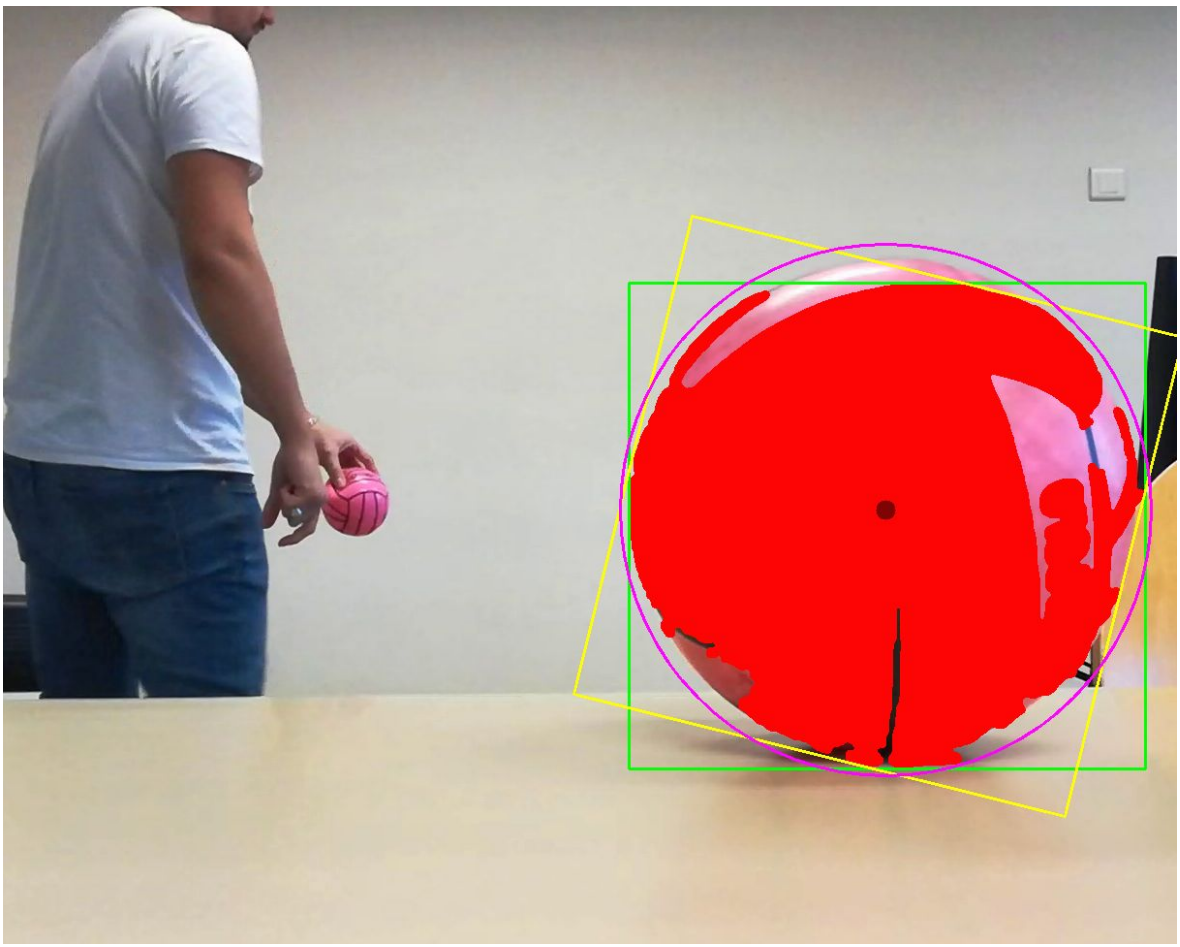
Antonin Bousquier
Ibrahim Elouard

CPE Lyon - 5 ETI - Robotique

15/11/2018

TP VISION : Couleur



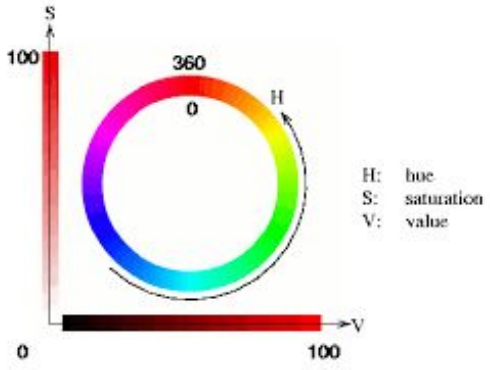
Question 1:

L'objectif premier de ce code est de trouver, en temps réel, le centre d'un objet vert présent dans une vidéo fournie en argument.

Question 2:

Dans cette partie, l'utilisateur va définir la couleur traquée. On lui demande donc la couleur qu'il veut traquer et on adapte les codes HSV suivant sa demande. On va modifier la fonction que l'on récupère de la question 1, la fonction cv2.COLOR_BGR2HSV va convertir le code RGB de la vidéo en code HSV pour que chaque pixel soit plus facile à analyser.

Nos valeurs HSV doivent avoir les caractéristiques suivantes :
H = Choix de la couleur (Compris entre 0 et 179, il suffit de diviser par 2 les valeurs du cercle)
S = Choix de la concentration des pigments : l'intensité (entre 0 et 255)
V = Choix de la luminosité : la brillance (entre 0 et 255)



On obtient bien un tracking très correct des 4 couleurs attendues :

Tracking du bleue

Tracking du rose

Tracking du jaune

Tracking du vert

Vous trouverez le code de cette partie dans notre rendu : “tp1_q2”

Question 4:

Le drone va d’abord récupérer les coordonnées du centre de la boule verte avec un code similaire à celui du tracking de la question 1 : ctr = greenball_tracker.track(image)
Ensuite, si il a trouvé le centre il va calculer la distance entre le centre de la boule et le centre de leur image. Il va ensuite, grâce à un correcteur de type PID calculer les vitesses des moteurs à envoyer au drone pour qu’il bouge de manière à recentrer la boule verte.
Pour utiliser le correcteur PID, il faut stocker les valeurs précédentes pour “lisser” le comportement du drone.
Vis à vis de la distance entre la balle et la caméra du drone elle n’est ici pas gérée, en effet il n’est ici que du centre de la tache verte qu’est la balle verte sur le retour image de la caméra d

drone. Le drone sait donc si la balle n'est pas centrée sur son axe de caméra et peut ainsi ajuster sa position. En revanche il ne sait pas à quelle distance est la balle.

Question 5:

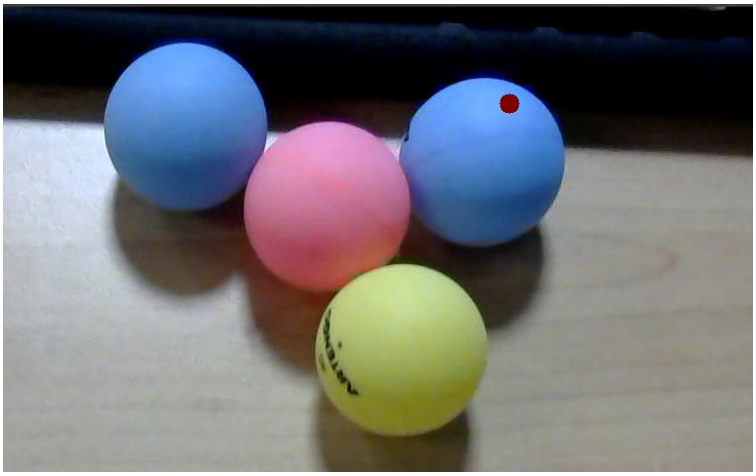
Si on essayait avec 2 boules cela ne marcherait pas correctement, en effet la fonction qui doit trouver le centre la balle verte retournerait le milieu du centre des deux boules. Un peu comme dans notre code avec les 2 boules bleus au début lorsqu'on traque le bleu :

En effet, ce comportement ne semble pas pertinent car le drone va just vouloir centrer le le point équidistant des 2 boules vertes et ce n'est pas ce que l'on souhaite.

Le code:https://github.com/JJLewis/ColorTracking-ARDrone2.0-Python/blob/master/Code/track.py Donne en effet une solution à ce problème, grâce à la détection de contour il va identifier les objets, puis il va uniquement analyser l'objet le plus gros, et donc le plus près si les boules sont identiques.

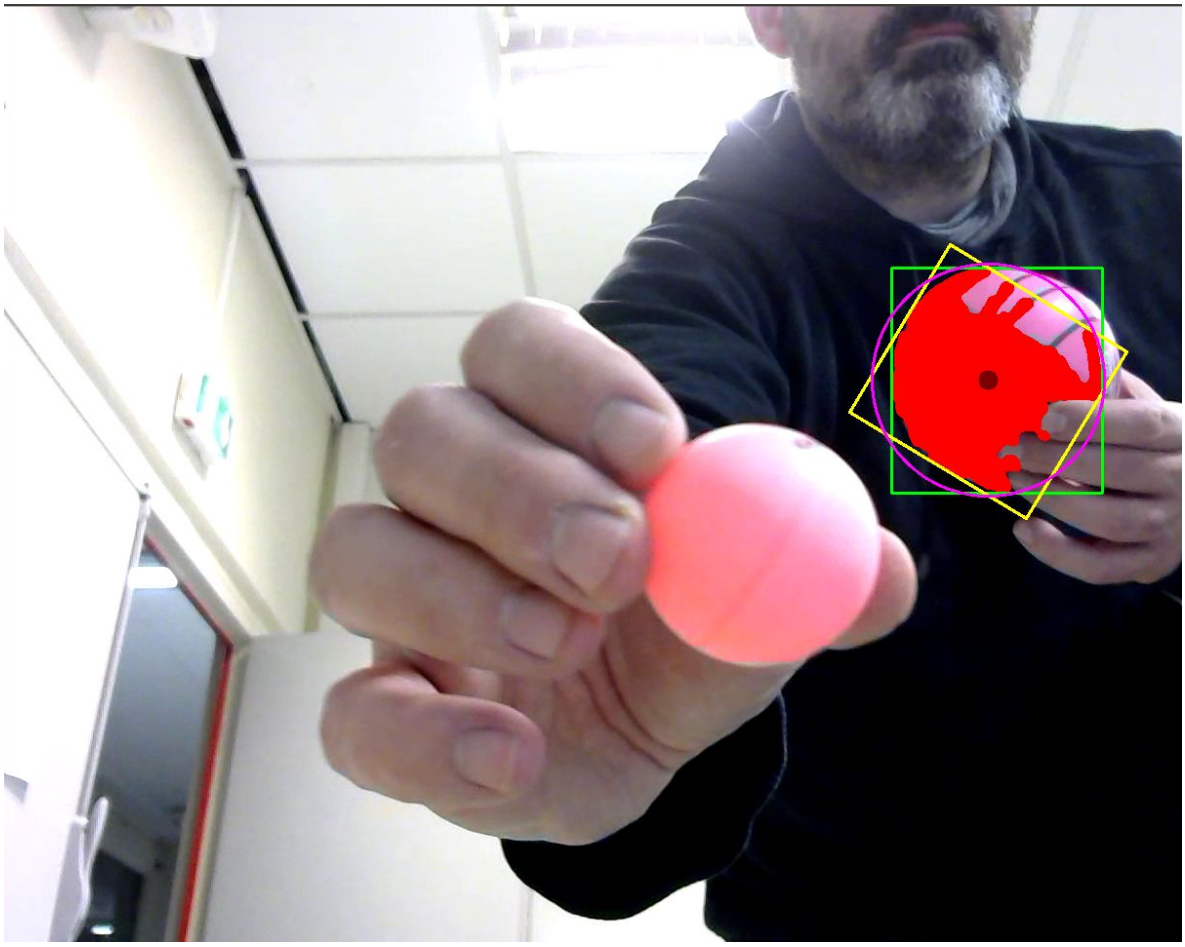
Question 6:

En s'inspirant du code étudié dans la question 5 on arrive au comportement voulu : il détecte 2 boulles blues et se place sur la plus grosse d'entre elle (voir code tp1_q6.py) :

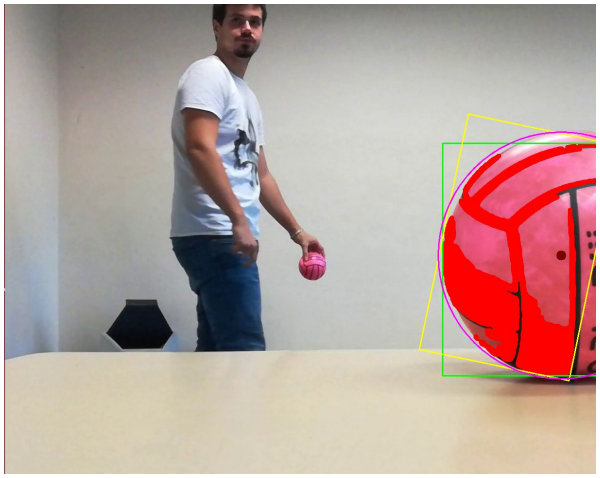


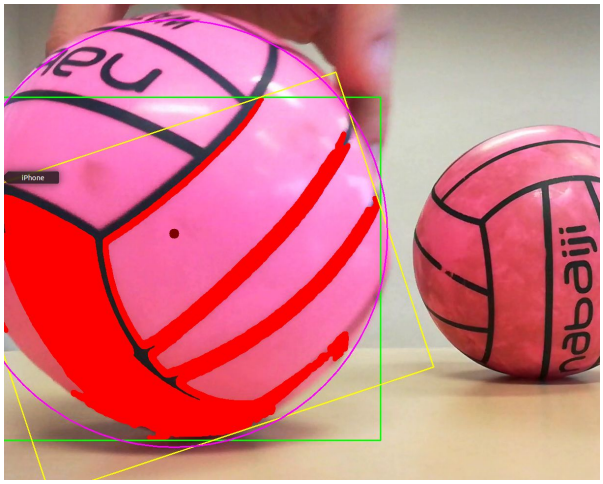
Question 7:

En s'inspirant du code de la question 4, on teste notre code avec la vidéo ball4.mp4. Dans cette vidéo nous avons 2 balles roses, notre code va identifier et suivre la balle la plus grosse à l'écran.

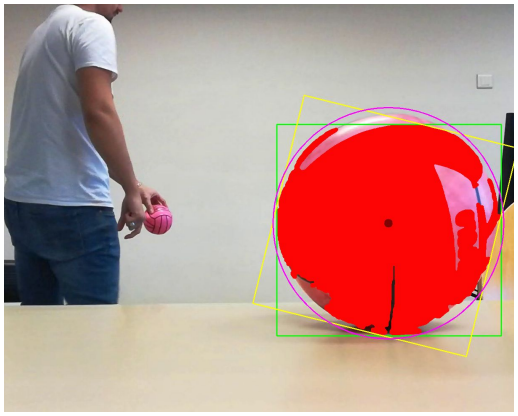


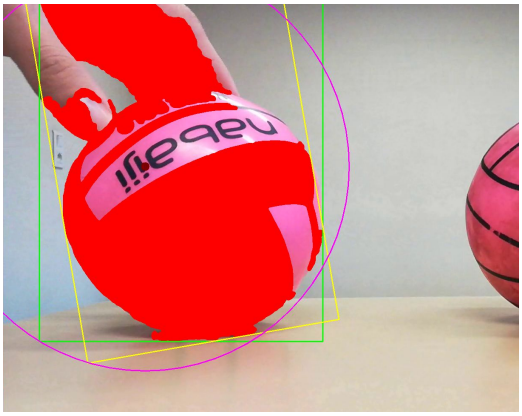
Notre code identifie bien la balle la plus grosse sur l'image, mais dans cette vidéo la balle la plus grosse reste toujours la même au cours du temps. On a donc tourné une autre vidéo pouvant tester la variation de la balle la plus grosse dans le temps et voir si notre code est capable de changer de "cible".





On lance notre scrypt python2 sur la vidéo "ballchap.mp4", on voit bien que dans le temps la balle sélectionnée varie. Cependant la balle étant très dur à identifier dans sa globalité (traits noirs et très grosses variations lumineuses) nous n'obtenons pas des résultats très satisfaisant. Il faudrait modifier les valeurs "upper" et "lower" de notre code pour les couleurs hsv servant à l'identification des pixels pour mieux coller à cette vidéo. Cependant cette modification spécifique pourra modifier le comportement de notre code sur une autre vidéo (identifier la peau en rose par exemple)





Question 8:

Afin de suivre tout au long de l'analyse le premier objet le plus gros trouvé par notre script même si un autre objet plus gros apparait à l'écran on peut restreindre en temps réel (c'est à dire à chaque appel de la fonction track) la zone d'analyse de l'image. On pourrait ainsi se

concentrer uniquement autour des points précédemments trouvés et ne pas s'intéresser à un autre objet rentrant dans la vidéo.

Question 9:

Le code permettant de réaliser cette fonction s'appelle "tp1_q8.py" il est similaire à celui de la question 7. Cependant quelques points changent pour permettre de suivre le premier objet repéré avec la couleur voulue. Le raisonnement est le suivant : identification de l'objet qui a la couleur recherchée, on enregistre les positions de ses contours. Dans l'appel suivant de la fonction track, on va créer une image noire de la taille de la vidéo, puis nous allons y coller la partie de l'image ou l'objet a été repéré grâce au contour stockée dans une variable globale. Nous allons également affecté un delta sur ce rectangle représentant les contours en faisant bien attention (grâce à une vérification) que le rectangle des contours reste toujours dans l'image. Nous allons ensuite dérouler le script sur l'image reconstituée (fond noir + contours). Ainsi le script ne va travailler que sur l'objet précédemment trouvé et non sur la vidéo dans son intégralité.
On affiche ensuite de la même manière sur l'image de départ la détection de l'objet coloré :

On a ici les 2 images, celle noire qui va être analysé par le script, et celle qui va être affichée : la détection de l'objet et le suivi unique de celui-ci.

Cependant ce code n'est efficace que dans des cas "simples" avec une vitesse de l'objet faible, pas de croisement avec des objets de la même couleur, de disparition momentanée etc...

Question 10:

Ce code (select_color.py) permet à l'utilisateur de sélectionner une zone de l'image:

```
def click_and_crop(self,event, x, y, flags, param):
        # grab references to the global variables
        # if the left mouse button was clicked, record the starting
        # (x, y) coordinates and indicate that cropping is being
        # performed
        if event == cv2.EVENT_LBUTTONDOWN:
                self.refPt = [(x, y)]
                self.cropping = True

        # check to see if the left mouse button was released
        elif event == cv2.EVENT_LBUTTONUP:
                # record the ending (x, y) coordinates and indicate that
                # the cropping operation is finished
                self.refPt.append((x, y))
                self.cropping = False

                # draw a rectangle around the region of interest
                cv2.rectangle(self.image, self.refPt[0], self.refPt[1], (0, 255, 0), 2)
                cv2.imshow("image", self.image)
```

Par la suite nous effectuons une moyenne sur les valeurs RGB de la zone sélectionnée, pour enfin les convertir en HSV grâce à la fonction opencv : cv2.COLOR_BGR2HSV()

```
    if len(self.refPt) == 2:
        roi = clone[self.refPt[0][1]:self.refPt[1][1], self.refPt[0][0]:self.refPt[1][0]]
        color = self.image[self.refPt[0][0], self.refPt[1][1]]
        # if image type is b g r, then b g r value will be displayed.
        # if image is gray then color intensity will be displayed.
        average = roi.mean(axis=0).mean(axis=0)
        # average=np.array([int(i) for i in average])
        average=average.astype(int)
        color=np.uint8([[average]])
        hsv_color = cv2.cvtColor(color,cv2.COLOR_BGR2HSV)
        cv2.destroyAllWindows()
        return hsv_color
```

Conclusion :

Durant ce TP nous avons pu comprendre comment fonctionne la reconnaissance d'objet et nous avons pu utiliser et jouer avec quelques paramètres. C'était un TP très intéressant pour voir les raisonnement à adopter dans des problèmes de vision et comment les mettre en place, notamment en python.