# Projet Python P2 – Réflexion

## Question n°1

Au début, j'ai eu l'idée instantanée de reprendre le code que l'on avait fait dans le dernier cours afin de récupérer l'étoile à l'aide du click de la souris, mais le premier problème arrive.

Lorsque je seuil mon image, l'étoile vient se coller au cadeau :

Il est donc évident que lorsque je clique sur l'étoile, le contour prend le cadeau et tout ce qui est collé sur l'image seuillée :

L'idée qui me vient alors à l'esprit est de faire une érosion de l'image, une première fois avec une seule itération :

C’est un petit peu mieux mais toujours pas super propre, je passe donc **iterations** à 2 :

Cool, mon étoile est correctement segmentée. Je me rends compte sur le coup que les bords ne sont pas parfaitement. Je tente d’autres chose, je me rends compte que plus j’augmente la taille de mon kernel plus les contours sont éloignés de l’étoile de base.

Exemple pour **kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(10,10))** :

Je décide donc de rester sur mon plan de base car parmi tous les essais, c’était le plus précis. On a donc une érosion avec **2 itérations** et comme élément structurant :

**kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))**


## Question N°2:

Pour ne pas mentir en lisant la question je me suis dit qu'il allait vraiment falloir réfléchir.

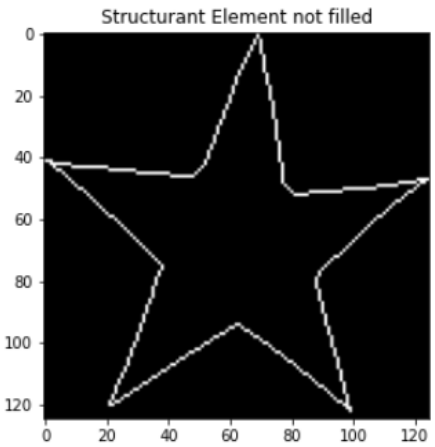
En relisant le code, j'ai une idée qui me vient tout de suite à l'esprit, tout se joue forcément à l'aide de la détermination du contour. Puisque **cv2.findContours** renvoie forcément les coordonnées de la région d'intérêt.

Je vais me renseigner sur la documentation d'openCV et bingo.

*Contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.*

Je continue donc ma stratégie, en parcourant chaque élément de contours, j'ai donc un point de l'étoile sous forme de tableau numpy, et je peux donc crée une nouvelle image avec uniquement une étoile dessinée. A force de recherche des coordonnées je trouve comment ajuster parfaitement l'image. Voici le code et le résultat :

```
hauteur_etoile = 125

largeur_etoile = 125

shift_hauteur_etoile = 44

shift_largeur_etoile = 175

milieu_etoile = (62,62)


empty_bis = np.zeros((hauteur_etoile,largeur_etoile),dtype="uint8")

for contour in c :

    empty_bis[contour[0][1]-shift_hauteur_etoile,contour[0][0]-
shift_largeur_etoile] = 255

plt.imshow(empty_bis)
```

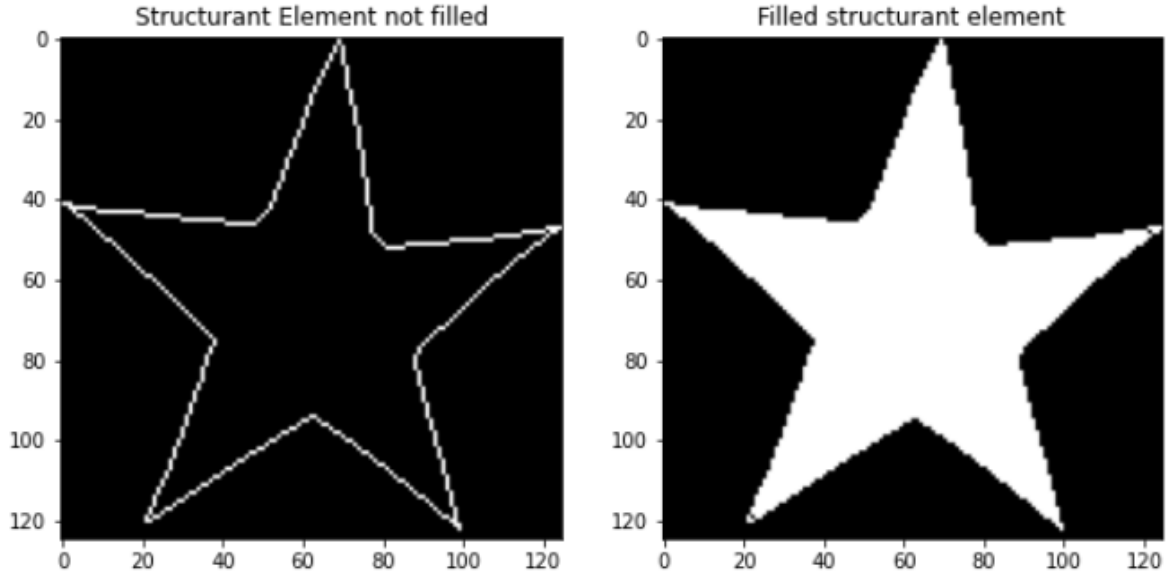


J’ai donc une image, de 125x125, contenant les contours de l’étoile avec des pixels à 255. Je me dis maintenant que si je remplis l’étoile et que je passe l’image en binaire. J’aurais un tableau numpy contenant des 0 pour les bords et des 1 pour l’étoile. Soit l’équivalent d’un élément structurant avec la forme de l’étoile.

Voici donc mon code ainsi que le résultat :

**cv2.floodFill(empty_bis,None,milieu_etoile,255)**

**(thresh, emEtoile) = cv2.threshold(empty_bis, 127, 1, cv2.THRESH_BINARY)**

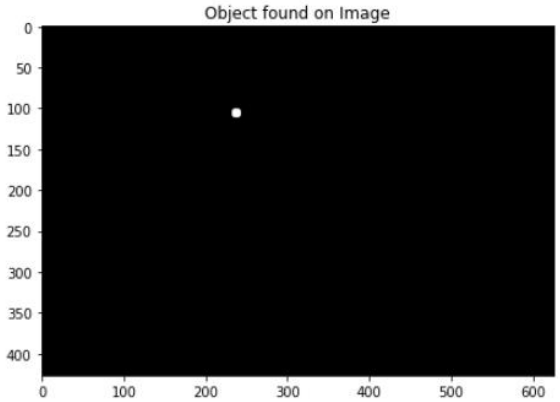



Nous possédons désormais une étoile blanche sur fond noir, soit un **tableau numpy** composé de 0 et de 1 formant une étoile.

## Question N°3:

Dans la logique des choses, je possède un élément structurant qui a la forme de mon objet, si je fais donc un **Hitmiss** avec mon élément structurant je devrai avoir un point blanc au centre de l'étoile, ce qui localisera donc mon objet.

```
img_bis = cv2.imread(path)

gray = cv2.cvtColor(img_bis,cv2.COLOR_BGR2GRAY)

(thresh,bw_img_bis) = cv2.threshold(gray,0,255,cv2.THRESH_BINARY | cv2.THRESH_OTSU)


kernel_bis = np.ones((3,3), np.uint8)


bw_img_bis = cv2.bitwise_not(bw_img_bis)


hitmiss1 = cv2.morphologyEx(bw_img_bis, cv2.MORPH_HITMISS, emEtoile)

dilation = cv2.dilate(hitmiss1, kernel_bis, iterations = 3)
```


Bingo, un point apparaît bien au centre de l'étoile. J'ai donc localisé mon objet.

Question N°4:

## Améliorations postérieures :

En revenant à postériori sur la deuxième entrée et notamment sur la délimitation de mon élément structurant. Je rentrais des valeurs en dur et j'étais presque sûr que je pouvais apporter une modification pour rendre cette délimitation dynamique.

Je test alors ce code :

```
max_hauteur = 0

max_largeur = 0

min_hauteur = hauteur

min_largeur = largeur


for contour in c :

    if(contour[0][0] < min_largeur):

        min_largeur = contour[0][0]

    if(contour[0][0] > max_largeur):

        max_largeur = contour[0][0]

    if(contour[0][1] < min_hauteur):

        min_hauteur = contour[0][1]

    if(contour[0][1] > max_hauteur):

        max_hauteur = contour[0][1]


shift_largeur = min_largeur

shift_hauteur = min_hauteur

hauteur_cut = max_hauteur - min_hauteur + 2

largeur_cut = max_largeur - min_largeur + 2

print(shift_largeur,shift_hauteur,hauteur_cut,largeur_cut)
```

```
175 44 124 126
```

Voici les résultats

On peut voir qu’ils sont très proche voir similaire à ceux que j’ai rentré manuellement :

**hauteur_etoile = 125**

**largeur_etoile = 125**

**shift_hauteur_etoile = 44**

**shift_largeur_etoile = 175**

Je remplace donc par le bout de code dynamique pour la question 2.

Bingo ça marche ! 😊

Ce qui veut dire que théoriquement, je peux faire un élément structurant avec n’importe quel objet de l’image.