ENSTA PARIS

MIV308 : RECONNAISSANCE DE FORMES

TP 1 : Extraction des marquages et détection des obstacles avec la v-disparité

Réalisé par :
Amin HENTETI

Enseignants :
M. Jean-Philippe Tarel
M. Taha RIDENE
M. IENG Sio-Song

Année universitaire: 2019/2020

1 Introduction

Dans la première partie de ce TP, on programme la méthode d'extraction des marquages MLT (Median Local Threshold). Puis on évalue cette extraction en utilisant la courbe ROC (Receiver Operating Characteristic). Dans la deuxième partie on réalise la détection des obstacles en utilisant la méthode v-disparité en utilisant un pair d'images stéréoscopiques. Cette nouvelle méthode consiste à utiliser **l'extracteur MLT** sur l'image de gauche et de droite de la paire stéréoscopique, puis **apparier** ces extractions entre les deux images pour en déduire l'histogramme **v-disparité**. À partir de ce dernier, le profil de **variation verticale de la disparité** de la route est estimé afin, ensuite, de détecter les obstacles.

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 vu qu'il y a découplage entre les partir à partir de la partie 6 Calcul des disparités. Et ceci permet de faciliter la lisibilité du code.

2 Lecture et affichage d'une image

Question 1

On stocke dans la variable "ima" l'image "01D.png" que l'on dispose.

```
1 ima = imread('01D.png');
```

Question 2

Le format de la variable **ima** est une matrice de dimension 258*330*3. Les pixels sont codés en uint8 donc ils sont de type entier. On affiche l'image dans une fenêtre nommé 'ima' (voir figure 1). L'image affiché est la même que l'originale.

```
1 figure('Name','ima');
2 imshow(ima);
```

Figure 1: imshow(ima)

Question 3

La variable **imad** contient la même image en format double. Cette conversion donne une double précision aux pixels de l'image ima mais la valeur reste la meme. L'affichage de l'image imad est donné dans la figure 2-(a). Il s'agit d'une image blanche. En fait l'intervalle d'affichage par défaut pour la fonction **imshow** pour une image de type double est [0,1] alors que l'image imad contient les mêmes valeurs que ima. Alors la quasi-totalité des pixels ont des valeurs supérieures à 1 qui représente le couleur blanc.

```
imad = double(ima);
figure('Name','imad');
imshow(imad);
```

(a) imad (b) imad/255

Figure 2: Affichage de l'image imad et sa normalisé

Pour remédier ce problème, on normalise la variable **imad** cad on le divise par $255 = 2^8 - 1$ (c'est le maximum des valeurs de pixels puisqu'ils sont codées en uint8 d'après la question précédent). On affiche la nouvelle image dans la figure 2(b).

```
imad = imad/255;
figure('Name','imad/255');
imshow(imad);
```

Question 4

L'image de départ est en couleur. Alors **ima** contient les 3 niveaux de couleurs RGB (Rouge, Vert, Bleu). On stocke dans **imagray** et **imadgray** la conversion en niveau de gris de l'image **ima** et **imad** respectivement.

```
imagray = rgb2gray(ima);
imadgray = rgb2gray(imad);
```

On fait la différence de ces 2 nouveaux images. Pour cela il faut que les deux images soient du même type et de plus multiplier **imadgray** par 255 puisque ses pixels $\in [0, 1]$. Je fais la valeur absolue pour éviter les valeurs négatives. On affiche dans la figure 3 le résultat de la différence. On remarque un nuage des points blancs qui est dû aux divisions par 255 pour la variable **imad**

```
ima_diff = abs(double(imagray) - 255*imadgray);
diff_fig = figure('Name','ima_diff');
imshow(ima_diff);
```

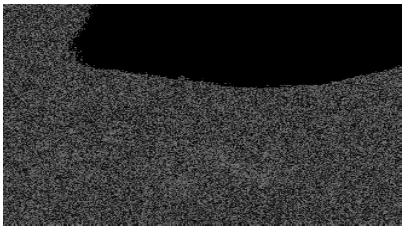


Figure 3: Résultat de la différence des images imagray et imadgray

Question 5

J'ai stocké la figure de l'affichage de la différence dans la variable **diff_fig**. On peut alors sauvegarder l'image résultant de la comparaison avec le format png et sous un nom soit "ima_diff.png"

```
saveas(diff_fig,'ima_diff.png');
```

3 Propriétés d'une image

On s'interesse à l'image **imadgray**

Question 1

La valeur du pixel (232,212) = 0.3288 La dimension de l'image = 258*330 On stocke le nombre de ligne nl=258 , nombre de colonne nc=300, et le nombre de bandes/canaux dans la variable nb=1.

```
pixel = imadgray(232,212)
size(imadgray)
[nl,nc,nb] = size(imadgray);
```

Question 2

En utilisant la boucle for, on stocke la valeur maximale de chaque ligne de l'image **imadgray** dans le vecteur "maxlig".

```
maxlig = ones(nl,1);
for i=1:nl
    maxlig(i) = max(imadgray(i,:));
end
```

En s'appuyant sur la documentation de la fonction **min** de Matlab on peut calculer en une seule ligne le minimum et la moyenne de chaque ligne de l'image **imadgray**

```
minlig = min(imadgray,[],2);
moylig = mean(imadgray,2);
```

On fait le tracé de ces trois vecteurs dans la figure 4

```
figure('Name','max min moy des lignes');
plot(maxlig);hold on;
plot(minlig);hold on;
plot(moylig);
legend('maxlig','minlig','moylig');
```

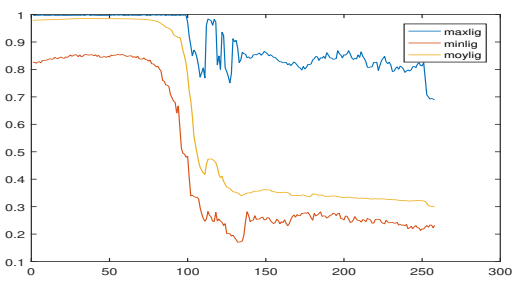


Figure 4: tracé des trois vecteurs maxlig, minlig et moylig

Question 3

Pour l'image **imadgray** : moyenne = 0.5961, maximum = 0.9984, minimum = 0.1706 et somme = 5.0748e+04

```
moyima = mean(imadgray(:));
maxima = max(imadgray(:));
minima = min(imadgray(:));
sumima = sum(imadgray(:));
```

4 Extraction des marquages

Question 1

On filtre l'image **imadgray** avec un filtre moyen de taille [3 55] et on affiche le résultat dans la figure 5.

```
avg_filter = fspecial('average',[3 55]);
imad_filtred_avg = imfilter(imadgray,avg_filter);
figure('Name','imad_filtred_avg');
imshow(imad_filtred_avg);
```

(a) filtre moyenne 3*55 (b) filtre médian 3*60

Figure 5: Filtrage de **imadgray** avec deux filtres

Question 2

On filtre l'image **imadgray** avec un filtre médian de taille [3 60]. Après plusieurs essaies pour différentes dimensions de filtre médiane, la meilleure était [3 60].

```
imad_filtred_med = medfilt2(imadgray,[3 60]);
figure('Name','imad_filtred_med');
imshow(imad_filtred_med);
```

La comparaison des deux filtrages donnés dans 5 montre que le filtrage par un filtre médian réussi à supprimer les marquages contrairement au filtre moyenne où le marquage persiste et figure toujours pour n'importe taille de filtre.

Question 3

On soustrait l'image filtré par le filtre médian à l'image **imadgray** et on affiche le résultat dans 12

```
extraction_by_med = abs(imad_filtred_med-imadgray);
figure('Name','extraction_by_med');
imshow(extraction_by_med);
```

(a) extraction_by_med (b) imask

Figure 6: Résultat extraction des marquages par filtre médian

Question 4

On stocke dans **imask** le binarisé de l'image **extraction_by_med**. On fixe alors un seuil $s_{marquage}$ et si la valeur du pixel **extraction_by_med**(i,j) $> s_{marquage} \longrightarrow$ **imask**(i,j) = 1, sinon **imask**(i,j) = 0 avec (i,j) représente la position du pixel. Ceci peut être implémenter en Matlab en une simple ligne en initialisant **imask** à 0. L'affichage de l'image **imask** est donné dans la figure 12. Malheureusement pas seulement mais en plus une partie de la voiture qui est en face puisqu'elle contient des niveaux de blancs proche que celle des marquages.

```
s_marquage = .22;
imask = 0*imad_filtred_med;
imask( extraction_by_med > s_marquage) = 1;
figure('Name','imask');
imshow(imask);
```

Question 5

On considère une ouverture morphologique rectangulaire de dimension 11*5. L'affichage des étapes d'extraction est donné dans la figure **??**

```
rectangle = strel('rectangle',[5 23]);
imad_morph_rect = imopen(imadgray,rectangle);
extraction_by_morph_rect = abs(imad_morph_rect-imadgray);
s_marquage = .25;
imask_morph_rect = 0*extraction_by_morph_rect;
imask_morph_rect( extraction_by_morph_rect > s_marquage) = 1;
```

5 Évaluation de l'extraction

Question 1

On peut s'appuyer sur l'extraction que je l'appelle **idéal** puisqu'elle réussit à extraire exactement les marquages sans fautes, pour évaluer les extractions avec la courbe ROC qui emploie les variables **tp, fn, fp** et **tn**

- tp : nombre de pixels où réellement on a marquage donc dans l'extraction idéal a un valeur égal à 1 **ET** on a reussi à l'extraire par notre méthode cad = 1. Il s'agit de correctement extraits.

Les autres significations des variables est donné dans les commentaires du code. Par ces significations on peut les calculer facilement grâce à Matlab.

```
ideal = imread('01DR.png');
tp = sum(sum(ideal & imask_morph_rect));
fn = sum(sum(ideal & not(imask_morph_rect)));
```

```
fp = sum(sum(not(ideal) & imask_morph_rect));
tn = sum(sum(not(ideal) & not(imask_morph_rect)));
tpr = tp/(tp+fn);
fpr = fp/(fp+tn);
```

Pour mon méthode d'extraction (avec $s_{marquage} = 0.22$), on a :
tp = 1843 fn = 415, fp = 614, tn = 82268, tpr = 0.8162, et fpr = 0.0074

Question 2 : ROC

Afin d'éviter la répétition des mêmes lignes de code, j'ai fait une fonction qui permet de tracer un courbe ROC pour une extraction par une méthode entrée soit MLT ou morphologique. Dans la figure 7 on affiche la courbe associée à la méthode MLT

```
function [fpr,tpr]=ROC(extraction)
ideal = imread('01DR.png');
tpr=[];fpr=[];
for s_marquage = 0:0.01:1;
    imask = 0*extraction;
    imask( extraction > s_marquage) = 1;
    tp = sum(sum(ideal & imask));
    fn = sum(sum(ideal & not(imask)));
    fp = sum(sum(not(ideal) & imask));
    tn = sum(sum(not(ideal) & not(imask)));
    tpr = [tpr tp/(tp+fn)];
    fpr = [fpr fp/(fp+tn)];
end
plot(fpr,tpr);
end
```

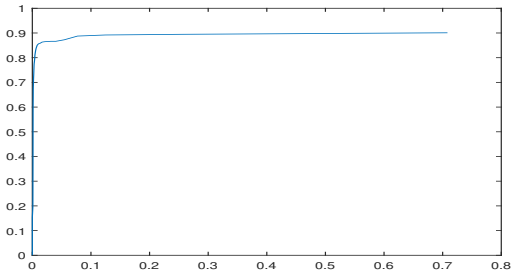


Figure 7: la courbe ROC pour MLT

Question 3 : Comparaison

Je fais deux autres méthodes d'extraction par deux ouverture morphologiques le premier est un rectangle de dimension [5 23], le deuxième est un octogone de rayon 3*6.
D'après la comparaison donnée dans la figure 8 on remarque ces deux ouvertures ont presque identiquement le même résultat d'autres part l'extraction par MLT est bien meilleur. On peut déduire que la méthode MLT est une très bonne méthode pour la détection des marquages.

```
octa = strel('octagon',3*6);
imad_morph_octa = imopen(imadgray,octa);
extraction_by_morph_octa = abs(imad_morph_octa-imadgray);
```

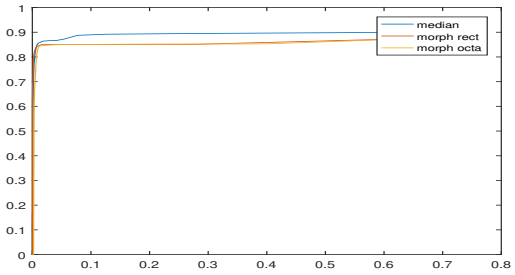


Figure 8: la courbe ROC pour 3 méthodes d'extraction

6 Calcul des disparités

Question 1

On affiche les deux images **'01G.png'** et **'01D.png'** côte à côte et on trace deux lignes qui s'intersecte en premier lieu loin du véhicule en figure 9(a) : on remarque le même point réel n'a pas changer de position dans les deux images alors que dans figure 9(b) le point proche du véhicule est largement changé de position. Ce qui justifie que la disparité est inversement proportionnelle à la profondeur.

```
imaG=imread('01G.png'); imaD=imread('01D.png');
[nl,nc,nb]=size(imaG);
imaGD = [imaG imaD];
width = 3;
imaGD(110:110+width,:,:)=0; imaGD(:,9*20:9*20+width,:)=0; imaGD(:,9*20+nc:9*20+nc+width,:)=0;
figure('Name','geometrie epipolaire 1');
imshow(imaGD)
imaGD = [imaG imaD]; %undo plotted lines
imaGD(11*20:11*20+width,:,:)=0; imaGD(:,6*20:6*20+width,:)=0; imaGD(:,6*20+nc:6*20+nc+width,:)=0;
figure('Name','geometrie epipolaire 2');
imshow(imaGD)
```

(a) pixels à l'horizon

(b) pixels prés

Figure 9: géométrie épipolaire

Question 2 : Centres

Au début on convertit vers le niveau de gris puis on applique MLT aux images gauches et droite. Ici on peut choisir des seuils de marquage plus petit qu'auparavant soit $s_{marquage}$]=0.17 puisque même si cela donne plus des pixels détectés mais on va les ignorer grâce à la fonction **bwmorph**.

```
imaGgray = rgb2gray(double(imaG)./255);
imaG_filtred_med = medfilt2(imaGgray,[3 60]);
extractionG_by_med = abs(imaG_filtred_med-imaGgray);
```

```
s_marquage = .17;
imaskG = 0*imaG_filtred_med;
imaskG( extractionG_by_med > s_marquage) = 1;
figure('Name','MLT G');
imshow(imaskG);
```

On affiche les résultats de leurs extractions dans la figure 11

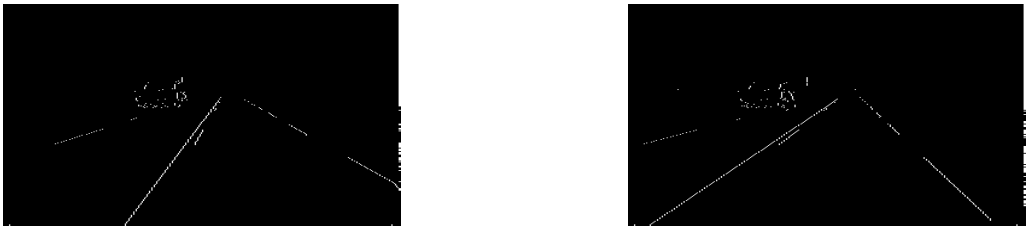
(a) image gauche (b) image droite

Figure 10: MLT appliqué aux images gauche et droite

On utilise la fonction **bwmorph** avec l'option 'shrink' pour extraire les centres horizontaux des images binaires avec 100 itérations maximum. Ces opérations se fait de parait à l'image droite que l'image gauche.

```
centreG = 0*imaskG;
for i= 1:nl
    centreG(i,:) = bwmorph(imaskG(i,:),'shrink',inf);
end
figure('Name','centre G');
imshow(centreG);
```

On obtient les centre dans la figure **??**



(a) image gauche (b) image droite

Figure 11: centres selon l'horizontale sur les images

Question 3+4 : Appariement et Sélection

On parcourt les centres extraits dans la question précèdent et on vérifie sa similarité dans un voisinage de pixels allant de 0 à 50 (en verticale parce que les pixels appairés doivent avoir la même ligne horizontale). Ce choix de voisinage est selon l'énoncé (question 4) pas possible d'avoir une disparité négative et avoir une disparité maximale soit 50 par exemple. Une fenêtre sera formée autour du pixel dans l'image gauche et on parcoure les pixels de l'image droite avec décalage de fenêtre sur la ligne du pixel du centre. Pour éviter les effet de bords on fait **padding** : ajout des valeurs nuls aux bords de l'image.

(a) image gauche (b) image droite

Figure 12: Centre selon horizentale

```
half_window_size = 8 ;
imaGgray_pad = im2double(padarray(imaGgray,[half_window_size,half_window_size], 'replicate'));
imaDgray_pad = im2double(padarray(imaDgray,[half_window_size,half_window_size], 'replicate'));
diparity_image = zeros(nl,nc);
max_disparite = 80;
tabD = []; tabG = [];
```

Pour le corps de l'algorithme d'apparaiment. On calcule la différence de somme au carré (distance eiclidienne) sur la totalité des deux fenêtres gauches et droite. Il s'agit de la corrélation entre les fenêtres ou encore des régions considéré de l'image gauche avec l'image droite. La disparité à laquelle on a la plus petite valeur distance euclidienne correspond à l'attribution en tant que position dans l'image droite apparié au pixel gauche. On peut se référer au code pour avoir des commentaires sur les lignes de code.

```
for i=1:nl
    tabG = find(centreG(i,:));
    tabD = find(centreD(i,:));
    for j=tabG
        regionG = imaGgray_pad(i:i+2*half_window_size, j:j+2*half_window_size);
        diparite_G = [1e6 0];
        for d = 0:max_disparite
            if j<d+1
                break
            else
                regionD = imaDgray_pad(i:i+2*half_window_size, j-d:j-d+2*half_window_size);
                correlation = sum(sum((regionG - regionD).^2));
                if correlation < diparite_G(1)
                    diparite_G = [correlation d];
                end
            end
        end
    end
    diparity_image(i,j)=diparite_G(2);
end}
```

Question 5

On a collecté les appariements dans la variable **diparity_image**. On peut alors construire la carte de disparité en cherchant les points qui n'ont pas de disparité nul. On trace cette carte de disparité dans la figure 13

```
figure('Name','carte de disparite');
[vG uG] = ind2sub([nl nc],find(diparity_image));
disp = zeros(size(vG));
for h = 1:length(vG)
    disp(h) = diparity_image(vG(h),uG(h));
end
scatter(vG,uG,disp);
```

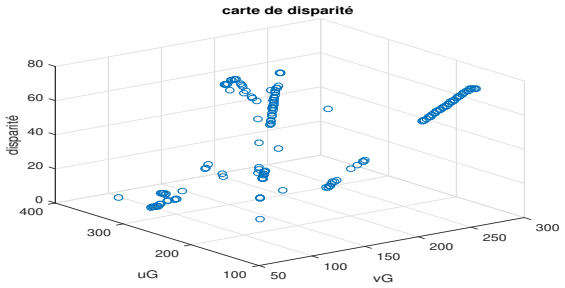


Figure 13: carte de disparité

7 Détection des obstacles

Question 1

On peut déclarer un variable $\mathbf{vert}_{profilquicalculelenombred'appariementsaucoursdel'appariement}$.

```
figure('Name','histogramme 2D');\\
scatter3(vG,disp,vert_profil);\\
xlabel('uG');ylabel('vG');zlabel('disparit[U+FFFD]');title('histogramme 2D');\\
```

Question 2

Question 3

```

```

Question 4

```

```