wA):
        imgCb[y,x,1]=int(CbEnBinaine[compteur:compteur+16],2)
        imgCr[y,x,2]=int(CrEnBinaine[compteur:compteur+16],2)
        compteur=compteur+16
#convertion de l'image
imgA[:,:,1]=imgCb[:,:,1]
imgA[:,:,2]=imgCr[:,:,2]
imgA = cv2.cvtColor(imgA, cv2.COLOR_YCrCb2BGR)
```

-Recepteur :

-Convertir l'image A résultante de l'émetteur en YCbCr :

```
#convertir imgA en YCbCr
imgA = cv2.cvtColor(imgA, cv2.COLOR_BGR2YCrCb)
```

-Division de l'image A en Y ,Cb ,Cr :

```
#division de l'image en Y / Cb et Cr
imgY = np.zeros(imgA.shape, imgA.dtype)
imgCb = np.zeros(imgA.shape, imgA.dtype)
imgCr = np.zeros(imgA.shape, imgA.dtype)
imgY[:,:,0]=imgA[:,:,0]
imgCb[:,:,1]=imgA[:,:,1]
imgCr[:,:,2]=imgA[:,:,2]
```

-Conversion de Cb et Cr en binaire : Une fois qu'on a récupérer Cb et Cr de l'étape précédente , Dans cette étape on a converti Cb et Cr en binaire ,on parcours les pixels ,on récupère les valeurs des couches 1 et 2 qui sont Cb et Cr qu'on convertit en binaire grâce a la fonction DecToBin16 définit précédemment la sortie des deux boucles on aura Cb et Cr de l'image A ou l'image B est caché .

```
#convertion de Cb et Cr en binaire
hA,wA,cA = imgA.shape
CbEnBinaine=""
CrEnBinaine=""
for y in range(hA):
    for x in range(wA):
        CbEnBinaine=CbEnBinaine+str(DecToBin16(imgCb[y,x,1]))
        CrEnBinaine=CrEnBinaine+str(DecToBin16(imgCr[y,x,2]))
```

-Récupération de la taille du message secret a partir de Cb :

Dans la fonction émetteur on a sauvegarder la taille du message en binaire dans les 7 -ème et 8-ème bits de Cb en Binaire, ici on va le récupérer en parcourant les 7eme et 8 -ème bits de Cb en binaire jusqu'à atteindre la fin de la taille du message qui est 32bits (jump de 7 jusqu'à 255 (fin de la taille du message)).Une fois sorti de la boucle on aura la taille du message en binaire .

```
#Récupération de la taille du message secret à partir de Cb
tailleMessageEnBinaire=[]
for compteurCb in range(7, 255, 16):
    tailleMessageEnBinaire.append(CbEnBinaine[compteurCb])
    tailleMessageEnBinaire.append(CbEnBinaine[compteurCb+1])
tailleMessageEnBinaire=''.join(tailleMessageEnBinaire)
tailleMessage=int(tailleMessageEnBinaire,2)
```

-Récupération du Message en binaire :

Dans cette partie On va récupérer le message en binaire d'abord on vérifie si on a insérer le message dans Cb ensuite Cr on récupère les 7 et 8eme bits de chaque 16bits de Cb (qui représente le message) jusqu'à la fin ensuite une fois terminée on récupère les 7eme 8eme bits de chaque 16bits de Cr qu'on stocke dans une liste qu'on a appelé messageEnBinaire .Sinon si la taille du message a été suffisante pour la cacher dans Cb uniquement on récupère uniquement le message a partir de Cb.

```
#Recuperation du message
messageEnBinaire=[]
if (tailleMessage)>((len(CbEnBinaine))-256):
    for compteurCb in range(263, len(CbEnBinaine), 16):
        messageEnBinaire.append(CbEnBinaine[compteurCb])
        messageEnBinaire.append(CbEnBinaine[compteurCb+1])
    for compteurCr in range(7, tailleMessage*8+256, 16):
        messageEnBinaire.append(CrEnBinaine[compteurCr])
        messageEnBinaire.append(CrEnBinaine[compteurCr+1])
else:
    for compteurCb in range(263, tailleMessage*8+256, 16):
        messageEnBinaire.append(CbEnBinaine[compteurCb])
        messageEnBinaire.append(CbEnBinaine[compteurCb+1])
messageEnBinaire=''.join(messageEnBinaire)
```

-Conversion du message en decimal :

Ici on parcours chaque 8 bits du message en Binaire qu'on convertit en décimal pour avoir notre message en décimal .

```
#convertion du message en decimal
message=[]
for i in range (0, len(messageEnBinaire), 8):
    message.append(int(messageEnBinaire[i:i+8], 2))
```

-Création de l’image qui contient le message secret :

On crée une image vide qu’on va ensuite remplir par le message obtenu de l’étape précédente . imgB sera ainsi l’image B reconstituée .

```
#creation de l'image qui contient le text secret
imgB = np.zeros((150,250,3), np.uint8)
i=0
for y in range(150):
    for x in range(250):
        imgB[y,x,0]=message[i]
        imgB[y,x,1]=message[i]
        imgB[y,x,2]=message[i]
        i=i+1
```

-Reconversion de l’image A en BGR :

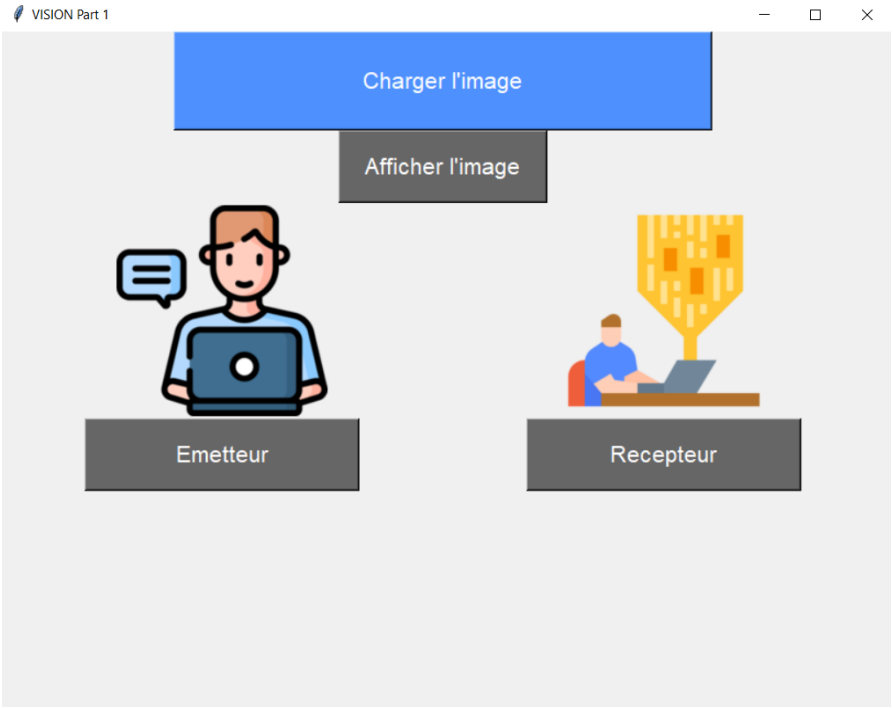
On doit d’abord convertir Cb et Cr en décimal ensuite on met dans la couche 1 de l’image A qui est Cb le résultat de Cb en décimal et on met dans la couche 2 de l’image A qui est CR le résultat de Cr en décimal . ensuite on fait une conversion de l’image de Y Cb Cr vers BGR pour avoir l’image A en BGR .

```
#conversion de Cb et Cr et decimal
compteur=0
for y in range(hA):
    for x in range(wA):
        imgCb[y,x,1]=int(CbEnBinaine[compteur:compteur+16],2)
        imgCr[y,x,2]=int(CrEnBinaine[compteur:compteur+16],2)
        compteur=compteur+16
#convertion de l'image
imgA[:,:,1]=imgCb[:,:,1]
imgA[:,:,2]=imgCr[:,:,2]
imgA = cv2.cvtColor(imgA, cv2.COLOR_YCrCb2BGR)
```

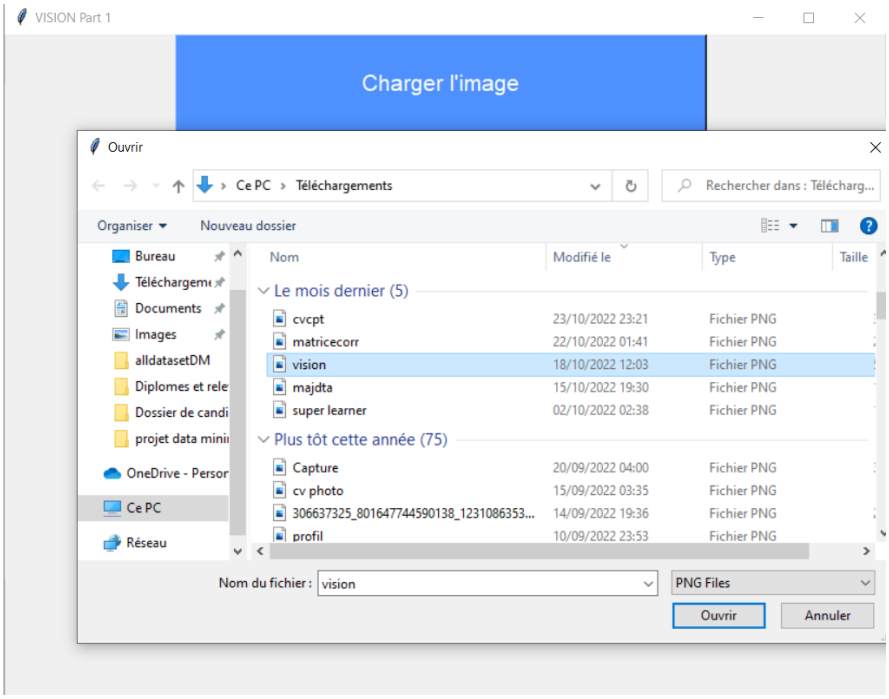
-Affichage de l’image B :

```
#affiche de imgB
cv2.imshow("imageB", imgB)
```

Captures d’écrans de l’interface :

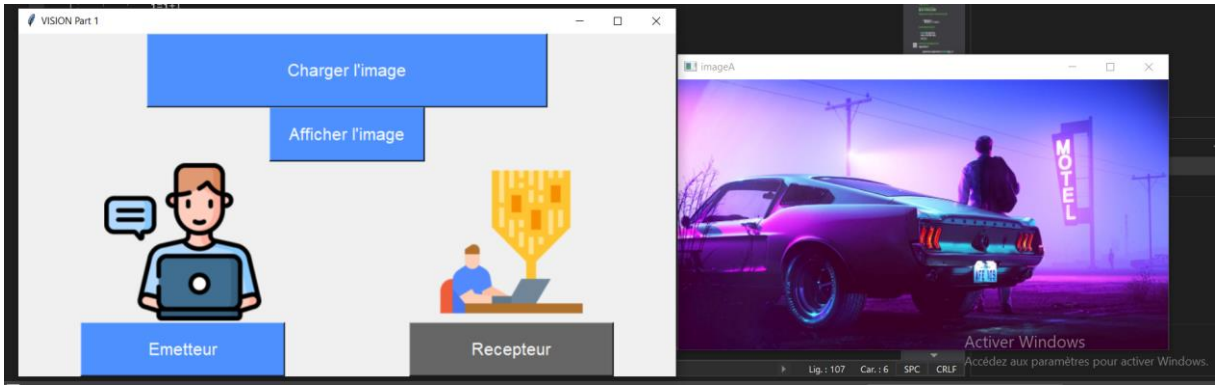


En appuyant sur charger l’image on pourra charger l’image png qu’on souhaite a partir de notre répertoire comme montre ci-dessous :



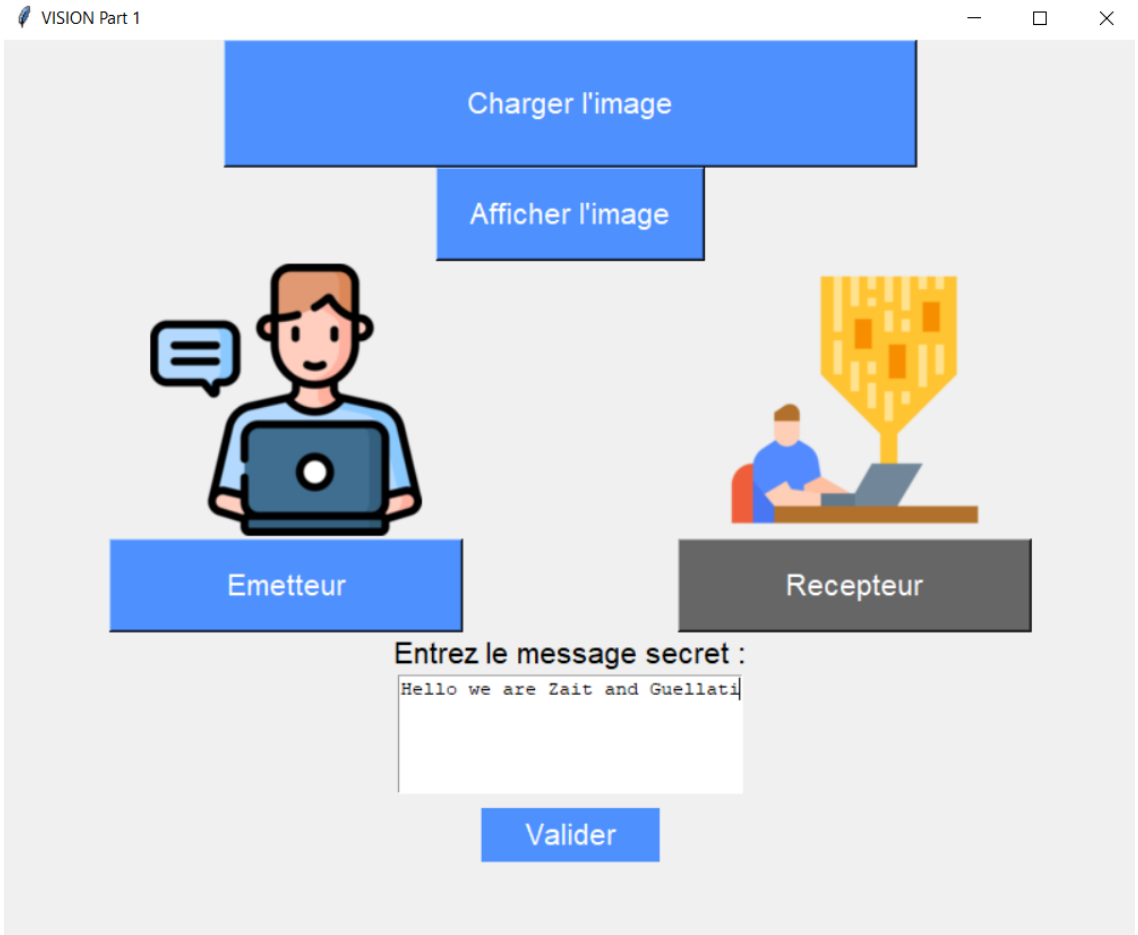
Une fois qu'on aura sélectionné une image on pourra l'afficher et émettre un message qu'on veut cache dans cette image :

-afficher :

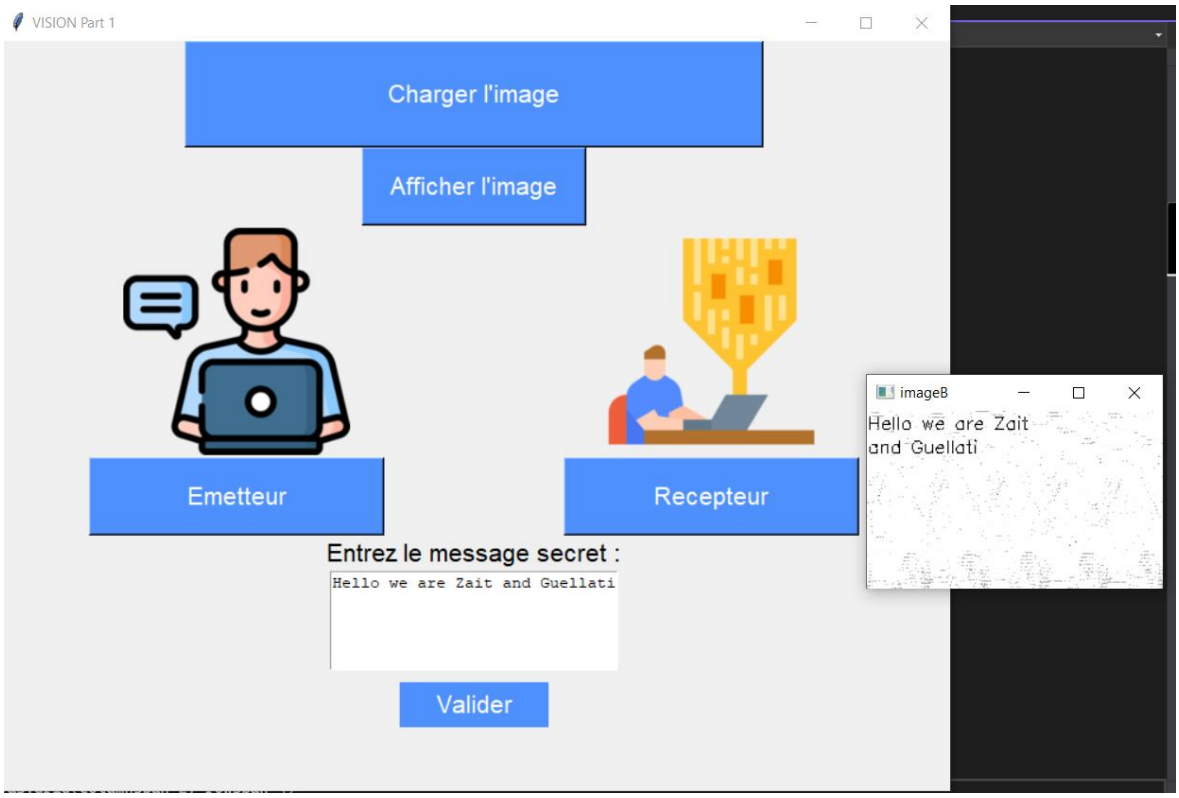


-Emetteur :

En cliquant sur Emetteur une zone de texte sera affiche pour écrire le message qu'on souhaite cache et en appuyant sur valider la fonction Emmeteur sera appeler pour cacher le message .



Apres l'envoi du message en cliquant sur Récepteur on pourra reconstitué le message qu'on a cacher .



Conclusion :

Dans ce rapport on a détaillé tout ce qu'on a fait durant cette première partie du projet dont le but était de cacher un message dans une image en utilisant toutes les techniques vu en cours et en TP .

Exécution :

Comme demandé , pour exécuter il faut simplement ouvrir le dossier contenant le code source et les images sur Visual Studio que vous trouverez en pièce jointe du mail avec ce rapport .

Installation au préalable : si les librairies utilisée n'ont pas été installé au préalable il est nécessaire d'exécuter dans la ligne de commande les installations suivantes :

pip install opencv-python
pip install tkinter
pip install numpy
pip install Pillow