ANNEXE PROTIS VISION INDUSTRIELLE

Code : Acquisition de l'image

```
cam=webcam('Logitech HD Webcam C270') ; %on définit la caméra que l'on veut utiliser
preview(cam) %cette commande permet de prendre une vidéo en direct
img=snapshot(cam) ; %on prend une photo de la caméra cette photo est une variable
appelée 'img'
imagesc(img)
clear('cam') %commande à ne pas oublier pour pouvoir reprendre des photos par la suite
```

Code : Traitement de l'image

```
%% Import de l'image
cam=webcam('Logitech HD Webcam C270') ; %on définit la caméra que l'on veut utiliser
img=snapshot(cam) ; %prend une photo avec la caméra

%% Prétraitement

sigma=0.5;                          %valeur de base de l'écart-type du filtrage gaussien.
img=imgaussfilt(img,sigma);

%% Traitement (choisir l'une ou l'autre des deux méthodes)

%méthode 1 : écart-type local
 imgC=zeros(dim+2);
 imgC(2:dim(1)+1,2:dim(2)+1)=img;          %imgC=img mais avec des bords égaux à zeros
 C1=conv2(img,ones(3)/9,'same');           %Calcul de l'espérance locale
 C2=zeros(dim);
 for i=1:dim(1)
    for j=1:dim(2)
        C2(i,j)=sum((imgC(i:i+2,j:j+2)-C1(i,j)).^2,'all');          %Calcul de la variance local
    end
 end
 img_trait=img./(0.1*C1+2*C2);        %coefficients *0.1 et *2 modifiables, le terme avec la
moyenne permet d'éviter de diviser par 0.
 img_trait=ones(dim)-img_trait/max(img_trait,[],'all');  %rétablissement de l'image (les maxs
deviennent des mins et inversement).
```

```
%méthode 2 : sobel
S=[-1 0 1;-2 0 2;-1 0 1];
imgsobx=conv2(img,S,'same');                %Equivalent au gradient selon X
imgsoby=conv2(img,S',’same');               %Equivalent au gradient selon Y
imgsob=sqrt(imgsobx.^2+imgsoby.^2);
img_trait=imgsob;

%Seuil dynamique :
M=(1/25)*ones(5,5);
M=conv2(img_trait,M,'same');
t=mean(img_trait,'all')*1;      %le *1 peut être modifié afin de changer la sensibilité du
seuillage.
img_trait=M>t;
img=img_trait;
%Ce seuillage, à l’image d’opérations morphologiques, permet d’affiner les bords et
d’éliminer les points isolés.

%% détection défauts

img_bords=conv2(img,[0 1 0;1 1 1;0 1 0]/5,'same'); %chaque élément vaut 1 s’il est entouré
(intérieur) ou moins si c’est un bord. C’est la définition qu’on donne ici à un bord.
imgcpy=zeros(dim);                          %Copie de l’image sur laquelle on va travailler
imgcpy(2:dim(1)-1,2:dim(2)-1)=img(2:dim(1)-1,2:dim(2)-1);
DEFAUTS=cell(10000,2);                      %Cellule contenant les objets détectés (bords et
remplissage)
m=0;
R=[-1 -1;-1 0;-1 1;0 1;1 1;1 0;1 -1;0 -1];      %matrice de déplacement pour le remplissage
des contours plus tard
for i=2:dim(1)-1
  for j=2:dim(2)-1                  %On parcourt l’image à la recherche de pixels blancs = objets
détectés
    if imgcpy(i,j)==1               % Comprendre “S’il y a un objet détecté en i,j après seuillage”
      m=m+1;                        %Compteur d’objets détectés
      c1=1;                         %Compteur de la pile
      c2=1;                         %Compteur pour les défauts
      c3=1;                         %Compteur pour les bords
      pile=zeros(600000,2);         %Pile qui va contenir l’ensemble des pixels à “scanner”
      defaut=zeros(600000,2);       %Liste avec l’ensemble des positions des pixels de
l’objet traité [i j]
      bords=zeros(60000,2);         %Comme defaut mais pour les bords
      pile(1,:)=[i j];
      while c1>0
        imgcpy(pile(c1,1),pile(c1,2))=0;    %On remplace le dernier pixel de la pile par 0 sur
l’image avant de le “scanner”
        defaut(c2,:)=pile(c1,:);
        c2=c2+1;
        if img_bords(pile(c1,1),pile(c1,2))<1       %Critère de définition d’un bord
          bords(c3,:)=pile(c1,:);
          c3=c3+1;
        end
        bi=pile(c1,1);          %On note la position de ce dernier pixel bi et bj
        bj=pile(c1,2);
        pile(c1,:)=[0 0];       %On retire le pixel en question de la pile
        c1=c1-1;
```

```
            for k=1:8                %Enfin on “scanne” le pixel qui vient d’être retiré, c'est - à - dire qu’on regarde les pixels blancs autour de lui et on les ajoute à la pile.
                if imgcpy(bi+R(k,1),bj+R(k,2))==1
                    pile(c1+1,:)=[bi+R(k,1) bj+R(k,2)];
                    c1=c1+1;
                    imgcpy(bi+R(k,1),bj+R(k,2))=0;        %Les pixels ajoutés à la pile sont mit à zero sur l’image au fur et à mesure.
                end
            end
        end
        %On réduit la taille des listes, initialement prévues grandes, au nécessaire et on les ajoute à la cellule. Defaut contient l’ensemble des
        defaut=defaut(1:c2-1,:);
        bords=bords(1:c3-1,:);
        DEFAUTS(m,1)={defaut};
        DEFAUTS(m,2)={bords};
    end
  end
end
if m==0
DEFAUTS=0
else
DEFAUTS=DEFAUTS(1:m,:);
end

%En pratique le "remplissage" en question n'est pas le remplissage de
%l'objet mais le remplissage du contour restant après seuillage. On réalise
%cette étape afin de pouvoir élimner tous les pixels d'un contour détectée
%et donc de ne pas compter un même contours deux fois.

%% Détection de la couleur

Couleur=zeros(m,3);
if m~=0
   for i=1:m
      roi=[min(DEFAUTS{i,2}(:,1)), max(DEFAUTS{i,2}(:,1)), min(DEFAUTS{i,2}(:,2)), max(DEFAUTS{i,2}(:,2))];   %Carré contenant l'objet détecté m.
      obj=imgc(roi(1):roi(2),roi(3):roi(4),:);    %Extraction de l'objet sur l'image en couleur
      Couleur(i,:)=[sum(obj(:,:,1),'all'), sum(obj(:,:,2),'all'), sum(obj(:,:,3),'all')]/((roi(2)-roi(1)+1)*(roi(4)-roi(3)+1));  %RGB moyen sur le roi définit
   end
end

%A ce stade là, la liste Couleur contient pour chaque objet détecté sa
%décomposition RGB (moyennée sur le ROI).
%Il faut ensuite faire un fit sur cette décomposition afin d'y associer une
%couleur. Ce fit est à déterminer expérimentalement en regardant quelle
%décomposition donne chaque cube selon l'éclairage choisi etc..

%Une fois la couleur obtenue par le fit, la commande du plateau tournant
%avec les contenaires est la suivante :
%write(NUCLEO,'a','char');
%Le PC envoie alors à la carte NUCLEO 'a' par la connexion série de la même
%manière que si l'on tappait 'a' dans la console TeraTerm.
```

Code : Communication Matlab Mbed

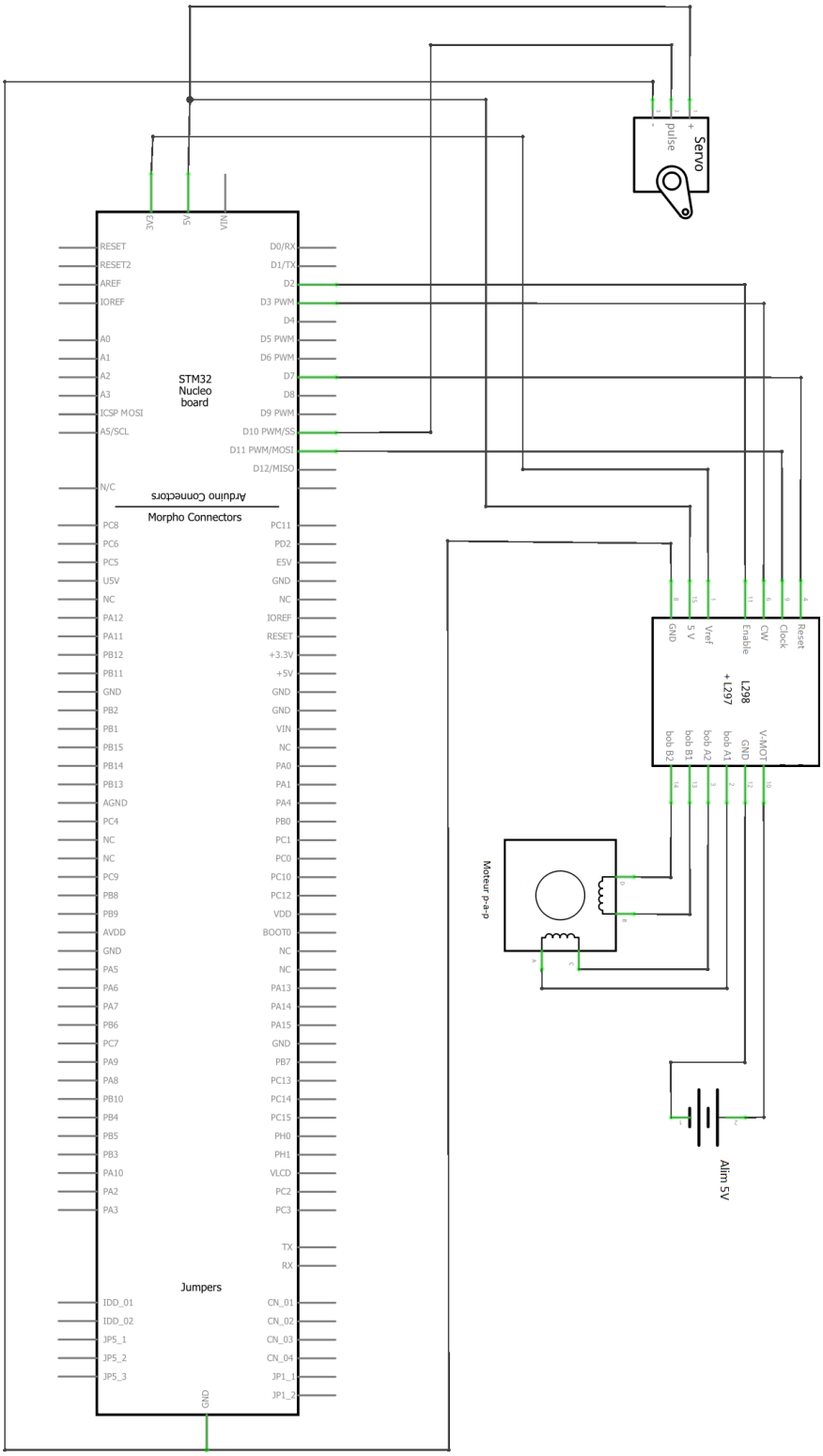
nucleo=serialport("COM30",115200); %il faut configurer le bon port en utilisant la commande serialportlist qui affiche la liste des ports séries utilisables (ici c'est le 30)
write(nucleo,'d',"char") %on envoi le caractère "d" à la carte nucléo via la connexion série, et la carte nucleo convertira la commande reçu au moteur.

Code : Commande du récipient

```
#include "mbed.h"
// inputs and outputs configuration



Serial      rs232(USBTX, USBRX);


PwmOut servo_mot(D10);      // pour le servomoteur du plateau tournant
PwmOut vitesse(D11);        // horloge du moteur pas à pas
DigitalOut Reset(D7,1);     // reset (indicateur de la carte qui doit être à
1 pour fonctionner)
DigitalOut CW(D3,1);        // indicateur booléen du sens de rotation
DigitalOut enable(D2,0);    // indicateur de fonctionnement ( doit être à 1
pour fonctionner)




// System functions
void ISR_get_data(void);    //initialisation de la fonction d'interuption
// Variables
char data_received = 0;     //  variable qui sert à la communication avec
Matlab via la liaison série
// Main function
int main() {

    vitesse.period_ms(1); // vitesse pour le moteur pas-à-pas qui convient,
trouvée à tâtons
    vitesse.write(0.7);   // 0.7 rapport cyclique de l'horloge, n'a pas
vraiment d'incidence, à moins de valeurs extrèmes



    servo_mot.period_ms(20);          // Initialisation période
    servo_mot.pulsewidth_us(1500);    // Initialisation en position 0

    rs232.baud(115200);
    rs232.attach(&ISR_get_data);
```

```
    while(1) {     }
}
void ISR_get_data(){
    data_received = rs232.getc();
    switch(data_received){

        case 'g':
            enable = 1;
            break;
        case 's':
            enable = 0;
            break;
        case 'a':
            rs232.printf("o\r\n");
            servo_mot.pulsewidth_us(450);
            break;
        case 'b':
            rs232.printf("o\r\n");
            servo_mot.pulsewidth_us(1000);
            break;
         case 'c':
            rs232.printf("o\r\n");
            servo_mot.pulsewidth_us(1600);
            break;
         case 'd':
            rs232.printf("o\r\n");
            servo_mot.pulsewidth_us(2150);
            break;


    }
}
```

Branchement Carte Nucléo, moteurs, Tapis, Récipient



Plan du récipient