**Détection de contour détaillée**

Introduction :

La détection de contour est une technique essentielle en vision par ordinateur permettant d'identifier les limites des objets dans une image. Dans ce rapport, nous présentons les résultats obtenus en utilisant la bibliothèque OpenCV (Open Source Computer Vision) pour détecter les contours, calculer l'aire du contour et déterminer l'aire maximale.

1. Détection de contour avec OpenCV :

OpenCV (Open Source Computer Vision Library) fournit une fonction pratique findContours() pour la détection de contours. La fonction findContours() prend une image binaire en entrée. On suppose que le premier plan est blanc et l'arrière-plan est noir.

* Elle utilise un algorithme de détection de contours basé sur la différence d'intensité des pixels adjacents. Elle parcourt l'image et identifie les zones où il y a une transition d'intensité significative, ce qui indique la présence d'un contour.
* Syntaxe de la fonction :

contours, hierarchy = cv2.findContours(image, mode, method, contours, hierarchy, offset)

* Résultats :

contours - Ce sont les contours détectés. Chaque contour est stocké en tant que vecteur de points.

hierarchy - C'est un vecteur de sortie facultatif qui contient des informations sur la topologie de l'image.

Cependant, pour visualiser ces résultats, nous pouvons utiliser la fonction cv2.drawContours(), qui dessine simplement les contours sur une image.

* Syntaxe de la fonction :

cv2.drawContours(image, contours, contourIdx, color, thickness, lineType, hierarchy, maxLevel, offset)

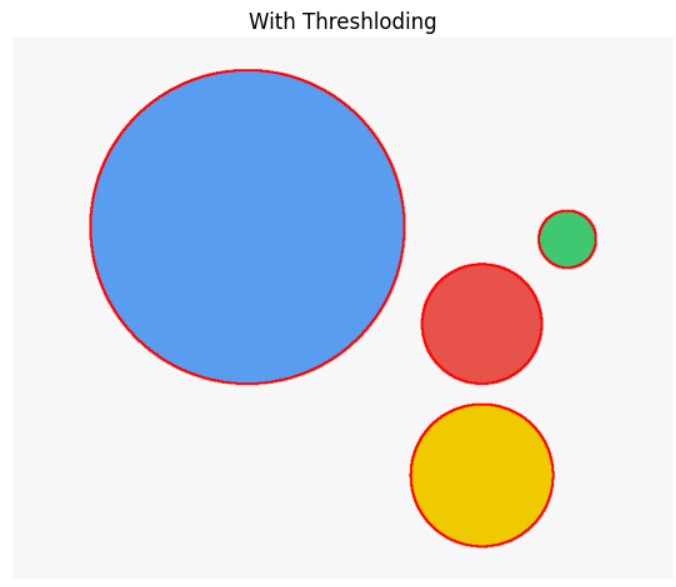
Comme vous l'avez vu précédemment, les fonctions cv2.findContours() prennent en entrée une image binaire à canal unique. Cependant, dans la plupart des cas, l'image d'origine ne sera pas une image binaire. La détection de contours dans des images en couleur nécessite un prétraitement pour produire une image binaire à canal unique pouvant ensuite être utilisée pour la détection de contours.

Les deux techniques les plus couramment utilisées pour ce prétraitement sont les suivantes :

Prétraitement basé sur le seuillage (Thresholding Based Pre-processing)

* Convertir l’image en niveaux de gris cv2.cvtColor()
* Inverser les valeurs des pixels dans une image binaire cv2.bitwise\_not()
* Créer une image seuillée binaire cv2.threshold()

Prétraitement basé sur les contours (Edge Based Pre-processing)

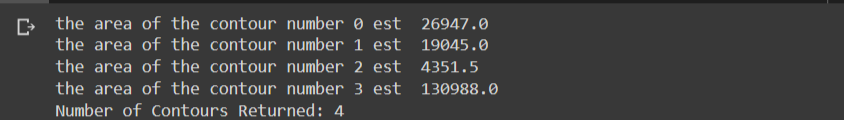


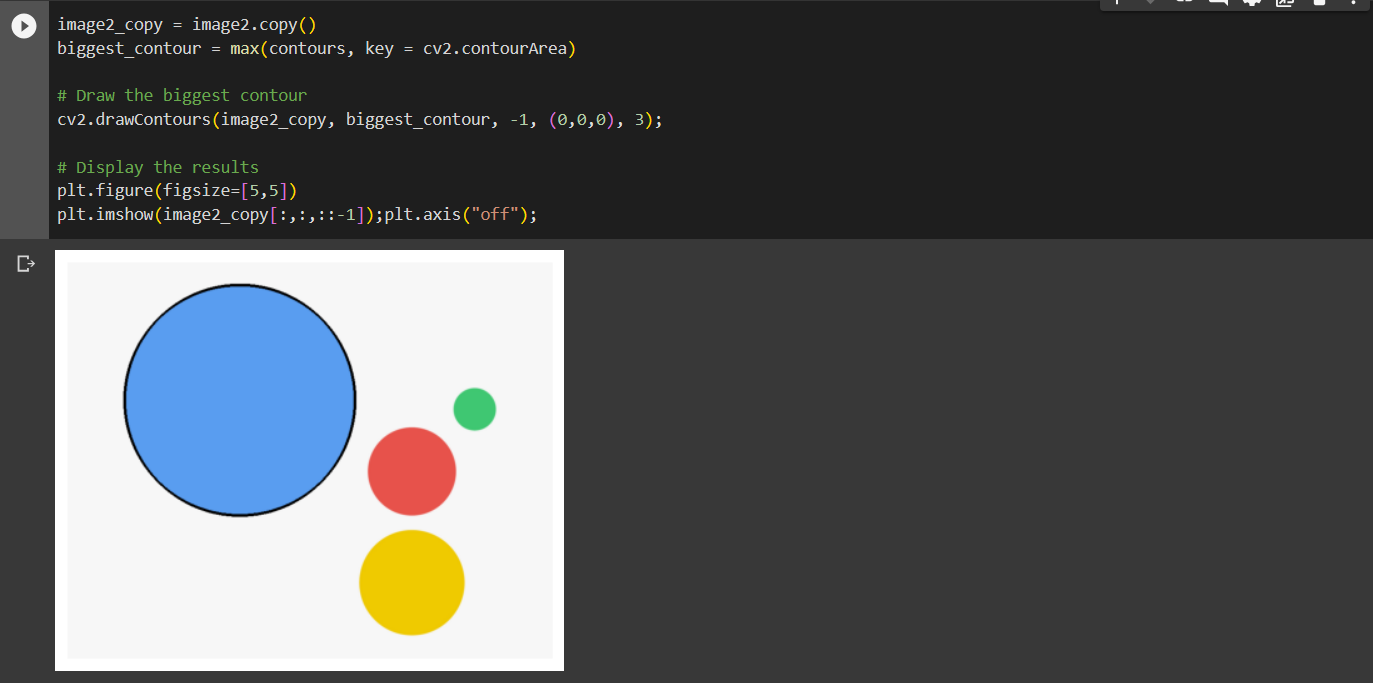
1. Calcul de l'aire du contour et détermination de l'aire maximale :

Une fois les contours détectés, nous avons utilisé la fonction cv2.contourArea() de la bibliothèque OpenCV pour calculer l'aire de chaque contour.

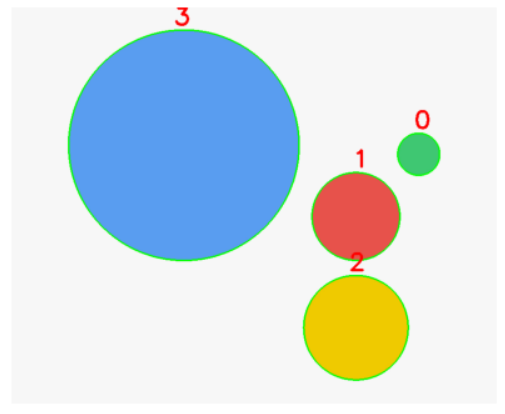
* La fonction utilise l'algorithme du polygone pour calculer l'aire du contour. Elle approxime le contour en un polygone convexe ou concave et mesure ensuite l'aire de ce polygone.

Ensuite nous avons utilisé la fonction max(contours, key = cv2.contourArea)





Pour comparer tous les contours dans la liste, nous pouvons utiliser la fonction sorted(contours, key=cv2.contourArea, reverse= False). Nous spécifions également l'ordre de tri reverse=False, c'est-à-dire dans l'ordre croissant de la taille de l'aire.



Conclusion :

Ce rapport a présenté les résultats de la détection de contour en utilisant la bibliothèque OpenCV. Grâce à cette approche, nous avons pu détecter avec précision les contours des objets dans l'image, calculer leur aire et identifier l'aire maximale.