



## Sommaire

- 1 Objectifs
- 2 A quoi sert la détection par couleurs ?
- 3 Ouverture de la Caméra (Multi-sources)
- 4 Ouverture du port série
- 5 Paramètres du programme
- 6 Détection & Transfert vers Arduino
  - 6.1 Principe
  - 6.2 La fonction getObj()
  - 6.3 Exemple
  - 6.4 Boucle principale
- 7 Le programme complet

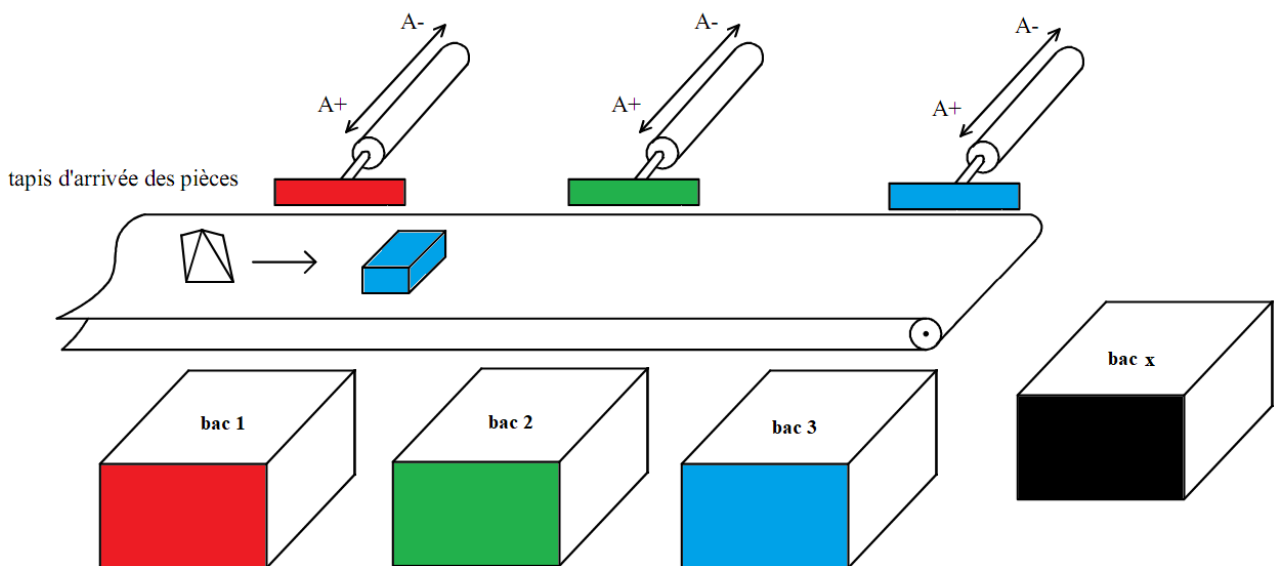
# Objectifs

- Savoir détecter un objet en fonction de la couleur
- Comprendre le principe de détection et classification par couleurs
- Savoir générer une action avec [Arduino](#) au moment de la détection
- Savoir transférer l'état du [détecteur](#) à la [carte](#) Arduino
- Savoir coder les couleurs sur 1 bit ou N bits
- Etc.



## A quoi sert la détection par couleurs ?

La couleur est une caractéristique importante d'un objet parmi d'autres. Elle peut être utilisée pour classer des objets distincts (pièces mécaniques, liquide, etc.). On distingue également la longueur, le diamètre, la forme géométrique, etc. On verra dans ce tutoriel le principe de détection en se basant sur la couleur d'un objet avec des exemples. L'exemple ci-dessous indique le tri des boîtes en fonction de leurs couleurs. Le commande de l'un des trois vérins est effectuée en fonction de la couleur des pièces. Le bac x est utilisé lorsqu'aucune vérin n'est actionnée (couleur différente). La commande des trois vérins peut être simulée avec trois LEDs.





## Ouverture de la Caméra (Multi-sources)

```
Type=1;      % 0(GRAY), 1(RGB)

Source=2;    % 1(CAM1), 2(CAM2), ...

cam=openCam(Type,Source);

% figure(1); imshow(getsnapshot(cam)); return;
```

*Le code de la fonction est disponible dans la partie #11 et #12*

## Ouverture du port série

```
namePort='COM3';

baudValue=9600;

SerialCOM = openSerial(namePort, baudValue);
```

*Le code de la fonction est disponible dans la partie #11 et #12*

## Paramètres du programme

```
M=512; N=512;
```



```
im=zeros(M,N,3);  
  
DET_RGB0=zeros(1,3);  
  
im_RGB=zeros(M,N,3);
```

# Détection & Transfert vers Arduino

## Principe

Le principe de la méthode utilisée dans ce [projet](#) est relativement simple. Il consiste à soustraire la couleur qu'on veut détecter de l'image en niveau de gris issue de l'image originale (image RGB) ! Exemple : On considère une image couleur nommée RGB, et im\_gray l'image en niveau de gris obtenue à partir de l'image RGB en utilisant la fonction de conversions rgb2gray (im\_gray=rgb2gray(RGB)). On considère im\_r l'image résultante obtenue à partir de la composante rouge im\_R. L'image im\_r est obtenue comme suit :

$$\mathbf{Im\_r=im\_R-im\_gray}$$

Les pixels les plus clairs de l'image im\_r indiquent la présence de la couleur rouge dans l'image. Les pixels sombres montrent l'absence de la cette dernière. L'image ci-dessous montre effectivement dans l'image résultante que la zone du marqueur est plus claire (valeur proche du blanc ou 255) en revanche les autres pixels de l'image ont tendance à avoir des valeurs proches de 0 (noirs). Il sera ensuite plus simple d'extraire uniquement la zone la plus claire de l'image et mettre à zéros le reste des pixels grâce au choix d'un [seuil](#) « optimal ».

## La fonction getObj()

La fonction getObj() prend en entrée une image couleur de type RGB, le seuil de détection et la composante qui intéresse ('R', 'G' ou 'B'). Le seuil est une valeur comprise entre 0 et 1. Elle agit sur la sensibilité du détecteur. La fonction retourne ensuite l'état de détection



(présence ou absence de la couleur dans l'image) et l'image binaire. Ci-dessous la définition de la fonction. La variable DET indique le nombre des pixels allumés de l'image binaire. Autrement dit, le nombre des pixels ayant la couleur visée. Le nombre des pixels allumés est étroitement liée à la couleur de l'objet ainsi sa taille dans l'image (voir le tuto pour plus de détails).

```
function [im_out, DET]=getObj(im_inRGB, Seuil, COLOR)

% Params
im_inRGB=im_inRGB/255;
[M, N, P]=size(im_inRGB);
im_out=zeros(M,N);
DET=0;

% Image NG
NG=rgb2gray(im_inRGB);

% Calcul de l'image résultante
if COLOR=='R'
    R=squeeze(im_inRGB(:,:,1));
    im_sub=imsubtract(R,NG);
    im_out=im_sub > Seuil;
    nPix=length(find(im_out(:)==1));
    DET=nPix;
    return;
end

if COLOR=='G'
    G=squeeze(im_inRGB(:,:,2));
    im_sub=imsubtract(G,NG);
    im_out=im_sub > Seuil;
    nPix=length(find(im_out(:)==1));
    DET=nPix;
    return;
end

if COLOR=='B'
    B=squeeze(im_inRGB(:,:,3));
    im_sub=imsubtract(B,NG);
    im_out=im_sub > Seuil;
    nPix=length(find(im_out(:)==1));
    DET=nPix;
```



```
return;  
end  
end
```

## Exemple

L'exemple ci-dessous fait appel à la fonction `getObj()` pour les trois composantes.

```
Seuil=0.1; % Seuil de conversion au format binaire  
numPix=10; % Seuil de comptage des pixels allumés  
[im_RGB(:,:,1), DET_RGB0(1)]=getObj(im, Seuil, 'R');  
[im_RGB(:,:,2), DET_RGB0(2)]=getObj(im, Seuil, 'G');  
[im_RGB(:,:,3), DET_RGB0(3)]=getObj(im, Seuil, 'B');
```

## Boucle principale

```
while 1  
    %% 1. Lecture de l'image courante  
    im0=getsnapshot(cam);  
    im(:,:,1)=imresize(im0(:,:,1),[M N]);  
    im(:,:,2)=imresize(im0(:,:,2),[M N]);  
    im(:,:,3)=imresize(im0(:,:,3),[M N]);
```



```
%% 2. Détection de l'objet (par couleur)

Seuil=0.1; % Seuil de conversion au format binaire
numPix=10; % Seuil de comptage des pixels allumés

[im_RGB(:,:,1), DET_RGB0(1)]=getObj(im, Seuil, 'R');
[im_RGB(:,:,2), DET_RGB0(2)]=getObj(im, Seuil, 'G');
[im_RGB(:,:,3), DET_RGB0(3)]=getObj(im, Seuil, 'B');

RGB8= DET_RGB0

% Conversion de format
DET_RGB=double(DET_RGB0>numPix)

DET_RGB(1)=10*DET_RGB(1); % Composante R: Valeur 10
DET_RGB(2)=20*DET_RGB(2); % Composante G: Valeur 20
DET_RGB(3)=30*DET_RGB(3); % Composante B: Valeur 30

%% 3. Transfert vers la carte Arduino

for j=1:3
    if DET_RGB(j)
        fprintf(SerialCOM,'%d\n',DET_RGB(j)); pause(0.1);
    end;
end;
```



```
%% 4. Affichage

figure(1);

subplot(121); imshow(im/255); title('Image Originale','fontsize',16);

subplot(122); imshow(im_RGB); title('Image Résultante','fontsize',16);

end
```

## Le programme complet

```
close all; clc; clear all;

%% Ouverture de la Caméra (Multi-sources)
Type=1;      % 0(GRAY), 1(RED)
Source=2;    % 1(CAM1), 2(CAM2), ...
cam=openCam(Type,Source);
% figure(1); imshow(getsnapshot(cam)); return;

%% Ouverture du port série
namePort='COM3';
baudValue=9600;
SerialCOM = openSerial(namePort, baudValue);

%% Paramètres du programme
M=512; N=512;
im=zeros(M,N,3);
DET_RGB0=zeros(1,3);
im_RGB=zeros(M,N,3);

%% Détection & Transfert vers Arduino
while 1
    %% 1. Lecture de l'image courante
    im0=getsnapshot(cam);
    im(:,:,1)=imresize(im0(:,:,1),[M N]);
    im(:,:,2)=imresize(im0(:,:,2),[M N]);
    im(:,:,3)=imresize(im0(:,:,3),[M N]);
    %% 2. Détection de l'objet (par couleur)
```





```
Seuil=0.1; % Seuil de conversion au format binaire
numPix=10; % Seuil de comptage des pixels allumés
[im_RGB(:,:,1), DET_RGB0(1)]=getObj(im, Seuil, 'R');
[im_RGB(:,:,2), DET_RGB0(2)]=getObj(im, Seuil, 'G');
[im_RGB(:,:,3), DET_RGB0(3)]=getObj(im, Seuil, 'B');
RGB8= DET_RGB0
% Conversion de format
DET_RGB=double(DET_RGB0>numPix)
DET_RGB(1)=10*DET_RGB(1); % Composante R: Valeur 10
DET_RGB(2)=20*DET_RGB(2); % Composante G: Valeur 20
DET_RGB(3)=30*DET_RGB(3); % Composante B: Valeur 30
%% 3. Transfert vers la carte Arduino
for j=1:3
    if DET_RGB(j)
        fprintf(SerialCOM,'%d\n',DET_RGB(j)); pause(0.1);
    end;
end;
%% 4. Affichage
figure(1);
subplot(121); imshow(im/255); title('Image Originale','fontsize',16);
subplot(122); imshow(im_RGB); title('Image Résultante','fontsize',16);
end
```

## Traitement d'Images | Matlab

Click to rate this post!

[Total: 1 Average: 5]