

Compte rendu - Perception

Introduction

L'objectif du projet est de détecter automatiquement une balle dans une image, une vidéo, ou même en direct à partir d'une webcam, en utilisant OpenCV.

L'approche repose sur l'analyse des caractéristiques colorimétriques d'une balle à partir d'exemples d'images diverses.

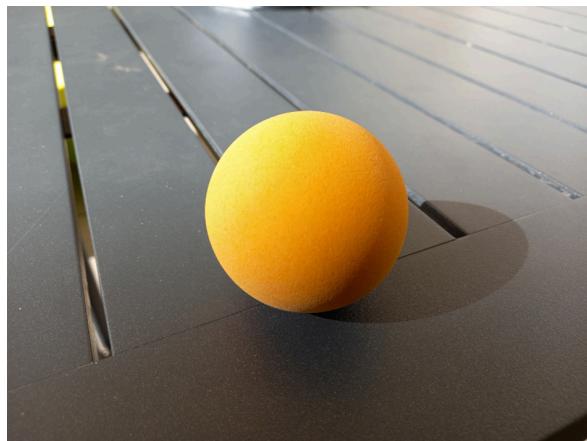


Figure 1 : Image de la balle

I - Premiers pas

La première étape a consisté à comprendre les composantes d'une image en décomposant celle-ci selon les espaces RGB et HSV. L'espace RGB dépend fortement de la luminosité, ce qui rend la détection de couleurs instable (le jaune change selon la lumière ou l'ombre). On s'intéressera plutôt à l'espace HSV, qui est beaucoup plus adapté à la détection de couleurs, car on peut isoler la teinte d'une balle (correspondant au H), indépendamment des variations de lumière (intensité lumineuse V et pureté S).

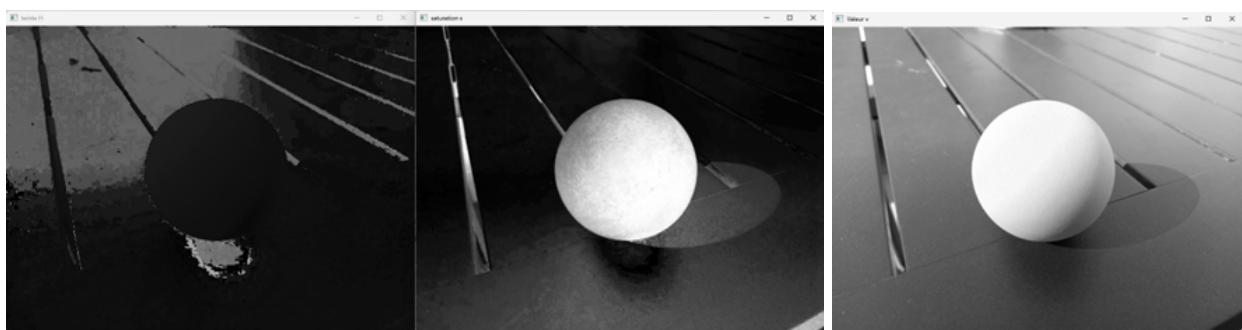


Figure 2 : Affichage des composantes HSV

Ainsi, on fixe une marge de tolérance pour les variations naturelles du jaune selon l'éclairage. J'ai d'abord choisi $\epsilon_H = 10$, $\epsilon_S = 80$ et aucune indication sur le V afin d'éviter de rejeter les zones lumineuses où la balle reste visible même très claire.

Le code réalise donc une conversion RGB → HSV, une sélection de couleur à l'aide d'un clic souris grâce à la fonction mouse_callback, puis un calcul de distance euclidienne entre la couleur choisie et chaque pixel de l'image. Je me sers ensuite d'un masque binaire (les pixels appartenant à la couleur cible = 1, et le reste = 0) pour détecter les zones susceptibles de contenir la balle.

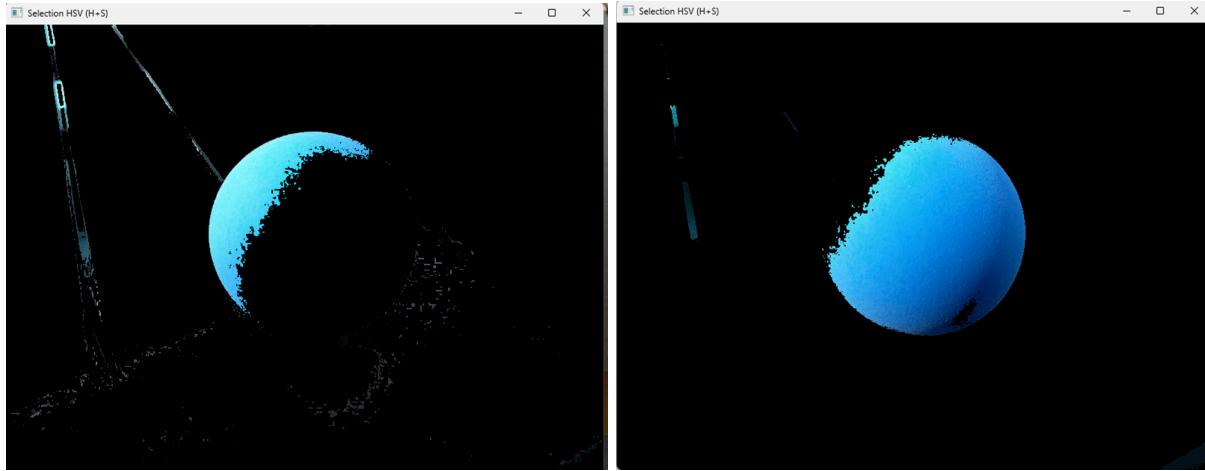


Figure 3 : Résultats différents selon l'endroit cliqué (partie sombre et éclairée de la balle)

On remarque des limites directement : lorsque je clique sur les zones très claires, la détection devient imprécise, et complètement différente lorsque je clique en dehors de la balle. Si je joue avec les tolérances de H et de S, il peut arriver de capter d'autres endroits de l'image ne correspondant même plus à la balle. Le contraste avec le fond est aussi important: les valeurs de V peuvent se confondre, et une balle jaune sur fond gris comme ici est difficile à isoler en vue de la teinte H :

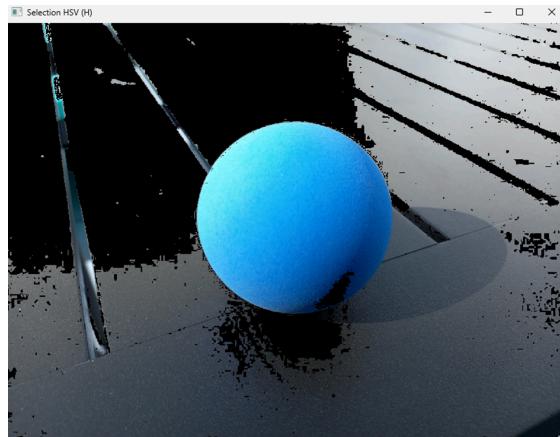


Figure 4 : Teinte H de la balle (clic sur la balle avec une petite tolérance H)

II - Détection de la balle

J'ai alors essayé de détecter une balle sur une image. Je me concentre sur les balles jaunes afin d'éviter le fait de devoir cliquer sur la balle obligatoirement pour pouvoir la détecter, auquel cas si on ne le fait pas la détection devient inutile :

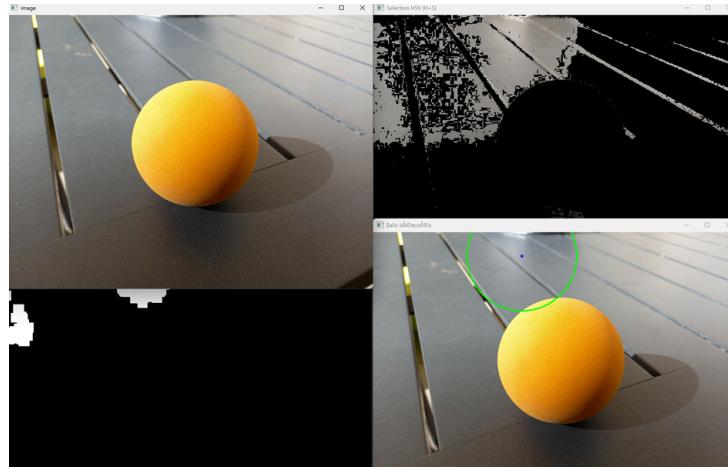


Figure 5 : Problème de détection visible (clic en dehors de la balle)

Dans le code final, on applique donc un masque HSV, une ouverture (érosion + dilatation) pour éliminer le bruit, une fermeture pour combler les petits trous dans les zones détectées, puis la fonction `findContours` pour extraire les contours des zones jaunes.

Pour chaque contour, un cercle minimal (`minEnclosingCircle`) est calculé afin de déterminer le centre et le rayon de la balle. Le clic souris (`mouse_callback`) déclenche cette détection et affiche la balle repérée avec son centre en rouge et son contour en vert :

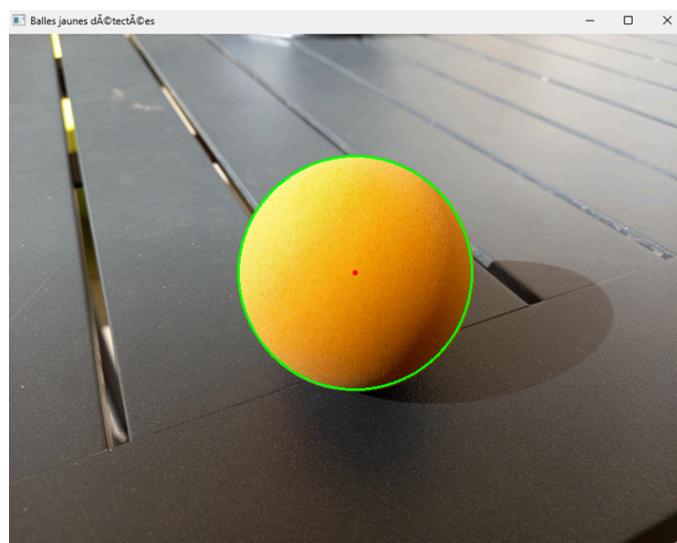


Figure 6 : Détection finale de la balle

Le système fonctionne très bien sur des balles jaunes isolées, même sur plusieurs images différentes :

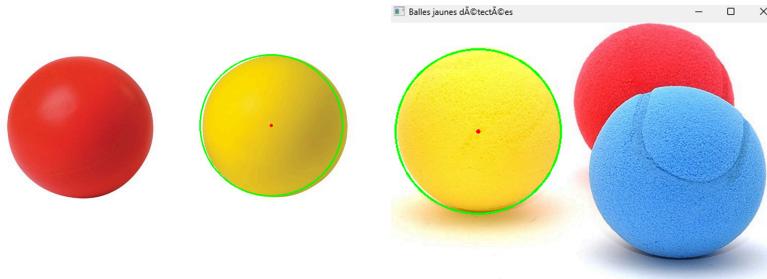


Figure 7 : Détection fonctionnant sur d'autres images

On remarque également que le nettoyage morphologique supprime efficacement les petits artefacts du 1er masque jaune :

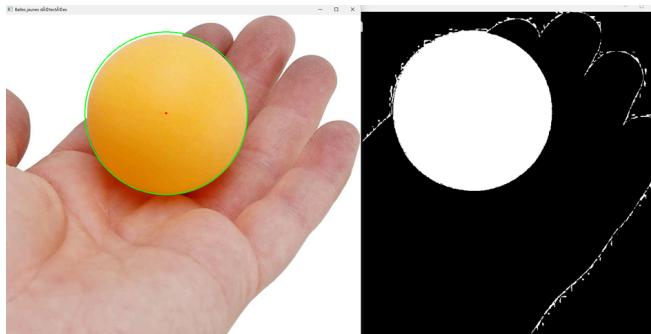


Figure 8 : Résultat de détection d'une balle et le masque jaune avant ouverture

Cependant, si plusieurs balles jaunes sont proches, elles peuvent être fusionnées en une seule forme :



Figure 9 : Résultat de détection de plusieurs balles proches l'une d'autre

La détection dépend aussi fortement des seuils HSV, donc de la luminosité et de la balance des couleurs. J'ai ensuite amélioré mon code avec une interface "Trackbars" pour ajuster ces seuils en temps réel pour pouvoir trouver les meilleurs paramètres, fonctionnant dans la plupart des cas. J'ai donc ici choisi H entre 15 et 50 (correspondant à la couleur jaune), S entre 150 et 255, et V entre 212 et 255.

III - Détection sur vidéo

En adaptant le code précédent, j'ai ensuite étendu la détection de balles jaunes à une vidéo. Le principe reste le même : convertir chaque frame en HSV, appliquer un masque jaune, puis détecter les contours circulaires correspondant aux balles.

J'ai modifié plusieurs parties du code précédent :

- Au lieu de lire une image unique avec `cv2.imread`, la vidéo est traitée image par image via `cv2.VideoCapture`. La boucle principale lit ainsi chaque frame (`ret, frame = cap.read()`).
- Les opérations de conversion (`cv2.cvtColor`) et de détection sont faites à chaque itération, contrairement à l'image fixe où elles sont faites une fois.
- J'ai aussi diminué S min (de 150 à 100) pour réduire la confusion avec des objets verts clairs. Par ex, la chaise verte de la vidéo pouvait par moments être perçue comme jaune, ce qui n'est plus le cas maintenant.

Le programme affiche ainsi la frame originale, le masque, et la détection finale, mise à jour à chaque image :

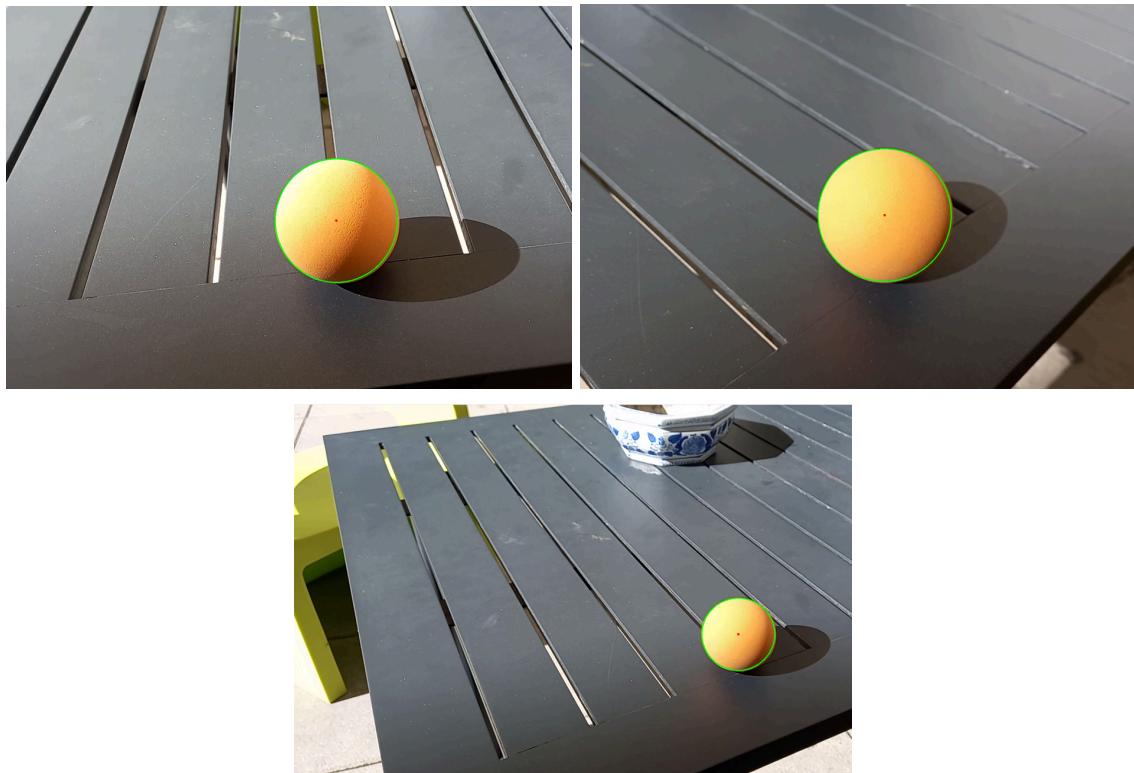


Figure 10 : Résultat de détection de la balle sur une vidéo

La détection fonctionne très bien, même sur des vidéos variées comme celle des boules de billard ci dessous :

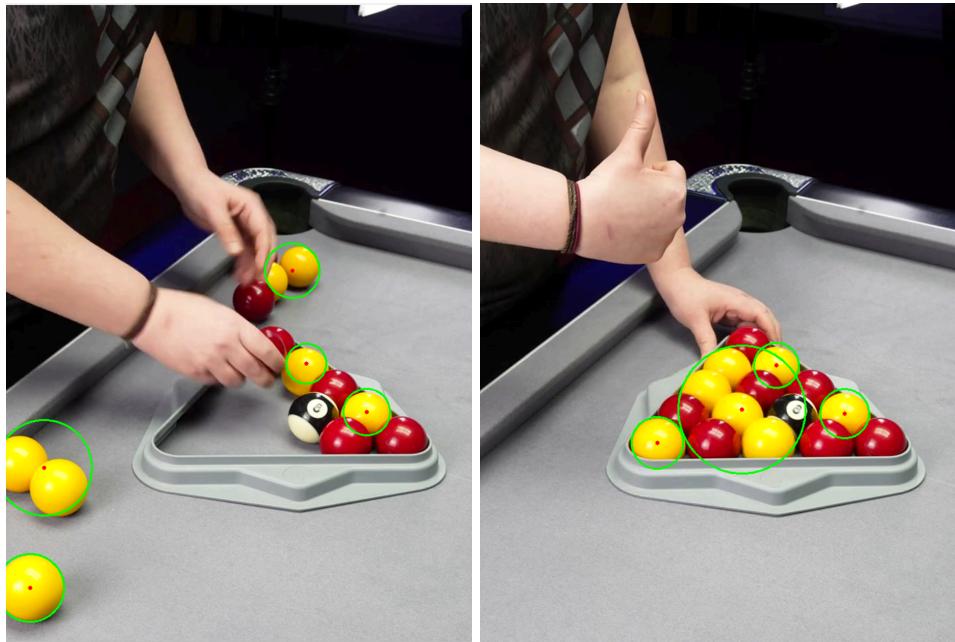


Figure 10 : Résultat de détection sur une vidéo de billard

Le seul problème persistant reste la fusion de plusieurs balles proches en une seule zone détectée, déjà observé sur les images fixes.

IV - Détection sur webcam

J'ai usé de la même logique pour une capture en direct via webcam. J'ai seulement remplacé la lecture de vidéo par `cv2.VideoCapture(0)` (source caméra), et changé la vitesse d'exécution avec : `key = cv2.waitKey(1) & 0xFF`. Il attend seulement 1 ms au lieu de 30ms entre chaque frame pour avoir un traitement en temps réel. En résultat, sur des balles bien éclairées, la détection est fluide et plutôt stable. Lorsque plusieurs balles collées sont à l'écran, il peut ne détecter qu'une partie comme précédemment mais reste performant sur les balles seules.

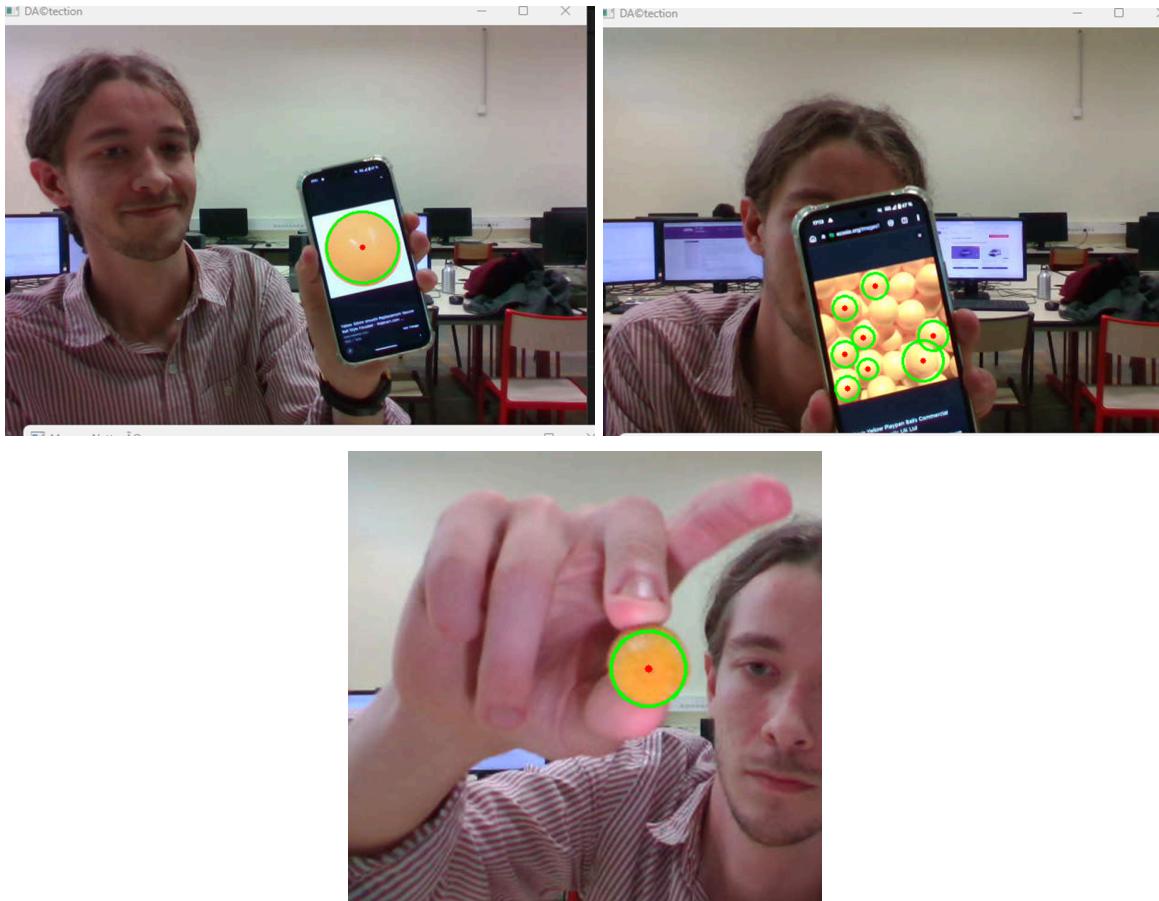


Figure 11 : Résultats de détection via la webcam

Les ajustements en direct via les trackbars HSV m'ont permis d'expérimenter et de vraiment comprendre l'influence de chaque paramètre sur la détection.

Conclusion

Ce projet de perception avec OpenCV, m'a permis de développer un système capable de détecter automatiquement une balle jaune sur une image fixe, une vidéo ou même en temps réel. On peut tout de même voir plusieurs principaux axes d'amélioration pour la suite, comme la séparation de plusieurs balles collées, une meilleure gestion de la détection face à la lumière, ou même la possibilité de détecter une balle de n'importe quelle couleur.