MASTER ÉLECTRONIQUE, ÉNERGIE ÉLECTRIQUE,
AUTOMATIQUE
PARCOURS SYSTÈMES AUTOMATIQUES MOBILES

PERCEPTION ÉTENDU
TRAVAIL PRATIQUE

Détection des obstacles avec la v-disparité

Étudiants :
Otmane ATTOU

Enseignant :
J.P. TAREL

Table des matières

Table des figures

1 Introduction :

Le calibrage d'une paire de caméras sur un véhicule donnant une paires d'images stéréoscopiques permettent de détecter les obstacles en utilisant la méthode de v-disparité.
Dans ce TP, nous allons construire premièrement une carte de disparité à partir de deux images issus de deux caméras calibrés sur un véhicule pour en déduire l'histogramme 2D v-disparité. Ensuite, nous estimerons le profil de variation vertical de la disparité de la route afin de détecter les obstacles avec un simple histogramme 1D.

2 Calcul des disparités :

1. On affiche les images avec le code suivant.

```
I1D = imread('01D.png');
I1G = imread('01G.png');
figure('name','images de départ');
subplot(1,3,1); imshow(I1G); title('I1 gauche');
subplot(1,3,2); imshow(I1D); title('I1 droite');

J = stereoAnaglyph(I1D,I1G);
subplot(1,3,3); imshow(J); title('Images I1G et I1D superposés en anaglyphe');
```

Nous obtenons ainsi :

FIGURE 1 – Image1 gauche et image 1 droite superposés en anaglyphe

On remarque que les plans image des deux caméras sont parallèles à la droite joignant les centres de projection. Cette configuration est appelé "cas stéréo normal". Nous nous plaçons souvent en géométrie rectifié où les deux plans d'images sont confondus dans un même plan comme le montre la figure 1. A l'inverse d'une géométrie quelconque, un point 3D de coordonnées (u_g, u_d) dans l'image de droite et de coordonnées (u_g, v_g) dans l'image de gauche a la même ordonnée $v = v_g = v_d$. Ainsi, la disparité est exprimé par $u_g - _d$ qui est inversement proportionnelle à la profondeur P.

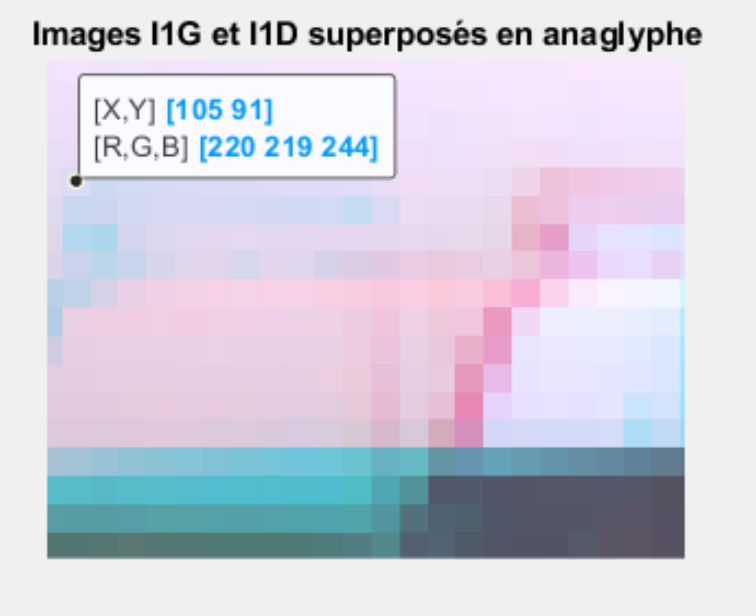

FIGURE 2 – cas stéréo normal

2. Nous travaillons avec les images en niveau gris, nous affichons les résultats avec les différentes fonctions de disparité *Disparity* , *DisparityBM* et *DisparitySGM* avec des intervalles des disparités différentes. Afin d'y arriver, nous avons créer une fonction *disparityFun* qui fait cela.

```
function disparityFun(I1D, I1G, range, n , m)
% I_d :  Image de droite en niveau gris
% I_G : Image de Gauche en niveau gris
% range
% n : nombre de ligne correspondant au subplot
% m : nombre de colonne correspondant au subplot
figure('NAME',['Avec la fonction disparity avec range',num2str(range(1,1)),',',num2s
for i = 1:size(range,1)
    disparity_Map = disparity(I1D,I1G,'DisparityRange', range(i,:));
    subplot(n,m,i); imshow(disparity_Map); title(['range [',num2str(range(i,1)),',',
end
% DisparityBM
figure('NAME',['Avec la fonction disparityBM avec range',num2str(range(1,1)),',',num
for i = 1:size(range,1)
    disparity_BM = disparityBM(I1D,I1G,'DisparityRange', range(i,:));
    subplot(n,m,i); imshow(disparity_BM); title(['range [',num2str(range(i,1)),',',n
end

% %DisparitySGM
figure('NAME',['Avec la fonction disparitySGM avec range',num2str(range(1,1)),',',nu
for i = 1:size(range,1)-1
    DSGM = disparitySGM(I1D,I1G, 'DisparityRange', range(i,:));%Difference between m
    subplot(n,m,i); imshow(DSGM); title(['range [',num2str(range(i,1)),',',num2str(r
end

end
```

Ainsi nous prenons des intervalles espacé par un nombre divisible par 16.

```
I1D = rgb2gray(I1D);
I1G = rgb2gray(I1G);
J = rgb2gray(J);
```

```
range1  = zeros(9,2); range2  = zeros(8,2); range3  = zeros(7,2); range4  = zeros(6,
range5  = zeros(5,2); range6  = zeros(4,2); range7  = zeros(3,2); range8  = zeros(2,

for i= 1: size(range1, 1)
   range1(i,2) = i*16;
   range1(:,1) = 0;
end
for i= 1: size(range2, 1)
   range2(i,2) = 16+i*16;
   range2(:,1) = 16;
end
for i= 1: size(range3, 1)
   range3(i,2) = 32+ i*16;
   range3(:,1) = 32;
end
for i= 1: size(range4, 1)
   range4(i,2) = 48+ i*16;
   range4(:,1) = 48;
end
for i= 1: size(range5, 1)
   range5(i,2) = 64+ i*16;
   range5(:,1) = 64;
end
for i= 1: size(range6, 1)
   range6(i,2) = 80+ i*16;
   range6(:,1) = 80;
end
for i= 1: size(range7, 1)
   range7(i,2) = 96+ i*16;
   range7(:,1) = 96;
end
for i= 1: size(range8, 1)
   range8(i,2) = 112+ i*16;
   range8(:,1) = 112;
end
disparityFun(I1D, I1G, range1, 3, 3)
disparityFun(I1D, I1G, range2, 2, 4)
disparityFun(I1D, I1G, range4, 2, 3)
disparityFun(I1D, I1G, range5, 1, 5)
disparityFun(I1D, I1G, range6, 2, 2)
disparityFun(I1D, I1G, range7, 1, 3)
disparityFun(I1D, I1G, range8, 1, 2)
```

Vous trouvez ci-dessous quelques résultats. Par la suite, nous choisissons de travailler tout d'abord avec la fonction *disparity* et l'intervalle $[0, 96]$. Puis, nous testerons d'autres intervalles et fonctions.

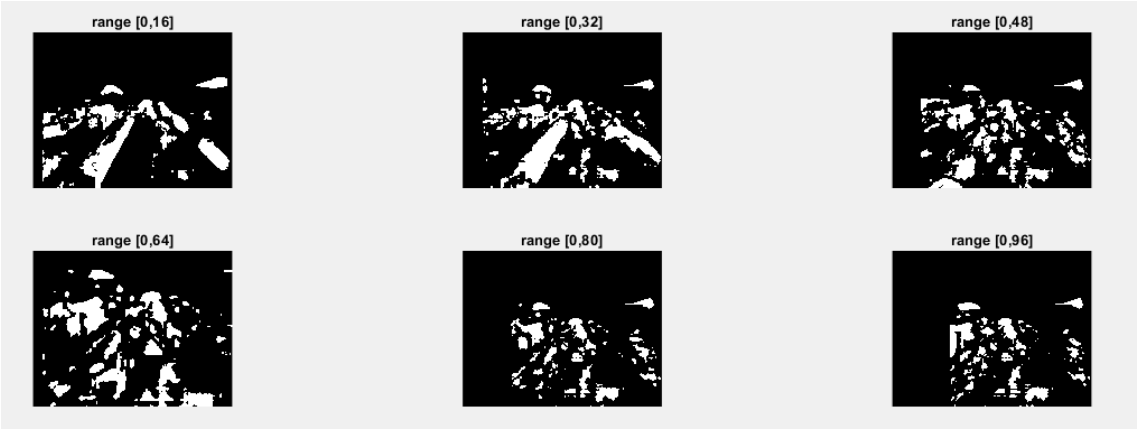

FIGURE 3 – Avec la fonction disparity

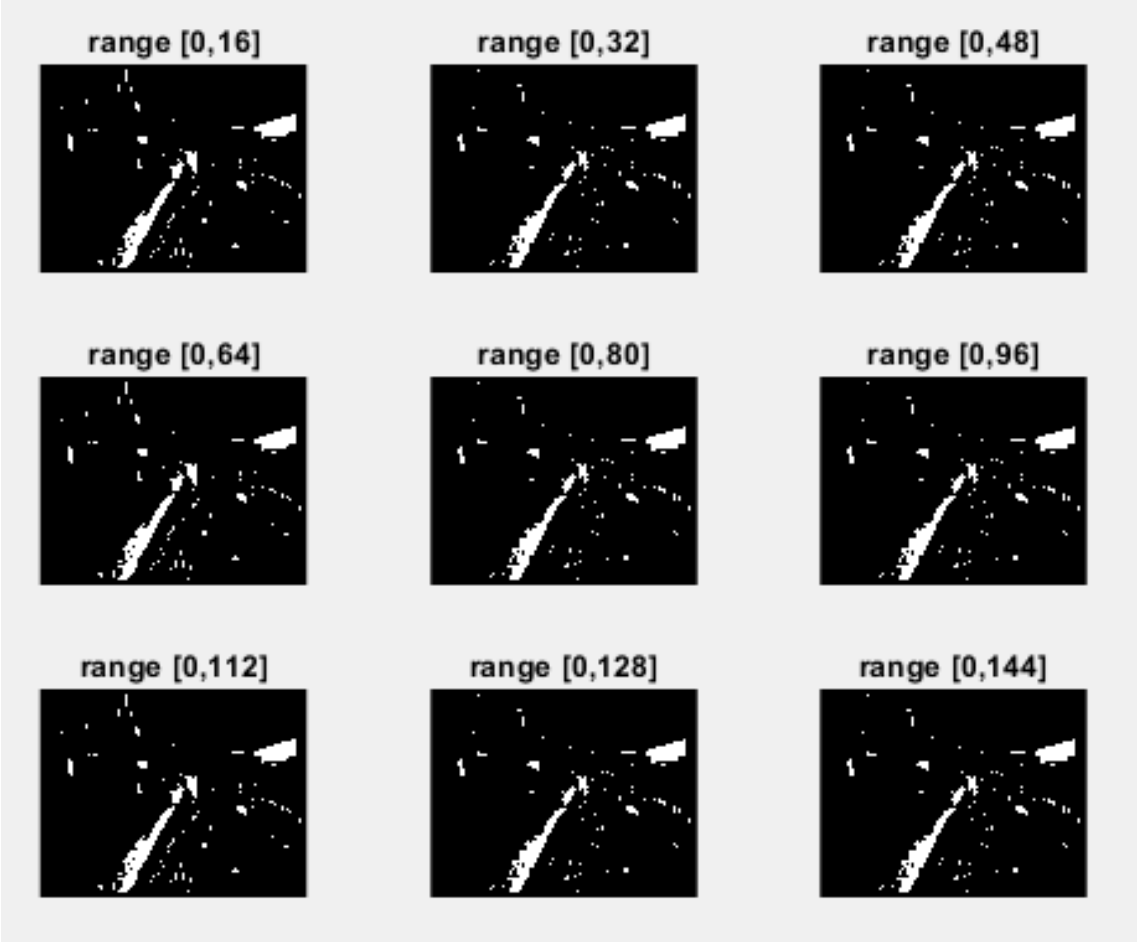

FIGURE 4 – Avec la fonction disparityBM

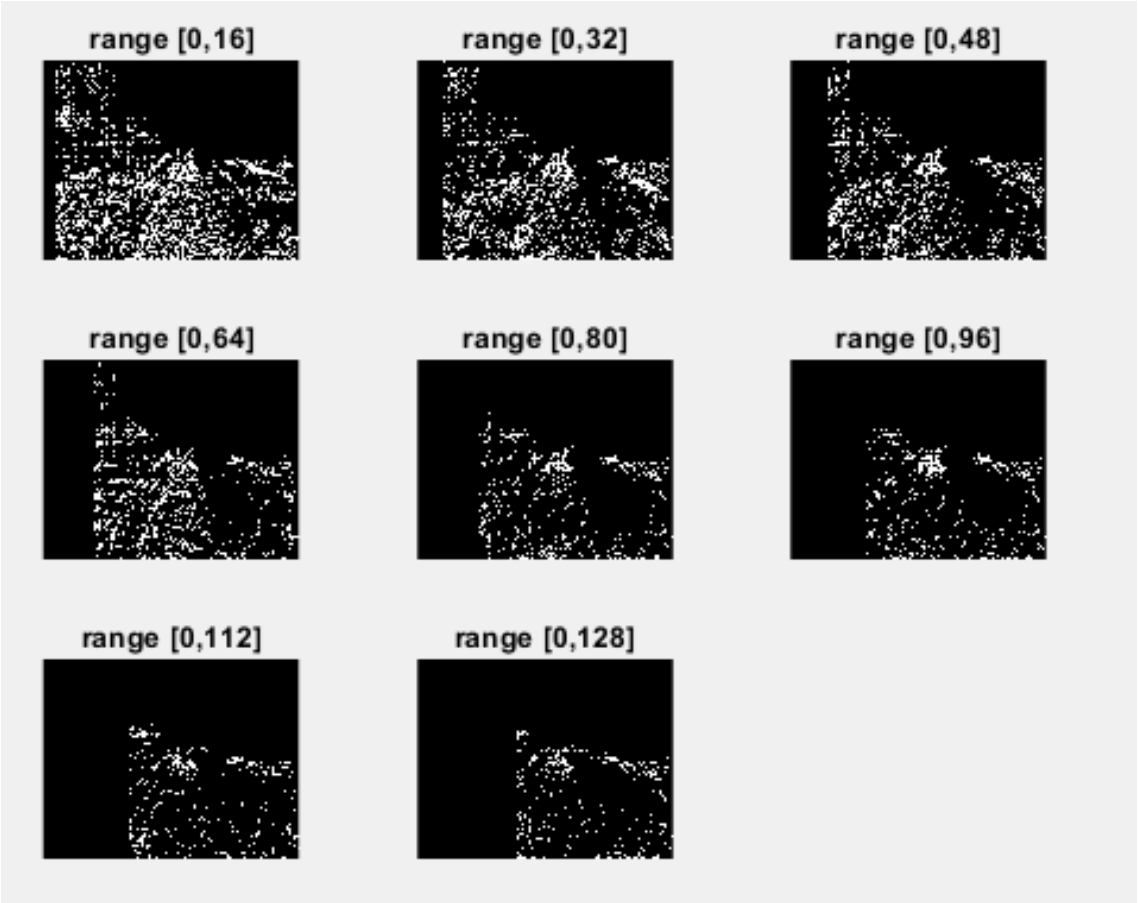

FIGURE 5 – Avec la fonction disparitySGM

Finalement, nous choisissons de travailler avec l'intervalle $[096]$ et avec la fonction *disparitySGM*. Le résultat est le suivant :

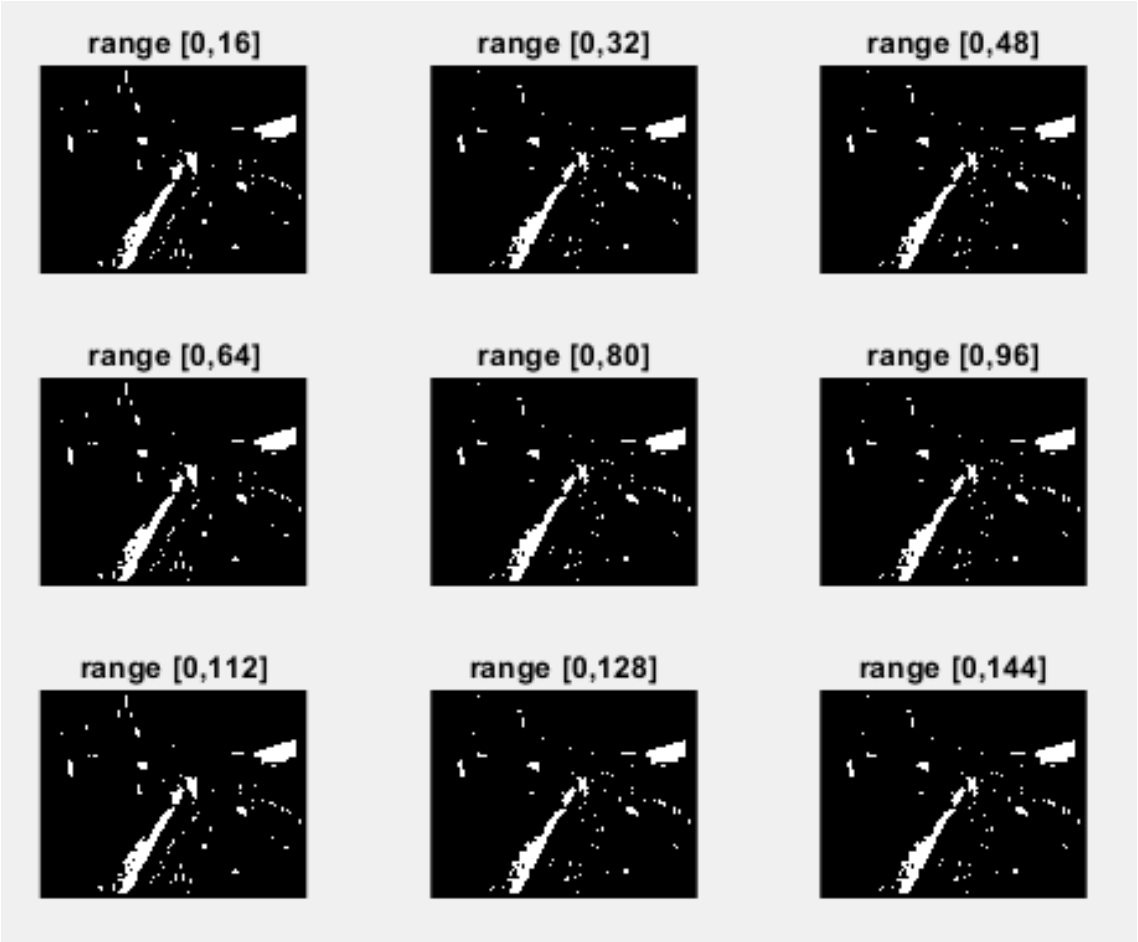

FIGURE 6 – Avec les différentes fonctions de disparity sur l'intervalle $[096]$

3 Détection des obstacles :

3.1 V-disparité :

Afin de détecter la route, nous construisons l'histogramme v-disparité. Les abscisses sont les couple $(v, disparite)$ et les ordonnées sont le nombre d'appariements de chaque couple.cet histogramme contient ainsi un profil vertical la route et un profil horizontal représentant respectivement la route et des obstacles. C'est avec cette cette méthode que nous détecterons la route et l'éloignement des obstacles. Afin de construire la carte de v-disparité, j'ai programmé une fonction qui prend en paramètre l'adresse de l'image gauche et droite et l'intervalle de disparité. Cette fonction est la suivante :

```
function v_disp = v_disparite(disparityMap,range,Ig)
    %Inputs :
    % disparityMap est la carte de disparité
    % range : intervalle de disparité
    % Ig : l'adresse de l'image de gauche de type chaine de caractere. Ex : '01G.png'
    %Outputs :
    % Carte v-disparité en gray
    % Carte v-disparité en color
    % Histogramme du nombre d'appariements pour chaque couple de valeurs
    % (v, disparité )

    %% Image de gauche
    Ig=rgb2gray(double(imread(Ig))/255);
```

```
    % creation matrice
    v_disp=zeros(size(Ig,1),size(disparityMap,1));
    %% construction v-disparity
    for i=1:size(disparityMap,1)
        for k=1:96
            for j=1:size(disparityMap,2)
                if round(disparityMap(i,j)+1)==k
                    v_disp(i,k)= v_disp(i,k)+1;
                end
            end
        end
    end
    % affichage des résultat
    figure;
    ax1=subplot(2,1,1);
    imshow([disparityMap v_disp],range)
    title('Carte v-disparité en gray')
    colormap(ax1,'gray')

    ax1=subplot(2,1,2);
    imshow([disparityMap v_disp],range)
    title('Carte v-disparité en color')
    colormap(ax1,'jet')

end
```

Nous appliquons cette fonction sur les image '01G.png' et '01D.png' sur l'intervalle [0 96], ainsi :

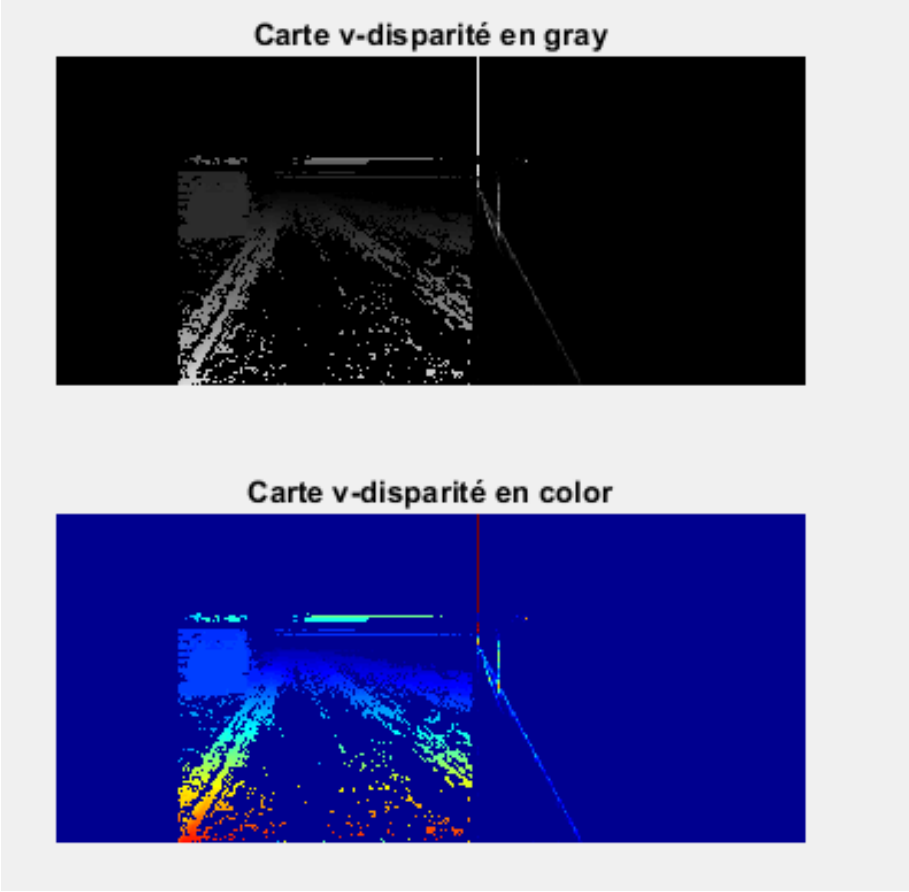

FIGURE 7 – La carte de v-disparité en niveaux de gris et en couleur et l'histogramme de nombre d'appariements 2D pour chaque couple de valeurs (v, disparité)

On remarque la construction d'un profil vertical représentant la route et d'un profil horizontale représentant l'obstacle.

3.2 Profil de la route :

Nous faisons tout d'abord une régression robuste d'une ligne sur le nuage de points de disparité en utilisant la fonction *robustfit*. La fonction suivante permet de faire cela :

```
function b = profilDeLaRoute(v_disp , svd )
    %Inputs
    % v_disp : est la v-disparité
    % svd : seuil
    %Outputs
    % b : le résultat de la fonction robustfit
    % Figure : La représentationde la ligne de la route sur l'histogramme

    prof_route = [];
    for i = 1:size(v_disp,1)
        for j = 1:size(v_disp,2)
            if v_disp(i,j) > svd
                prof_route = [prof_route;[i j]];
            end
        end
    end


    % fonction robustfit renvoie vecteur b
    b = robustfit(prof_route(:,2), prof_route(:,1));
    x = 0:size(v_disp,1);
    y = b(1) + b(2).*x;

    subplot(3,3,svd);
    imshow(v_disp);
    hold on
    % plotage de la ligne de la route en rouge
    plot(x,y,'r');
    title(sprintf('Detection de la route avec svd=%d',svd))
end
```

Ainsi, nous prenons plusieurs valeurs de seuil svd afin de remarquer son influence :

```
%% Q2 : Profil de la route
figure
for svd = 1 :9
    b = profilDeLaRoute(v_disp , svd );
end
b = profilDeLaRoute(v_disp , 1 );
```

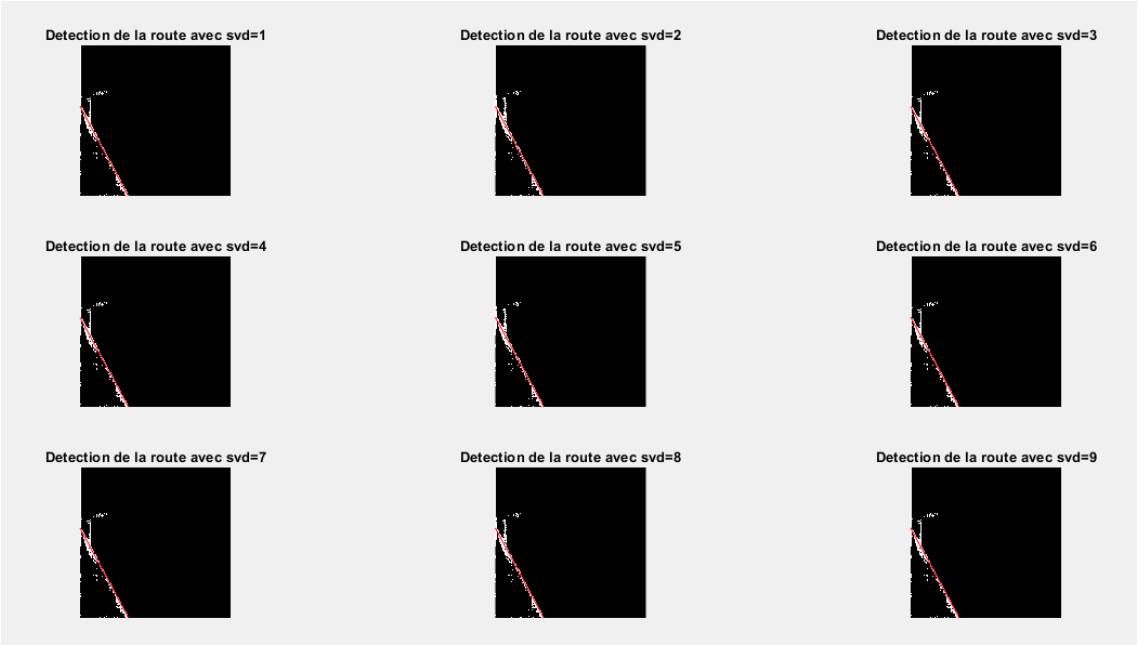

FIGURE 8 – Régression robuste d'une ligne sur le nuage de points $(v, disparite)$ pour des valeurs de seuil allant de 1 à 9

Pour moi, la variation de la valeur de seuil n'a pas une grande importance. Peut-être que cette dernière affectera les routes qui contient des virages ou ne sont pas plates.
On décide ainsi de prendre la valeur de $svd = 1$, le résultat est la suivante :

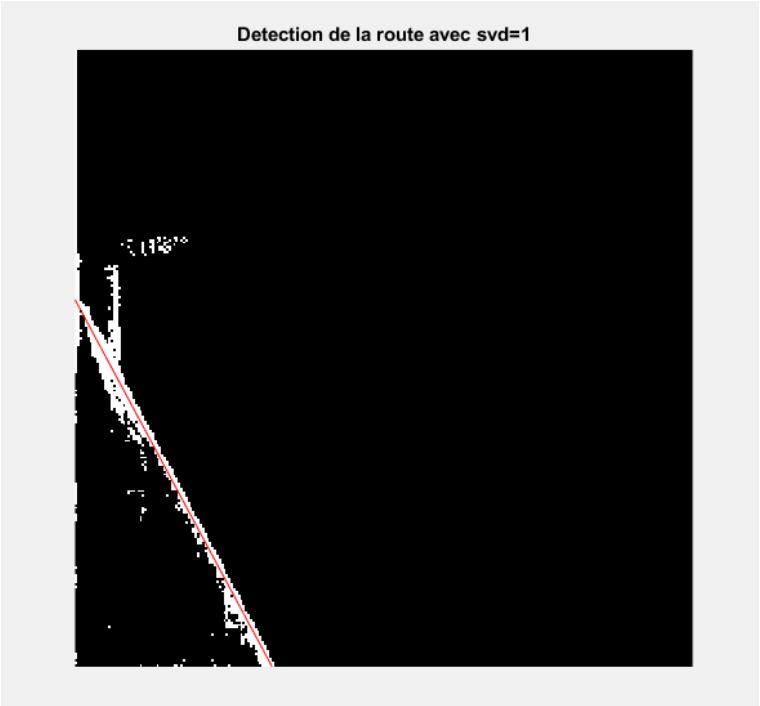

FIGURE 9 – Régression robuste d'une ligne sur le nuage de points $(v, disparite)$ pourla de seuil allant de 1

3.3 Obstacles :

Nous construisons l'histogramme en disparité du nombre de pixels dans l'image v-disparité qui ont une valeur supérieure à un seuil s_{vd} que nous choisissons d'une manière adéquate.Pour cela nous

utilisons la fonction *obstacles.m*.

```
function histo = obstacles(v_disp, b, range, s_obstacle, Id)
    %Inputs :
    % v_disp : est la v-disparité
    % b : le résultat de la fonction robustfit
    % range : intervalle de disparité
    % svd : seuil obstacle
    % Id : l'adresse de l'image de droite de type chaine de caractere. Ex : '01D.png'
    %Outputs :
    % figure 1 : Histogramme en disparité
    % figure 2 : Détection d'obstacle

    histo=zeros(size(v_disp,2));% création d'un tableau de 1D
    for i=1:size(v_disp,2) % parcour de l'image suivant les colonnes
        for j=1:size(v_disp,1)
            if v_disp(j,i)>s_obstacle
                histo(i)=histo(i)+1; % incrémentation de la valeur du tableau lorsque une
            end
        end
    end
    subplot(1,2,1);
    plot(histo); % affichage de l'histogramme
    title('Histogramme en disparité');

    obstacle=zeros(size(histo));
    for max=1 : length (histo)
        if histo(max)> s_obstacle
            obstacle(max)=1;
        end
    end
    max=range(2);
    while obstacle(max)~= 1
        max=max-1;
    end
    Y=b(1)+b(2).*max;
    subplot(1,2,2);
    % affichage de la ligne jusqu'a laquelle il n ya pas d'obstacle sur l'image original
    Id=rgb2gray(double(imread(Id))/255);
    imshow(Id);
    title("Détection d'obstacle");
    hold on
    plot(0:size(Id,2),Y+0*(0:size(Id,2)),'r');
end
```

Nous afficherons la ligne qui jusqu'à elle, il n'y pas d'obstacles.

```
%% Q3 : Obstacles :
for s_obstacle = 6 :9
    figure('Name',sprintf('s_obstacle=%d',s_obstacle));
    histo = obstacles(v_disp, b, range, s_obstacle,'01D.png' );
end

%On valide les parametres suivants :
% range = [0 96]; svd = 1; s_obstacle =9;
```

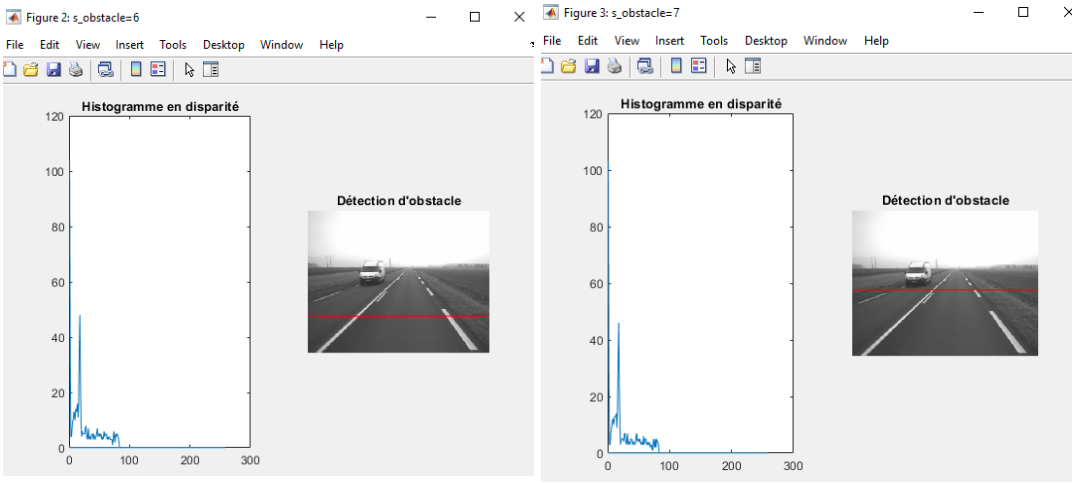

FIGURE 10 – la ligne jusqu'à laquelle il n'y pas d'obstacle pour s_obstacle = 6,7

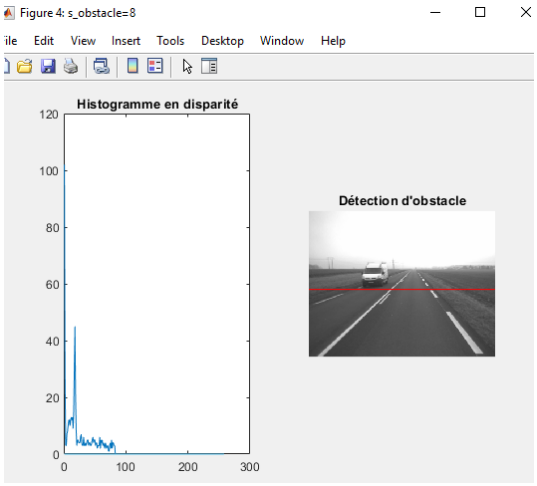

FIGURE 11 – La ligne jusqu'à laquelle il n'y pas d'obstacle pour s_obstacle = 8

Nous remarquons que a valeur de seuil modifie considérablement la ligne jusqu'à laquelle il n'y pas d'obstacles. Par mesure de sécurité, nous prenons $s\_obstacle = 9$

4 Application sur les images "03.png" et "03D.png" :

De même, nous réglons les paramètres pour cette paire d'images. Nous avons testé plusieurs valeurs de différents seuils, voilà les résultats :

```
%% Q5 : Sur les images  03G et 03D

range =[0 96];
disparitéSGM=calculDeDisparite('03D.png','03G.png', range);
v_disp = v_disparite(disparitéSGM, range,'03G.png');
figure
for svd = 1 :9
    b = profilDeLaRoute(v_disp , svd );
end
%on  valide svd =5
b = profilDeLaRoute(v_disp , 1 );

for s_obstacle = 10 :12
```

```
    figure('Name',sprintf('s_obstacle=%d',s_obstacle));
    histo = obstacles(v_disp, b, range, s_obstacle,'03D.png' );
end
% Par mesure de sécurité, on valide s_obstacle =6
s_obstacle =6;
figure('Name',sprintf('s_obstacle=%d',s_obstacle));
histo = obstacles(v_disp, b, range, s_obstacle,'03D.png' );

% Avec les valeurs de svd = 5 et s_obstacle = 6, le résultat n'est pas bon
% On conclut ainsi, qu'il faut reparamétrer

% Par mesure de sécurité, on valide s_obstacle =12
s_obstacle =12;
figure('Name',sprintf('s_obstacle=%d',s_obstacle));
histo = obstacles(v_disp, b, range, s_obstacle,'03D.png' );
```

Ainsi, nous trouvons :

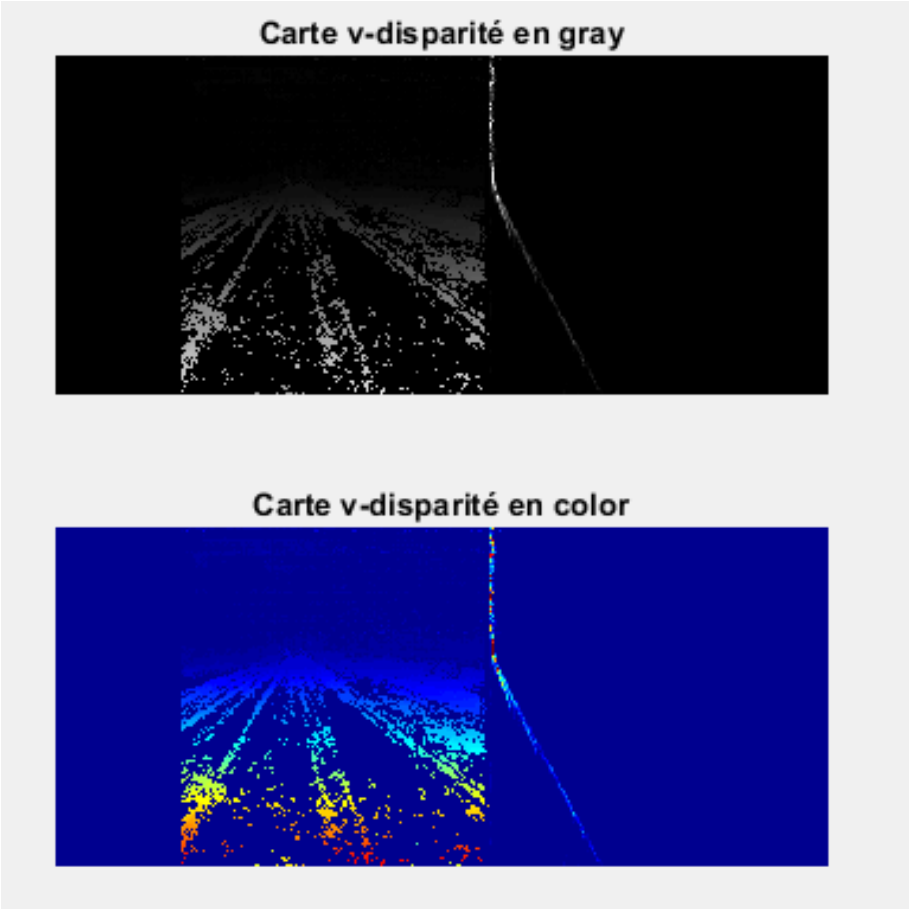

FIGURE 12 – histogramme du nombre d'appariements pour chaque couple de valeurs (v, disparité)

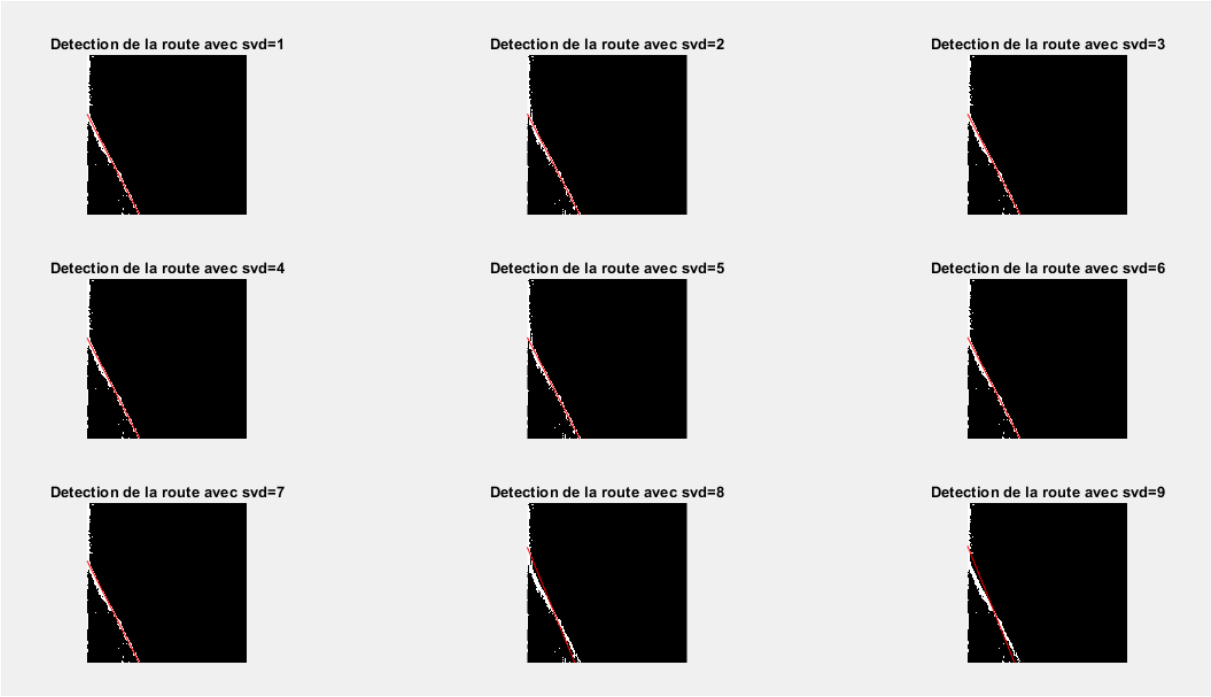

FIGURE 13 – Régression robuste d'une ligne sur le nuage de points (v, disparité) pour différentes valeurs de svd

Nous remarquons que la régression n'est pas faite comme dans l'exemple précédent. On pourra proposer une autre manière de modéliser la route que la régression linéaire. Ainsi, nous choisissons $svd = 5$, car il me semble le meilleur modèle. Bien évidement, nous pouvons utiliser des métriques permettant de choisir d'une manière optimale la valeur de svd.

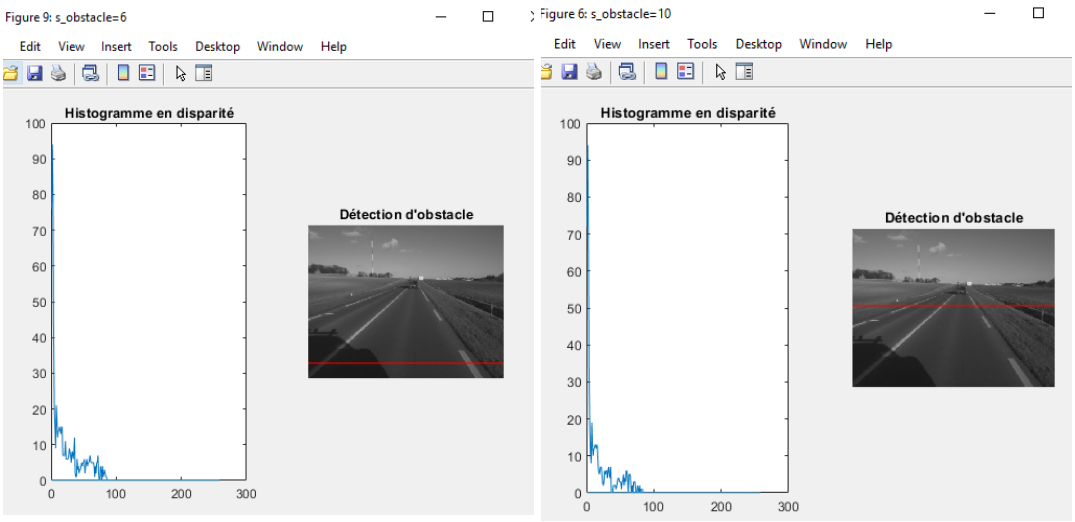

FIGURE 14 – la ligne jusqu'à laquelle il n'y pas d'obstacle pour s_obstacle = 6,10

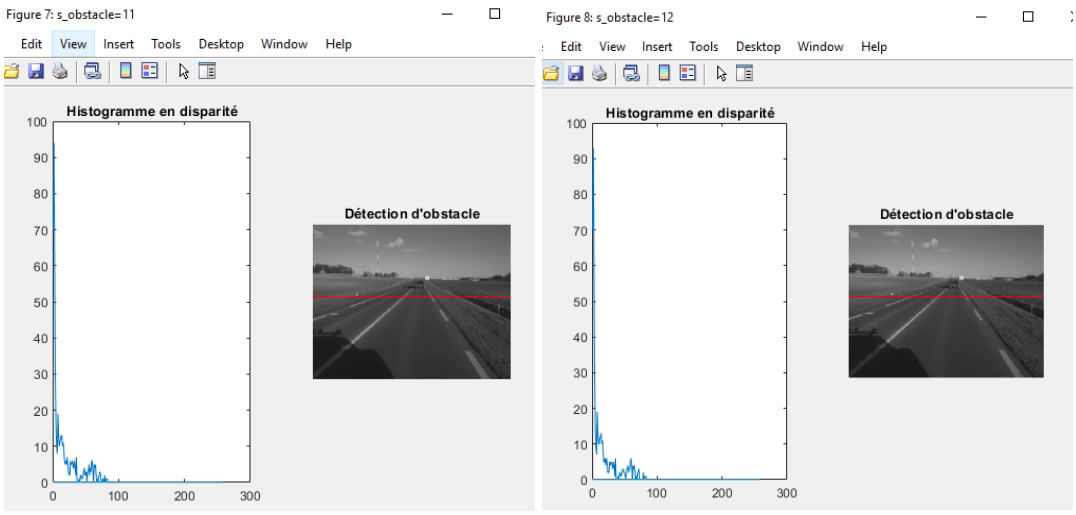

FIGURE 15 – la ligne jusqu'à laquelle il n'y pas d'obstacle pour s_obstacle = 11,12

Nous remarquons que a valeur de seuil modifie considérablement la ligne jusqu'à laquelle il n'y pas d'obstacles. Par mesure de sécurité, nous prenons $s\_obstacle = 12$.

5 Application sur les images "03G.png" et "03D.png" :

De même, nous réglons les paramètres pour cette paire d'images. Nous avons testé plusieurs valeurs de différents seuils, voilà les résultats :

```
%% Q4 : Sur les images  02G et 02D

range =[0 96];
disparitéSGM=calculDeDisparite('02D.png','02G.png', range);
v_disp = v_disparite(disparitéSGM, range,'02G.png');
figure
for svd = 1 :9
    b = profilDeLaRoute(v_disp , svd );
end
%on  valide svd =5
b = profilDeLaRoute(v_disp , 1 );

for s_obstacle = 6 :9
    figure('Name',sprintf('s_obstacle=%d',s_obstacle));
    histo = obstacles(v_disp, b, range, s_obstacle,'02D.png' );
end
% Par mesure de sécurité, on valide s_obstacle =6
s_obstacle =6;
figure('Name',sprintf('s_obstacle=%d',s_obstacle));
histo = obstacles(v_disp, b, range, s_obstacle,'02D.png' );
```

Ainsi, nous trouvons :

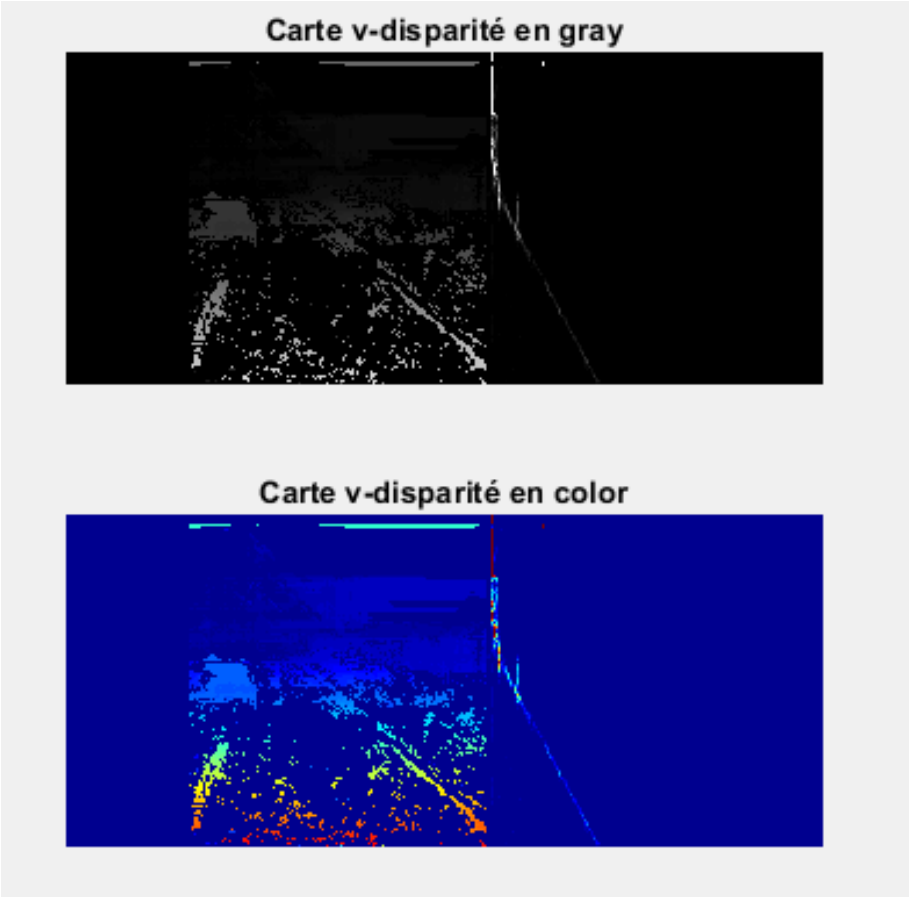

FIGURE 16 – histogramme du nombre d'appariements pour chaque couple de valeurs (v, disparité)

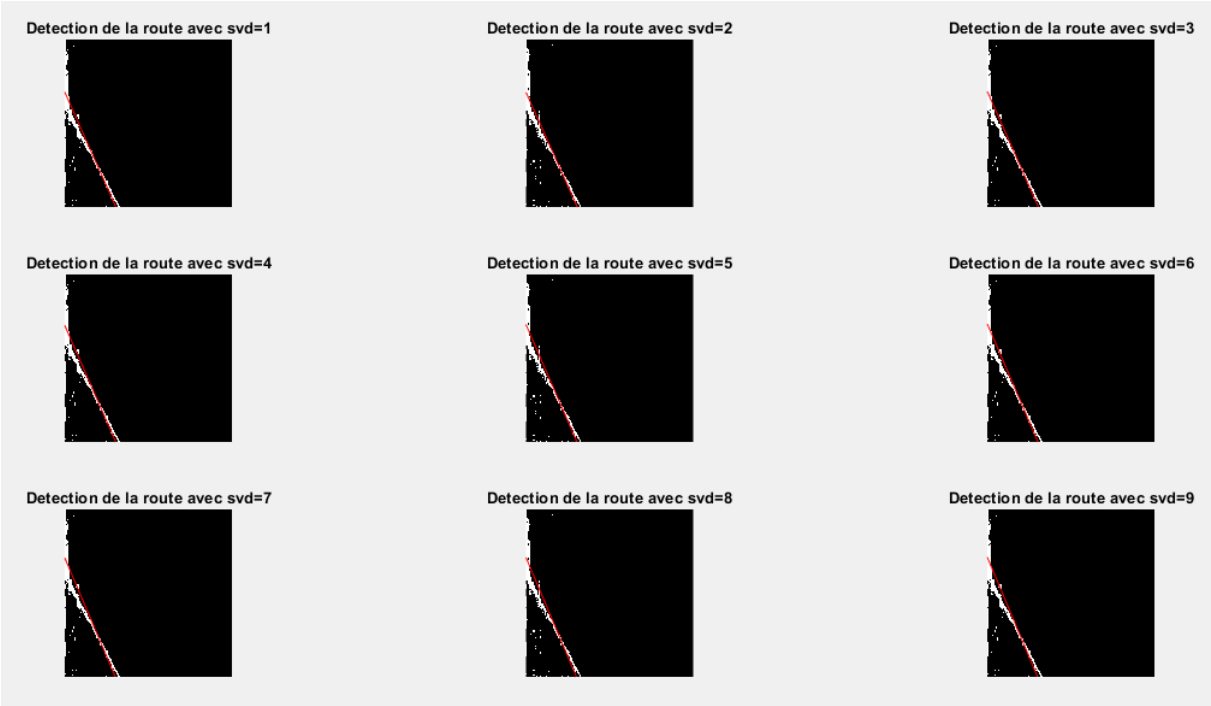

FIGURE 17 – Régression robuste d'une ligne sur le nuage de points (v, disparité) pour différentes valeurs de svd

Nous remarquons que la régression n'est pas faite comme dans l'exemple précédent. On pourra proposer une autre manière de modéliser la route que la régression linéaire. Ainsi, nous choisissons

$svd = 5$, car il me semble le meilleur modèle. Bien évidement, nous pouvons utiliser des métriques permettant de choisir d'une manière optimale la valeur de svd.

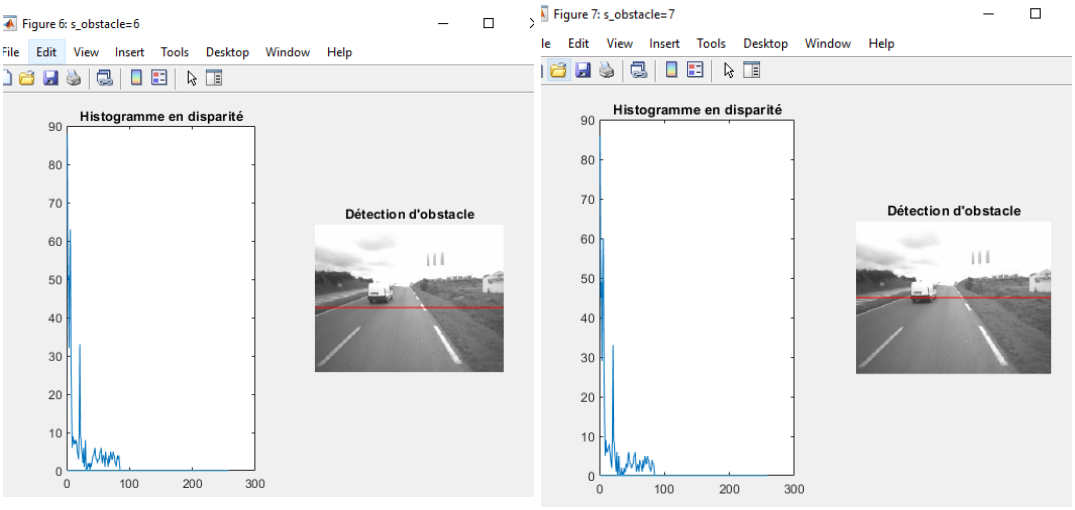

FIGURE 18 – la ligne jusqu'à laquelle il n'y pas d'obstacle pour s_obstacle = 6,7

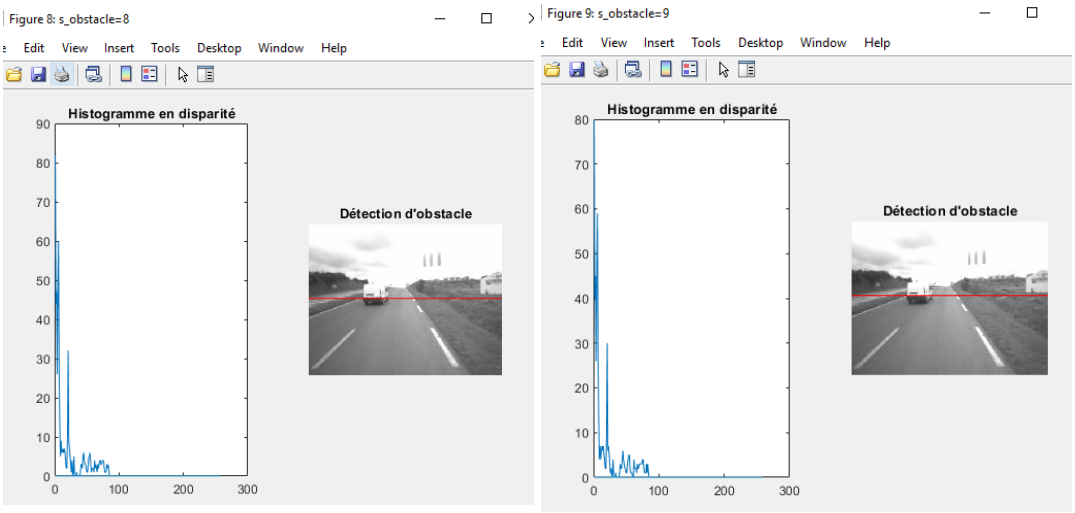

FIGURE 19 – la ligne jusqu'à laquelle il n'y pas d'obstacle pour s_obstacle = 8,9

Nous remarquons que a valeur de seuil modifie considérablement la ligne jusqu'à laquelle il n'y pas d'obstacles. Par mesure de sécurité, nous prenons $s\_obstacle = 12$.

6 Conclusion :

Nous concluons ainsi que les modifications sont nécessaires pour détecter correctement la route et les obstacles.

7 Bibliographie :

1. R. LABAYRADE, D. AUBERT et J.-P. TAREL : Real time obstacle detection in stereo vision on non-flat road geometry through v-disparity representation. Dans les actes de IEEE Intelligent Vehicle Symposium (IV'2002), volume 2, paghes 646-651, Versailles, France, juin 2002.