ENSTA PARIS

MIV308 : RECONNAISSANCE DE FORMES

TP : Détection de véhicule

Réalisé par :
Amin HENTETI

Enseignant :
M. Taha Ridene

Année universitaire: 2019/2020

1 Introduction

Le but de ce TP est d'implanter un détecteur de véhicule à partir d'une caméra embarquée. Les images utilisées sont issues d'un simulateur de conduite et sont extraites de la base d'évaluation d'algorithmes de restauration de la visibilité nommée FRIDA.

Pour chaque question dans ce compte rendu, j'ai ajouté le code de la réponse qui correspond aux questions. Et dans certaines questions j'ajoute des résultats graphiques nécessaires. De plus le code de Matlab est sous forme de script structuré par des blocs et pour chaque de question il en suit le code de la réponse. J'ai décomposé le code en deux parties vues que les deux parties de l'énoncé sont indépendants. Et ceci permet de réduire le nombre des lignes de code de chaque partie dans un seul fichier.

2 Exploration des données fournies

Question 1

On lit et affiche les 5 images sans brouillard de nom "00?.bmp" avec '?' varie entre 1 et 5 et les 5 images avec brouillard de nom "U00?.bmp". L'affichage est de mettre une image sans brouillard à côté du même image avec brouillard.

```
for num=1:5
    image_name=strcat('00',num2str(num),'.bmp');
    image = imread(image_name);
    image_avec_brouillard_name=strcat('U00',num2str(num),'.bmp');
    image_avec_brouillard = imread(image_avec_brouillard_name);
    deux_images = [image image_avec_brouillard];
    figure('Name',strcat('couple image',num2str(num)));
    imshow(deux_images);
end
```

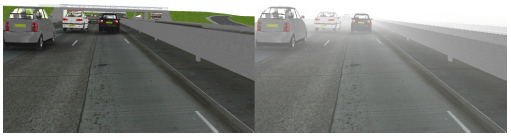
On affiche dans la figure 1 le résultat de ce code.

Question 2

On utilise la fonction **load** de Matlab pour lire le fichier BDD_vehicule.mat fournie dans l'énoncé. Ce fichier contient 3 variables 'BDD_vehicule','BDD_vehiculesmall','Search_Pattern'.

```
load('BDD_vehicule.mat','BDD_vehicule','BDD_vehiculesmall','
    Search_Pattern');
```

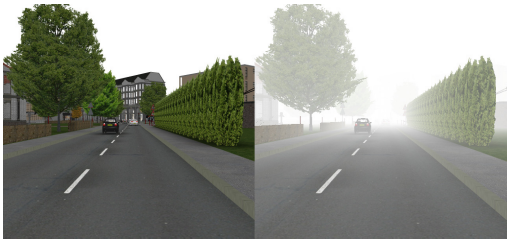
Cette ligne de code stocke les 3 variables dans le Workspace de Matlab. Les deux premières représentent la base d'apprentissage : les imagettes de références qu'on va comparer avec.
Et le troisième **Search_Pattern** contient les zones d'intérêt pour différentes échelles et des valeurs de seuils employé pour la détection.



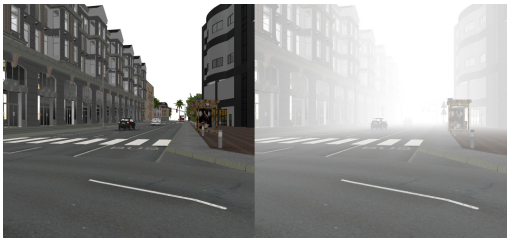
(a) image 001.bmp et U001.bmp

(b) image 002.bmp et U002.bmp

(c) image 003.bmp et U003.bmp



(d) image 004.bmp et U004.bmp



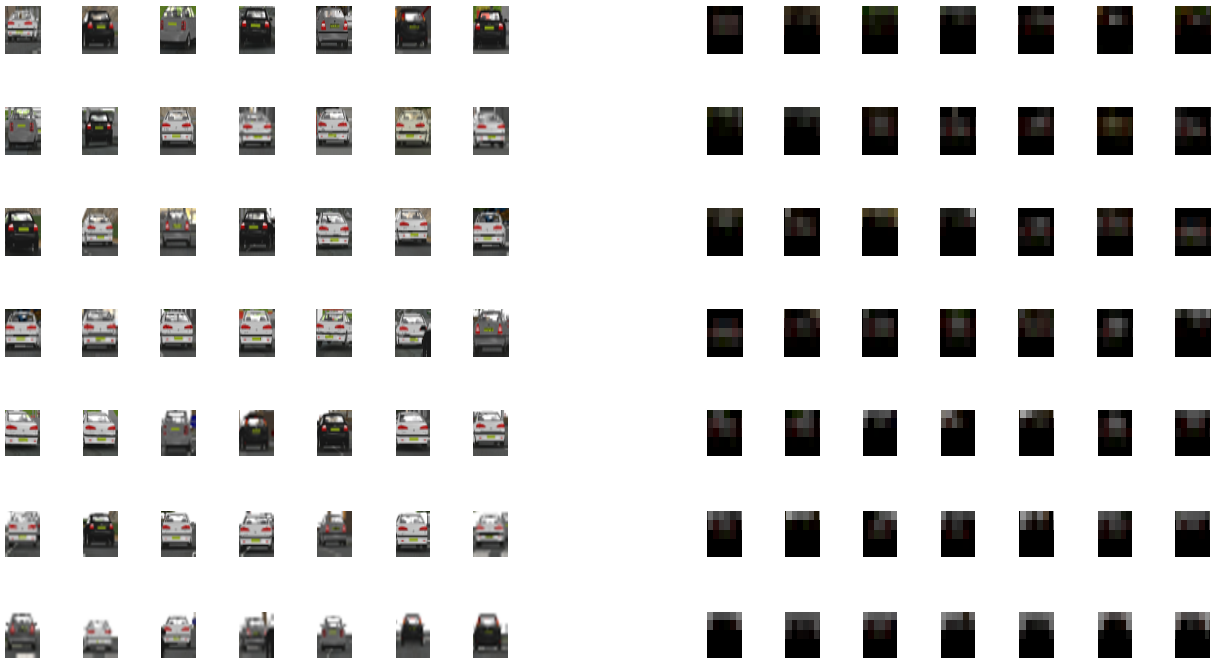
(e) image 005.bmp et U005.bmp

Figure 1: Images sans et avec brouillard

Question 3

On affiche les 7x7 premières imagettes des deux bases d'apprentissage **BDD_vehicule** et **BDD_vehiculesmall** dans la figure 2.

```
figure('Name','Vehicule');
k=0;
for scale=1:7
    for j=1:7
        k=k+1;
        subplot(7,7,k);
        x=BDD_vehicule{k,1}; %il suffit de changer en BDD_vehiculesmall
           pour faire de meme le deuxieme base d'aprentissage
        imshow(x);
        hold on;
    end
end
```



(a) BDD_vehicule (b) BDD_vehiculesmall

Figure 2: Bases d'apprentissage

3 Détection à plusieurs échelles dans une image

Pour réaliser la détection sur une image, des imagettes sont extraites de l'image et chaque imagette extraite va être comparée à celles contenues dans la base d'apprentissage. La comparaison entre deux imagettes se fera simplement par le calcul de la distance Euclidienne. Afin de pouvoir les comparer à celles de la base d'apprentissage réduite **BDD_vehiculesmall**, elles seront donc centrées en intensité et réduites à la taille 5x6x3. La base d'apprentissage contenant 51 imagettes de référence, le score de dissemblance entre une imagette test et la base d'apprentissage sera calculé comme la moyenne des 3 distances les plus petites à ces 51 imagettes de référence. Ce score de dissemblance sera donc d'autant plus petit que l'imagette considérée est ressemblante avec quelques imagettes de la base d'apprentissage.

Question 0

Avant d'aborder la question 1, j'ai préféré de bien comprendre le fonctionnement de la fonction **rectangle** de Matlab ainsi que la variable ROI.
La fonction **rectangle** de Matlab n'a pas le mémé "repère" (son origine est en bas à gauche) que celle de l'image (son origine est en haut à gauche) d'après la description de cette fonction dans Matlab. Cependant il faut préciser que ce repère se change et devient la même que celle de l'image lorsqu'on trace le rectangle sur une image ce qui n'est pas préciser. Pour mieux comprendre ce détail on peut se référer à l'image 3.

```
image = imread('001.bmp');
figure('Name','rectangle et image');
subplot(121);
```

```
imshow(image);
rectangle('Position',[1 1 50 200],'EdgeColor','r');
subplot(122);
rectangle('Position',[1 1 50 200],'EdgeColor','r'); %on trace le meme
    rectangle mais pas sur l'image
axis([1 size(image,2) 1 size(image,1)]);
```

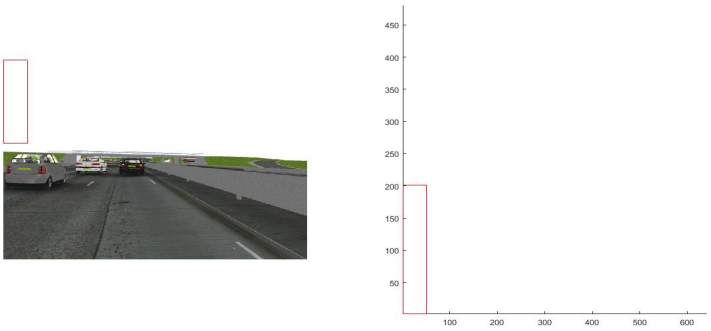


Figure 3: Changement du repère de la fonction **rectangle**

Autre souci est de comprendre comment utiliser ROI avec indexage. Comme montre la figure 4 il faut "inverser" les indexages dans l'image. On remarque après inversion d'indexage le rectangle entoure la région d'intérêt qu'on a choisi dont on connaît ses frontières.

```
image = imread('001.bmp');
figure('Name','ROI');
subplot(121);
image(ROI(1):ROI(1)+ROI(3),ROI(2):ROI(2)+ROI(4),:)=0.5;
imshow(image);
rectangle('Position',ROI,'EdgeColor','r');
%il faut inverser les ordres d'indexage
image = imread('001.bmp'); % reset image original
image(ROI(2):ROI(2)+ROI(4),ROI(1):ROI(1)+ROI(3),:)=0.5;
subplot(122);
imshow(image);
rectangle('Position',ROI,'EdgeColor','r');
```

Figure 4: Nécessité pour inverser l'ordre d'indexage usuelle dans l'image

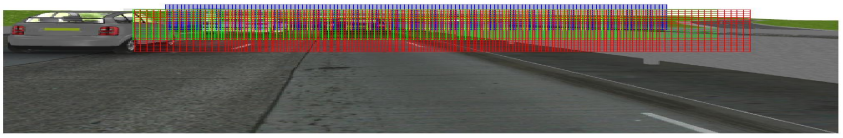
Question 1

On va parcourir toutes les images. Pour cela on utilise 2 boucles. La première parcoure le nombre des images et la deuxième les deux types d'images sans ou avec brouillard. On utilise indice (b = 1) pour image sans brouillard et indice (b = 2) pour image avec brouillard. On utilise les données du variable Search_Pattern afin de parcourir la zone d'intérêt ROI pour les différentes échelles avec les pas StepX et StepY.

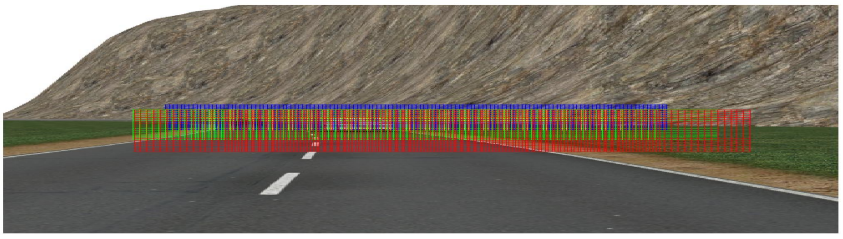
```
for num=1:5
    for b=1:2
        %on utilise indice (1) pour image sans brouillard
        %et indice(2) pour image avec brouillard
        if b==1
            image_name=strcat('00',num2str(num),'.bmp');
        else
            image_name=strcat('U00',num2str(num),'.bmp');
        end
        image = imread(image_name);
        image = double(image)./255.0;
        fig = figure('Name',image_name);
        imshow(image);

        for i=1:length(scale_list)
            scale = scale_list(i);
            Spattern = Search_Pattern(i);
            for col=Spattern.ROI(1):Spattern.StepX:Spattern.ROI(1)+
               Spattern.ROI(3)
                for row=Spattern.ROI(2):Spattern.StepY:Spattern.ROI(2)+
                   Spattern.ROI(4)
                    rectangle('Position',[col row 6*scale 5*scale],'
                       EdgeColor',Spattern.color);
                end
            end
        end
    end
end
```

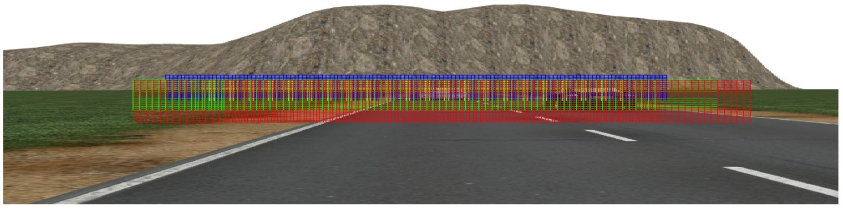
On affiche le parcours demandé dans la figure 5, et on se limite seulement aux images qui sont sans brouillard pour alléger le rapport de ces images. Cependant mon code fait ce parcours pour tous les 5 images sans et avec brouillard.



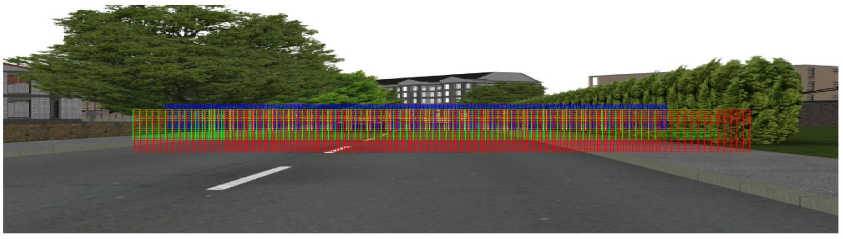
(a) Parcours de la zone ROI dans l'image '001.bmp'



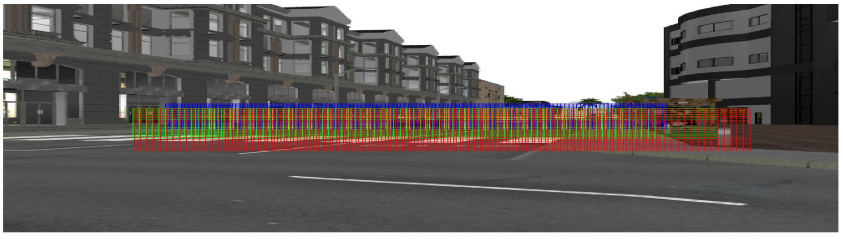
(b) Parcours de la zone ROI dans l'image '002.bmp'



(c) Parcours de la zone ROI dans l'image '003.bmp'



(d) Parcours de la zone ROI dans l'image '004.bmp'



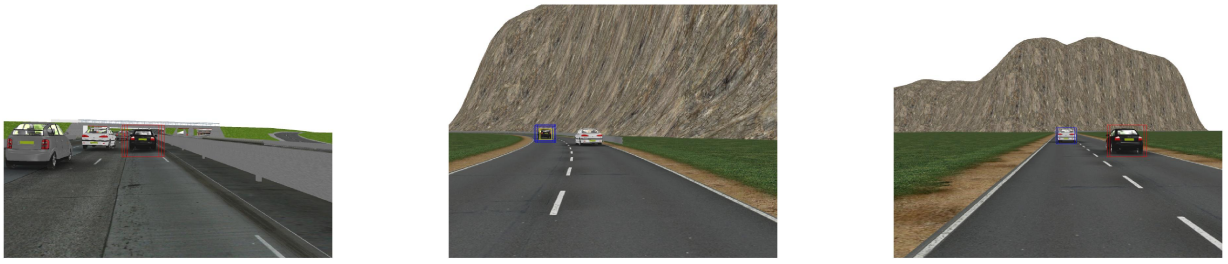
(e) Parcours de la zone ROI dans l'image '005.bmp'

Figure 5: Parcours de la zone ROI dans les images

Question 2

Comme dans cette question, il n'est pas précisé quel est l'image test sur laquelle on va travailler, j'ai choisi les trois premières images sans brouillard. En suivant la démarche proposée par l'énoncé, on

obtient les détections données dans la figure 6 avec le code suivant (voir le code Matlab qui contient des commentaires explicatif)



(a) image 001.bmp (b) image 002.bmp (c) image 003.bmp

Figure 6: Détection sur les 3 premières images sans brouillard

```
scale_list=[4 6 8 12];
image_name = '001.bmp'; %puis '002.bmp' ensuite '003.bmp'
image = imread(image_name);
image = double(image)./255.0;
fig2 = figure('Name',strcat('detection ',image_name));
imshow(image);
for i=1:length(scale_list)
    scale = scale_list(i);
    Spattern = Search_Pattern(i);
    for col=Spattern.ROI(1):Spattern.StepX:Spattern.ROI(1)+Spattern.ROI
        (3)
        for row=Spattern.ROI(2):Spattern.StepY:Spattern.ROI(2)+Spattern.
            ROI(4)
            imagette = image(row:row+5*scale,col:col+6*scale,:);
            imagette = imagette - mean(imagette(:));
            resized_imagette = zeros(5,6,3);
            %on diminue  la dimension de chaque couleur
            for ind_couleur=1:3
                resized_imagette(:,:,ind_couleur) = imresize(imagette
                    (:,:,ind_couleur),[5 6]);
            end
            %calculer les distances
            score=[];
            for j=1:51
                imagette_ref=BDD_vehiculesmall{j,1};
                score=[score norm(resized_imagette(:)-imagette_ref(:),2)
                    ];
            end
            score = sort(score);
            dissamblance_score = sum(score(1:3))/3;
            if dissamblance_score < Spattern.Thresh
                rectangle('Position',[col row 6*scale 5*scale],
                                'EdgeColor',Spattern.color);

            end
```

```
        end
    end
end
```

Question 3

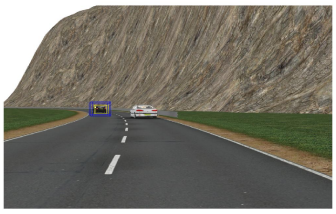
Ici on fait la détection seulement sur les images avec brouillard vue leurs structure de nom on peut employer une boucle pour les parcourir.

```
scale_list=[4 6 8 12];
for num=1:5
    image_name=strcat('U00',num2str(num),'.bmp');
    image = imread(image_name);
    image = double(image)./255.0;
    figd = figure('Name',strcat('detection ',image_name));
    imshow(image);
    for i=1:length(scale_list)
        scale = scale_list(i);
        Spattern = Search_Pattern(i);
        for col=Spattern.ROI(1):Spattern.StepX:Spattern.ROI(1)+Spattern.
            ROI(3)
            for row=Spattern.ROI(2):Spattern.StepY:Spattern.ROI(2)+
                Spattern.ROI(4)
                imagette = image(row:row+5*scale,col:col+6*scale,:);
                imagette = imagette - mean(imagette(:));
                resized_imagette = zeros(5,6,3); %initialisation
                for ind_couleur=1:3
                    resized_imagette(:,:,ind_couleur) = imresize(imagette
                        (:,:,ind_couleur),[5 6]);
                end
                score=[];
                for j=1:51
                    imagette_ref=BDD_vehiculesmall{j,1};
                    score=[score norm(resized_imagette(:)-imagette_ref(:)
                        ,2)];
                end
                score = sort(score);
                dissamblance_score = sum(score(1:3))/3;
                if dissamblance_score < Spattern.Thresh
                    rectangle('Position',[col row 6*scale 5*scale],...
                     'EdgeColor',Spattern.color);

                end
            end
        end
    end
end
```

Comme le montre la détection des images sans brouillard dans la figure 7, il n'y a pas de fausses alarmes alors que on remarque qu'il y a plusieurs non-détection (image 1 et 2) voire rien de détection (image 4 et 5).

(a) image 001.bmp



(b) image 002.bmp

(c) image 003.bmp

(d) image 004.bmp

(e) image 005.bmp

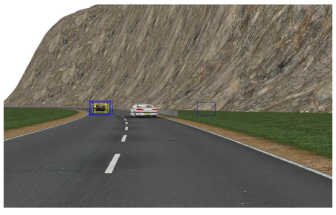
Figure 7: Détection sur les images sans brouillard

Question 4

On ajoute un facteur qui multiplie le dissemblance_score puisque on a obtenu des images qui ne détecte rien alors qu'il existe des véhicules. Il doit être inférieur à 1 pour ajouter le nombre des détections. Le code précédent se rectifie en ajoutant le code ci-dessous. Et on remarque que la détection s'améliore et on peut détecter les voitures de l'image 5 (voir figure 8).
Il est impossible d'avoir un facteur optimal sans fausse alarme et sans non détection puisque les plus qu'on diminue cette valeur pour détecter plus de voiture dans une image plus qu'on va avoir des fausses alarmes dans d'autres.

```
facteur = 0.93;
if facteur*dissamblance_score < Spattern.Thresh
```

(a) image 001.bmp



(b) image 002.bmp

(c) image 003.bmp

(d) image 004.bmp

(e) image 005.bmp

Figure 8: Détection sur les images sans brouillard avec facteur

Question 5

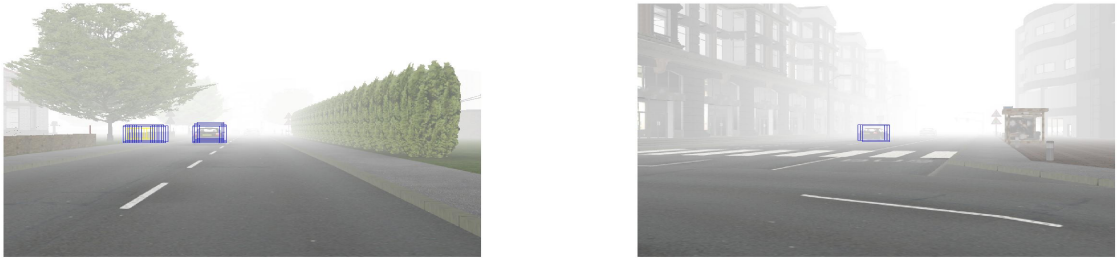
Pour traiter les images avec brouillard il suffit d'ajouter la lettre 'U' dans la variable **image_name** dans le code du question 3. On remarque dans la figure 9 qui n'emploie pas le facteur multiplicatif qu'on ne détecte rien

(a) image U001.bmp (b) image U002.bmp (c) image U003.bmp

Figure 9: Détection sur des images avec brouillard sans facteur multiplicateur

Ceci s'améliore un peu en ajoutant un facteur multiplicateur = 0.95 (voir figure 10) mais ceci reste limité et on peut avoir plusieurs fausses alarmes. En fait vus à cause du brouillard le véhicule est parfois caché et ceci complique leurs recherches. D'autre part notre base de données ne tient pas compte de l'effet de brouillard sur les arrières des véhicules. La solution est de restaurer les images comme on a fait pour un autre TP et puis on lance le programme de détection ça peut donner des résultats comparables au précédent.

(a) image U001.bmp (b) image U002.bmp (c) image U003.bmp



(d) image U004.bmp (e) image U005.bmp

Figure 10: Détection images sans brouillard avec facteur multiplicateur