Series 5 – Camera

Introduction

Dans cette série, on s'intéresse à la détection de différents objets grâce à la caméra présente à l'avant du robot. L'objectif principal des exercices suivants est de récupérer et interpréter les différentes valeurs de couleurs perçues par la caméra pour en déduire la véritable couleur de l'objet.

Durant toute la série, la définition de l'image de la caméra est fixée à 100 pixels de largeur et seulement 4 pixels de hauteur, car les variations dans la hauteur importent peu.

Développement

La petite bibliothèque utilitaire développée pour les séries précédentes a été augmentée de fonctions permettant de manipuler la caméra. Le nouveau fichier source est `util/camera.c`, avec son fichier d'en-tête `util/camera.h`.

Pour le moment, seules deux fonctions ont été définies : une première permettant d'initialiser la caméra, et une seconde renvoyant la couleur moyenne de l'image actuelle.

Ces nouvelles fonctionnalités n'ont toutefois pas été utilisées dans le premier exercice, car le code fourni était déjà suffisamment complet pour réaliser l'expérience.

1 Camera measurements

1.1 Mise en place

L'objectif de cet exercice consiste à mesurer les valeurs des couleurs captées par la caméra pendant plusieurs *steps* et d'en générer des graphes. Le code fourni et le script Python sont suffisants pour réaliser l'expérience sans nécessiter d'ajouts particuliers.

1.2 Exécution

L'expérience s'est déroulée en plaçant le robot devant différents objets : un mur de l'arène, un bloc rouge, un bloc vert, un bloc bleu et un second e-puck inactif. Le robot utilisé pour cet exercice est le modèle n° 3461.

À la fin de chaque exécution, un graphe est généré. On peut observer, pour chaque composantes (rouge, vert, bleu) ainsi que pour le niveau de gris, la **quantité** de pixel en pourcentage (axe vertical) pour une certaine **intensité** de la couleur (axe horizontal).

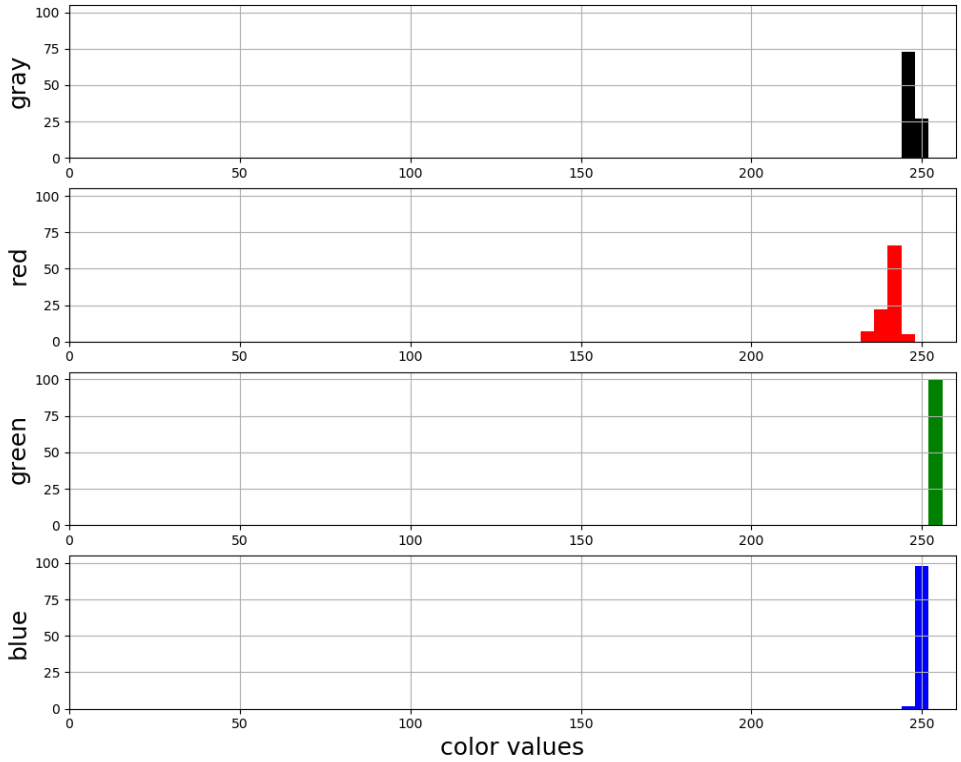

FIGURE 1 – Mesure des couleurs du mur blanc de l'arène

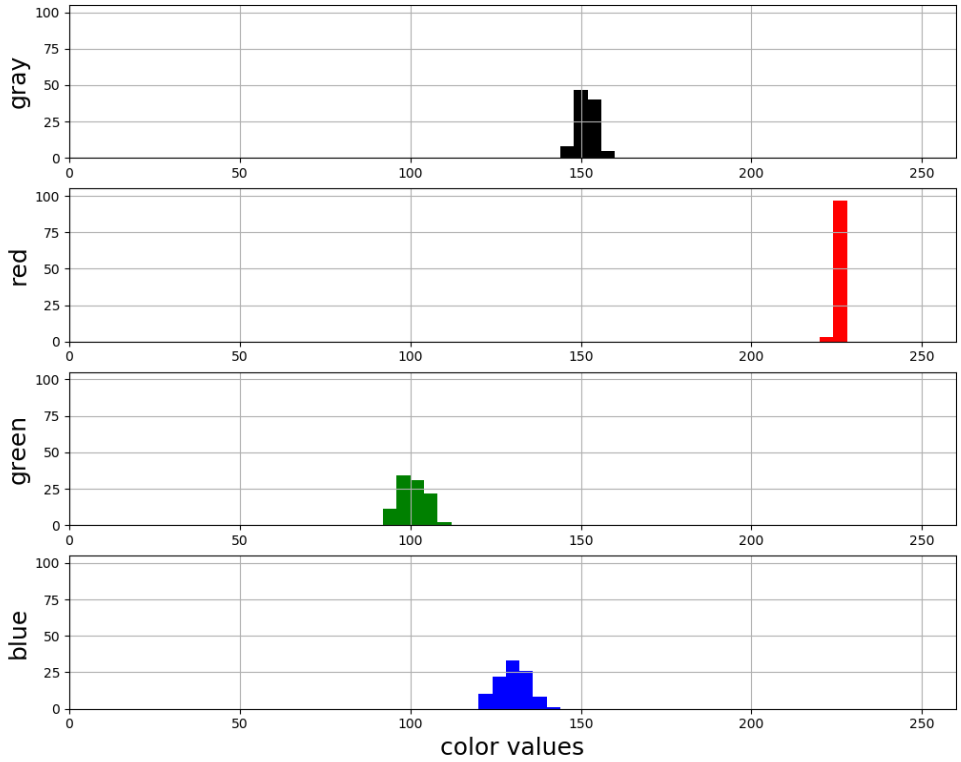

FIGURE 2 – Mesure des couleurs du bloc rouge

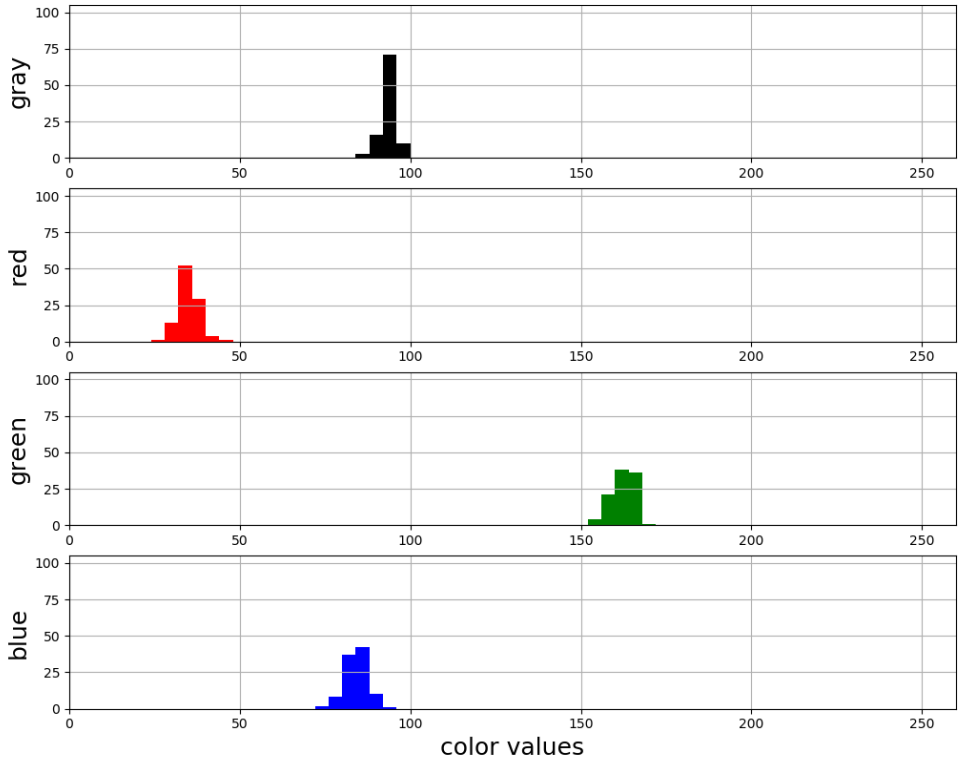

FIGURE 3 – Mesure des couleurs du bloc vert

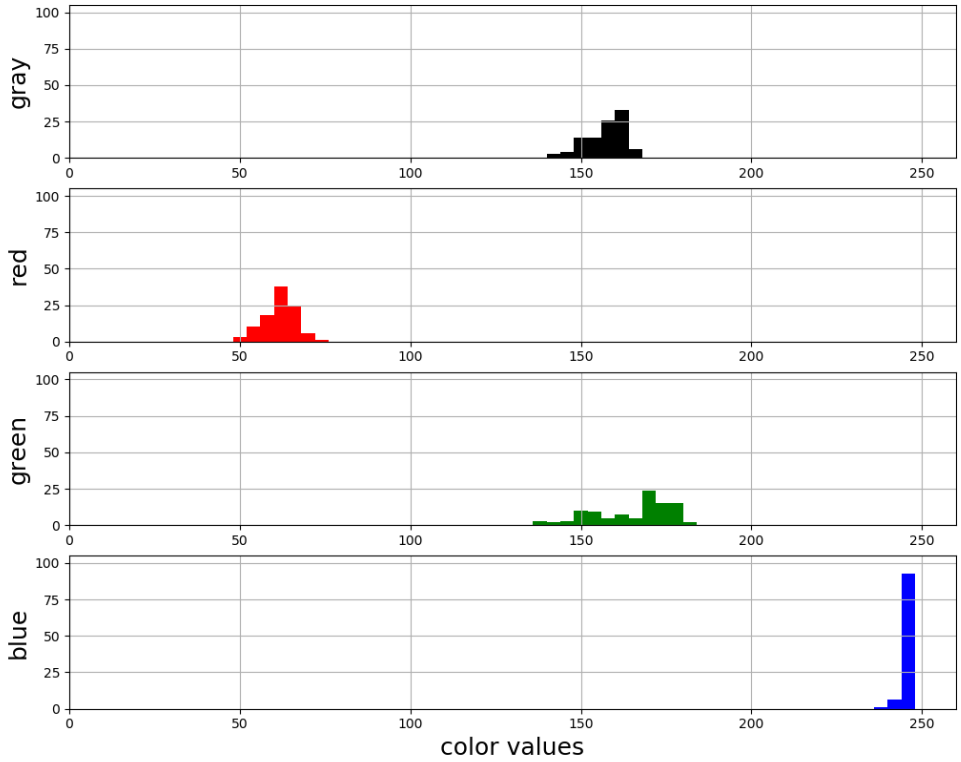

FIGURE 4 – Mesure des couleurs du bloc bleu

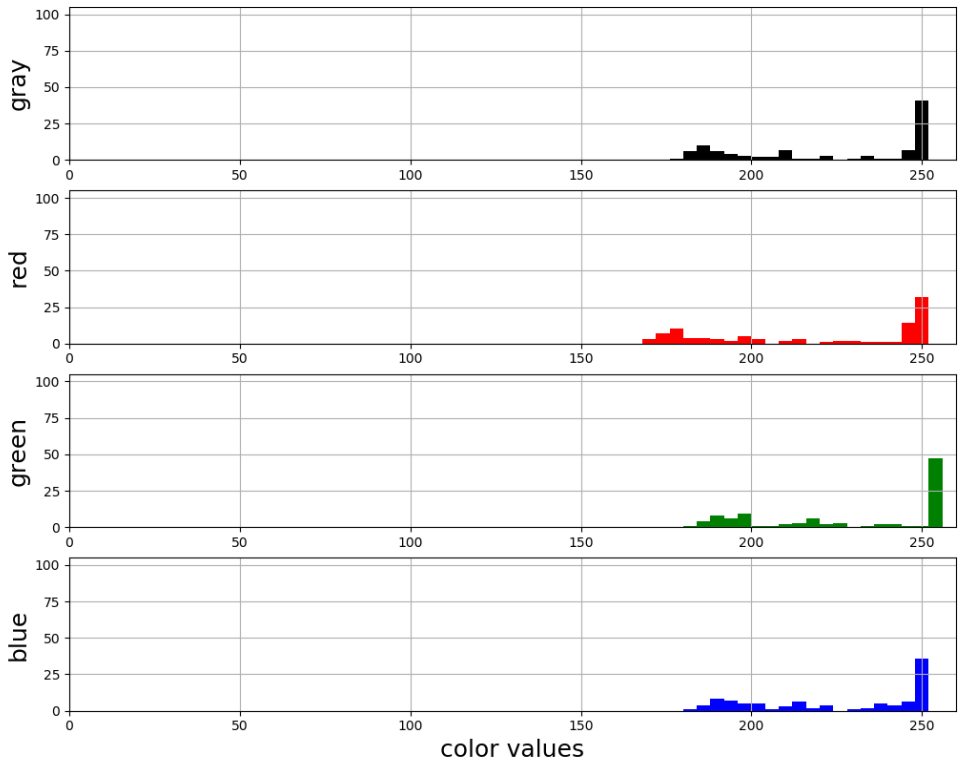

FIGURE 5 – Mesure des couleurs de l'e-puck inactif

1.3 Observation

Les résultats correspondent bien à ce dont on pouvait s'y attendre : la couleur blanche du mur est reflétée par une haute intensité de chaque composante ; les blocs rouge, vert et bleu mettent en évidence la composante correspondant à leur couleur ; quant à l'e-puck inactif, l'intensité des composantes est assez semblable, il s'agit donc d'un gris clair.

On notera toutefois quelques constatations :

— La couleur détectée semble être influencée par différents facteurs (il s'agit sans doute de variation de lumière, d'ombre ou de reflet), et ce parfois de manière assez considérable. Cette observation s'est faite principalement grâce à l'image de la caméra visible directement dans l'interface de Webots.

— Le rouge a tendance a être détecté comme une couleur tendant légèrement vers le magenta (cf. figure 2). Par moment, dans certaines situations, le rouge est perçu comme un magenta très saturé et très clair.

— Le bleu, quant à lui, tend légèrement vers le cyan (cf. figure 4). De la même manière, il arrive que la couleur détectée soit un cyan très saturé et très clair (avec parfois la composante verte à une intensité maximale).

— Les composantes bleues et vertes semblent en général assez rapprochées. Différencier la couleur verte du bleu pourrait alors être plus compliqué que la différencier du rouge.

2 Color recognition

2.1 Mise en place

Dans cet exercice, il s'agit de développer un contrôleur capable de détecter les blocs de couleur rouge, verte et bleue, ainsi que l'absence d'une telle couleur. Le robot allume une LED (gauche pour rouge, centrale pour vert, droite pour bleu) lorsqu'une couleur est détectée, ou aucune LED dans le cas contraire. Aucun mouvement du robot n'est nécessaire.

La première étape consiste donc à récupérer la couleur moyenne (i.e. la valeur moyenne de chaque composante) de l'image actuelle. Cette information s'obtient grâce à la fonction `camera_get_average_color`, définie dans `util/camera.c`, renvoyée dans une structure de type `rgb_color` (une simple structure encapsulant les trois composantes RVB). Pour y parvenir, cette fonction effectue simplement une moyenne arithmétique à partir de tous les pixels de l'image actuelle.

À partir de là, il faut trouver un moyen de déduire si la couleur actuelle correspond bien à un des trois cas à détecter. Pour ce faire, différentes méthodes ont été testées.

2.1.1 Tests sur les composantes RVB

Une première méthode naïve consiste à tester si chaque composante respecte différentes contraintes, telles que : se trouver dans un intervalle défini, ne pas trop différer d'une autre composante, ne pas être trop proche d'une autre composante, etc.

La définition de ces contraintes se fait donc en fonction des graphes générés à l'exercice 1. En réalité, cette tâche est assez laborieuse car le nombre de contraintes est élevé et elles sont souvent peu intuitives.

2.1.2 Tests sur les composantes TLS

Une seconde méthode consiste à convertir les composantes RVB en composantes TSL (teinte, saturation et luminosité). Les contraintes sont alors plus simple à imaginer, car les composantes TSL reflètent mieux la réalité à nos yeux. Pour définir ces contraintes, les intensités des couleurs des graphes générés à l'exercice 1 ont été converties en composantes TSL (à l'aide d'une application) afin de définir les bornes des intervalles.

Pour réaliser cette méthode, ces contraintes d'intervalles ont donc été définies dans la boucle principale du contrôleur. Pour ce faire, une macro `is_between(value, min, max)` a été définie afin de tester si une valeur se trouve bel et bien entre deux bornes.

```
96  leds_set(false);
97
98  if(is_between(color.h, RED_H_MIN, RED_H_MAX) &&
99     is_between(color.s, RED_S_MIN, RED_S_MAX) &&
100    is_between(color.l, RED_L_MIN, RED_L_MAX))
101     led_set(RED_LED, true);
102
103 else if(/* idem... */)
104     led_set(GREEN_LED, true);
105
106 else if(/* idem... */)
107     led_set(BLUE_LED, true);
```

Il ne reste qu'à définir et ajuster les valeurs constantes des bornes de chaque composante.

$$\text{rouge} \begin{cases} t \in [0.85, 1] \\ s \in [0.2, 1] \\ l \in [0.1, 0.9] \end{cases} \quad \text{vert} \begin{cases} t \in [0.25, 0.45] \\ s \in [0.3, 1] \\ l \in [0.1, 0.9] \end{cases} \quad \text{bleu} \begin{cases} t \in [0.45, 0.7] \\ s \in [0.3, 1] \\ l \in [0.1, 0.95] \end{cases}$$

Pour convertir les composantes RVB en TSL, une fonction `to_hsl` a été implémentée en s'inspirant d'une formule trouvée sur le Web. Cette fonction utilise les macros `get_max(a, b)` et `get_min(a, b)` qui permettent d'obtenir respectivement la plus grande et la plus petite de deux valeurs, et renvoie une structure de type `hsl_color`, fondamentalement semblable à `rgb_color`, mais avec des valeurs de composante $\in [0, 1]$.

```
46 hsl_color to_hsl(rgb_color input)
47 {
48     unsigned max = get_max(get_max(input.r, input.g), input.b);
49     unsigned min = get_min(get_min(input.r, input.g), input.b);
50 
51     float maxf = (float) max / 255;
52     float minf = (float) min / 255;
53 
54     float hue = 0;
55     float sat = 0;
56     float light = (maxf + minf) / 2;
57 
58     if(max != min)
59     {
60         float rf = (float) input.r / 255;
61         float gf = (float) input.g / 255;
62         float bf = (float) input.b / 255;
63         float delta = maxf - minf;
64 
65         if(light > 0.5)
66             sat = delta / (2 - maxf - minf);
67         else
68             sat = delta / (maxf + minf);
69 
70         if(max == input.r)
71             hue = (gf - bf) / delta + (gf < bf ? 6 : 0);
72         else if(max == input.g)
73             hue = (bf - rf) / delta + 2;
74         else if(max == input.b)
75             hue = (rf - gf) / delta + 4;
76         hue /= 6;
77     }
78 
79     return (hsl_color) {hue, sat, light};
80 }
```

Le contrôleur correspondant à cet exercice est `S05_Color_Recognition`. Le code source de celui-là se trouve dans le fichier `S05_Color_Recognition.c`.

2.2 Exécution

Le robot utilisé pour cet exercice est à nouveau le modèle n° 3461.

L'expérience a montré que la seconde méthode est bien plus efficace que la première et particulièrement plus simple à mettre en place et à ajuster. En effet, la première méthode interprète mal les légères variations de teinte ou de luminosité, et il n'est pas évident d'ajuster les contraintes sur les composantes RVB pour corriger ces problèmes.

Avec la seconde méthode, les bornes des intervalles ont pu être aisément ajustées. Le robot parvient donc sans souci à détecter une couleur de bloc et à allumer la LED correspondante, et ce avec différentes variations.

2.3 Observation

On peut observer que l'utilisation de la caméra crée une latence assez considérable. Étrangement, cette latence semble plus ou moins forte en fonction de l'ordinateur sur lequel tourne Webots. Sur un ordinateur portable, on a pu observer un facteur de vitesse du robot réel (affiché à droite du bouton « run ») à une valeur inférieure à 0.05, ce qui est extrêmement faible. En revanche, sur un ordinateur du laboratoire, le facteur est bien plus élevé (environ 0.30).

3 Color recognition (bis)

3.1 Mise en place

Cet exercice est une simple combinaison de l'exercice précédent (*Color recognition*) et de l'exercice 2 de la série 2 (*Explorer & Advanced Lover*). En effet, il s'agit de développer un contrôleur effectuant un comportement « explorer » tout en donnant des informations sur la couleur de l'objet qui fait face au robot.

En plus de cela, il est demandé d'exécuter le contrôleur sur deux robots différents. Pour y parvenir, il suffit d'apparier deux robots via Bluetooth et de relier les deux ports aux e-puck dans l'environnement de Webots.

Le code pour cet exercice est une simple copie et fusion des codes développés pour les deux exercices cités ci-dessus. Le seul changement nécessaire est la vitesse du robot redéfinie à 2.5 rad/s (au lieu de 4.0 rad/s comme précédemment), car la latence causée par l'utilisation de la caméra empêche le robot de réagir suffisamment vite à grande vitesse.

Le contrôleur correspondant à cet exercice est `S05_Color_Recognition_Bis`. Le code source de celui-là se trouve dans le fichier `S05_Color_Recognition_Bis.c`.

3.2 Exécution

Les robots utilisés pour cet exercice sont les modèles n° 3461 et n° 3426.

L'expérience se déroule sans imprévu. La qualité du comportement « explorer » et de la reconnaissance de couleur est techniquement identique à celle des exercices déjà effectués.

Le seul léger problème est l'application du comportement « explorer » lorsque les deux robots se rencontrent. En effet, le comportement n'a pas été prévu pour réagir à des obstacles mobiles. En conséquence, il arrive que les deux robots s'entrechoquent légèrement avant de s'éloigner l'un de l'autre.

3.3 Vidéo

Une vidéo des deux robots en exécution est disponible à l'adresse suivante. À cause de la faible vitesse des robots, elle a été légèrement accélérée.

https://youtu.be/mPN5DWDjdfs

On peut observer que les robots sont en principe capables de détecter les couleurs alors qu'ils sont encore loin de l'objet duquel ils s'approchent.