BOISSARD Benjamin
BAROCHI Guillaume L3 ESI

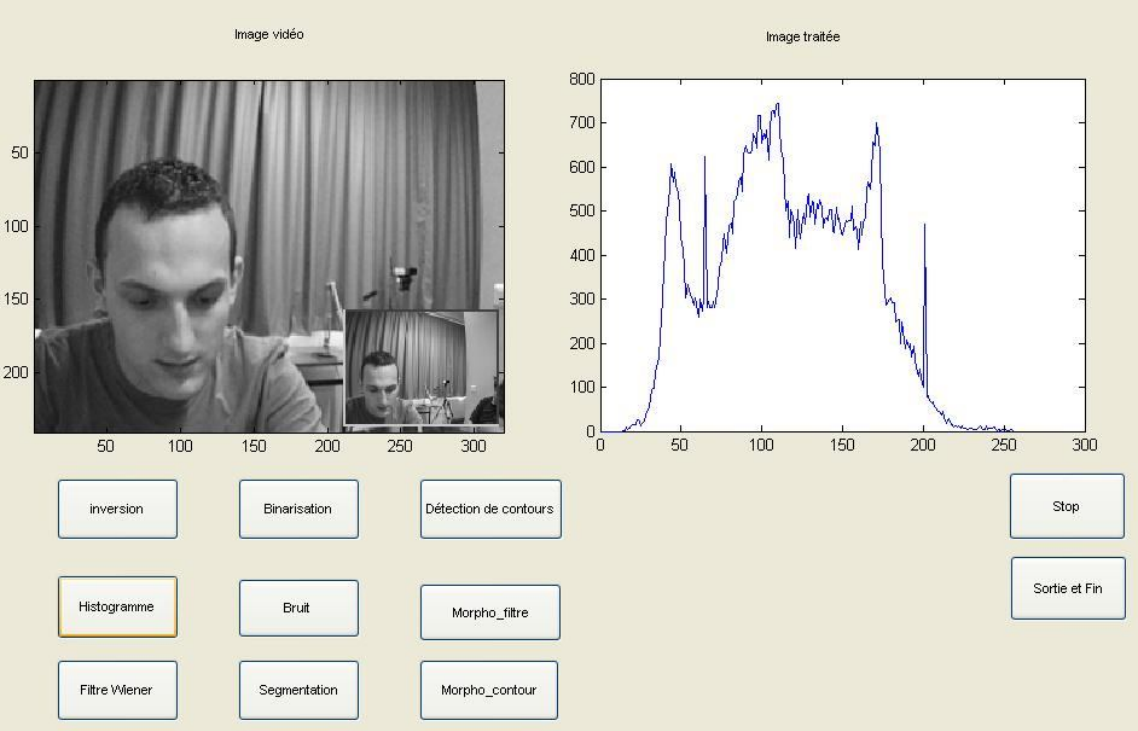
TP
Traitement d'images

TP 1 : Histogramme et Binarisation
TP 2 : Filtrage de bruit et de flou
TP 3 : Détection de contours (méthodes linéaires)
TP 4 : Segmentation d'images par seuillage
TP 5 : Morphologie mathématique

TP1 : Histogramme et binarisation :

Histogramme :

L'histogramme permet d'obtenir le nombre de pixel de la même couleur dans une image.



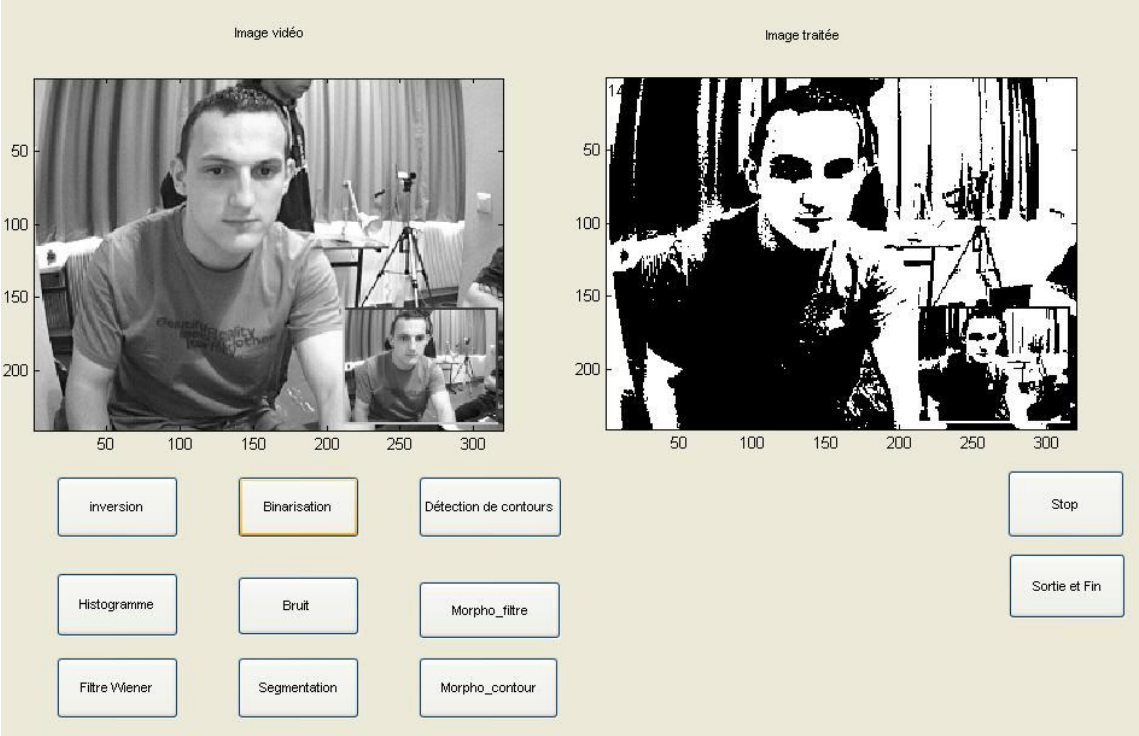
Si on fait varier la luminosité on voit que les pic se déplace soit vers la gauche quand la luminosité diminue et vers la droite quand la luminosité augmente.

Programme :

```
    y = (getdata(handles.vid,1,'uint8'));   %traitement
    I4 = rgb2gray(y);

% création de l'histogramme manuellement
    %histo=zeros(256);
%       for j=1:320
%           for i=1:240
%               histo(I4(i,j)+1)=histo(I4(i,j)+1)+1;
%
%           end
%       end
    axes(handles.axes1);
    subimage(I4);

    axes(handles.axes2);
    plot(imhist(I4)) ; % histogramme via matlab
```

Binarisation :

Le but d'une binarisation est de passer une image en noir et blanc, ce qui consiste à avoir que des pixels noir et blanc.



On définie un seuil qui permettra de savoir si un pixel deviendra noir ou blanc. Si le pixel est inférieur au seuil alors le pixel deviendra noir si non le pixel devient blanc.

Pour déterminer le seuil on peut utiliser l'histogramme et on donnera la valeur du seuil manuellement ou on peut utiliser un seuillage automatique via Matlab.

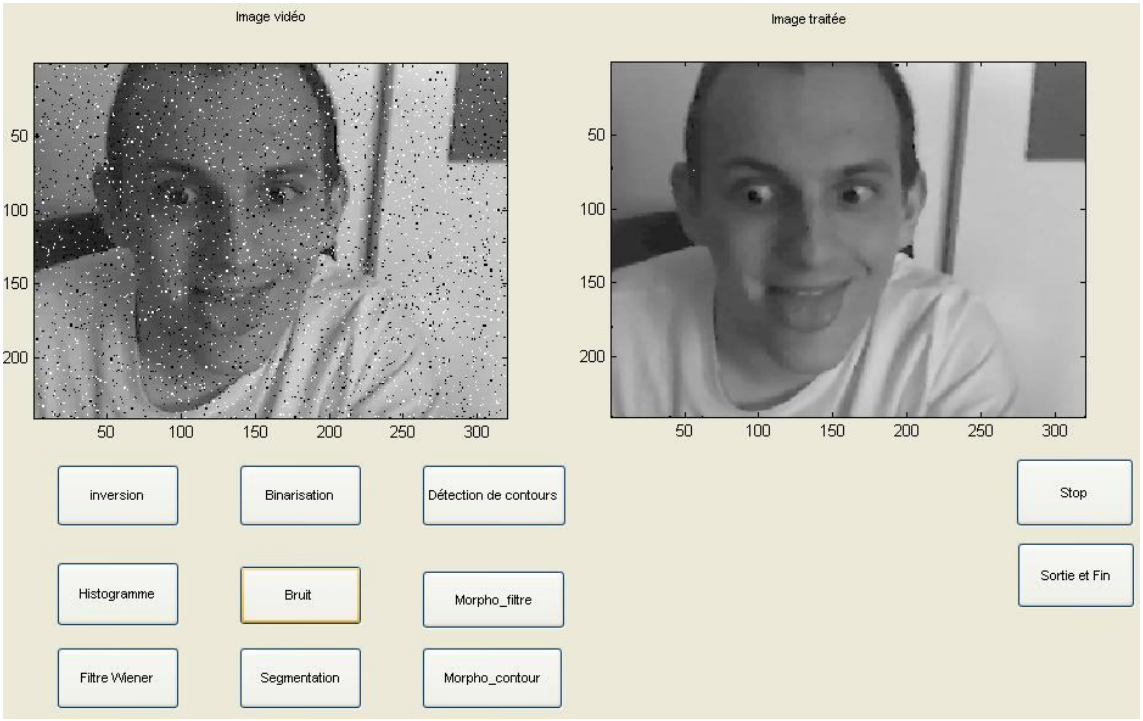
Programme :

```
y = (getdata(handles.vid,1,'uint8'));   %traitement
      I4 = rgb2gray(y);
%*****************************
%       Calcule du seuil
%*****************************
      %calucle des m:
      m0=1;
      m1=mean2(I4);
      m2=mean2(I4.^2);
      m3=mean2(I4.^3);

      %calcule des C:
      C1=(m3-(m1*m2))/(m2-m1);
      C0=(-m2-(C1*m1))/m0;

      %calcule des z:
      z1=(-C1-sqrt(C1^2-4*C0))/2;
      z2=(-C1+sqrt(C1^2-4*C0))/2;
```

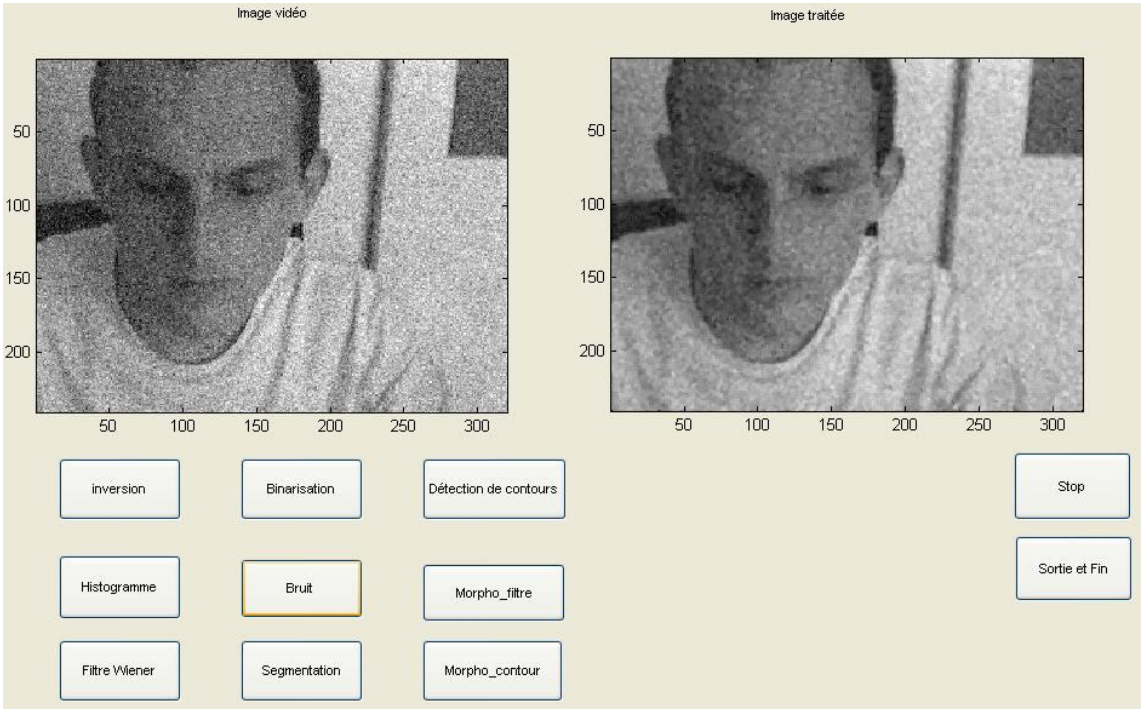
```
seuil=(z1+z2)/2;


%       première solution:

%       bin=zeros(240,320);
%       for i=1:240
%           for j=1:320
%             if I4(i,j)>seuil;
%                bin(i,j)=255;
%             end
%           end
%       end

bin=(I4>seuil)*255;

text(2,10,num2str(seuil));
%
%         Solution via matlab de la binarisation :
%       level=graythresh(I4);                   %calcule seuil
%       bin = im2bw(I4,level);                  %binarisation matlab


axes(handles.axes1);
subimage(I4);

axes(handles.axes2);
subimage(bin);
```

TP 2 : Filtrages de bruit et de flou :

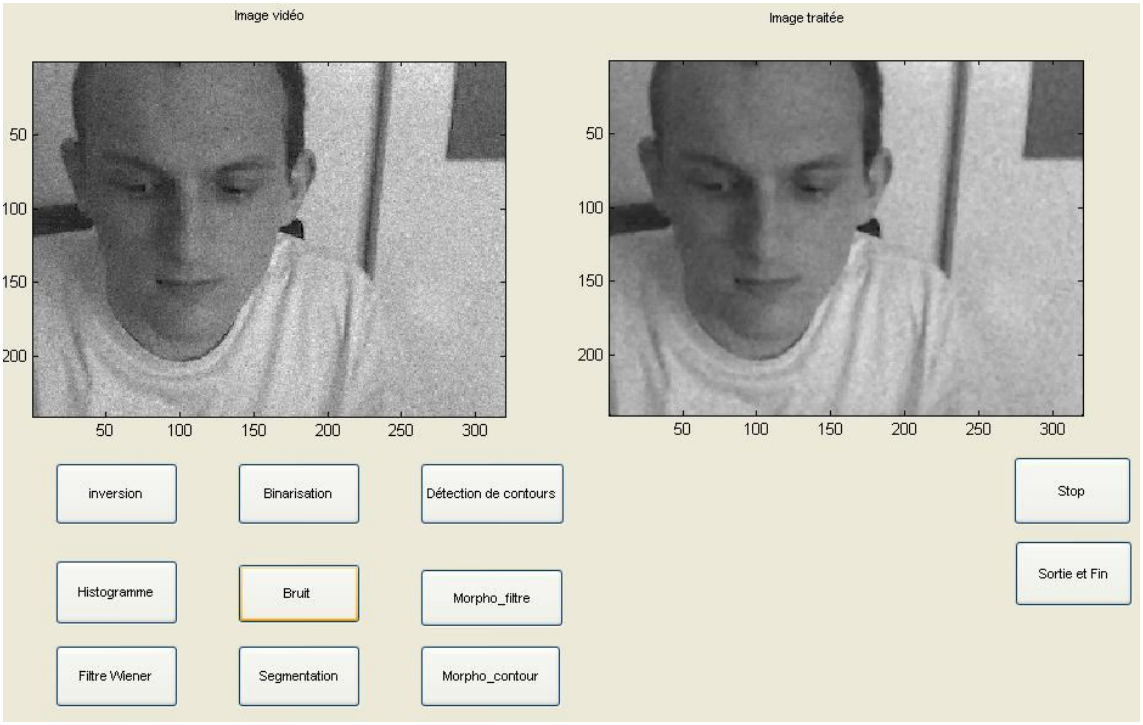
Filtre de bruit : filtre médian



Sel et poivre : Il ajoute un tout petit peu de flou à l'image mais reste le meilleur par rapport aux autres car le bruit est totalement éliminé.



Gaussien : L'image reste un peu floue et il reste des pixels visibles sur l'image et on constate que les pixels sont moins fins.



Poisson : L'image reste toujours floue donc on n'a pas d'amélioration sur l'image.

Programme :

```
y = (getdata(handles.vid,1,'uint8'));   %traitement
        I4 = rgb2gray(y);
        blanc = imnoise(I4,'poisson'); %création du bruit sur l'image
        L = medfilt2(blanc,[3 3]);

        axes(handles.axes1);
        subimage(blanc);

        axes(handles.axes2);
        subimage(L);
```

Filtrage de flou : filtre de Wiener

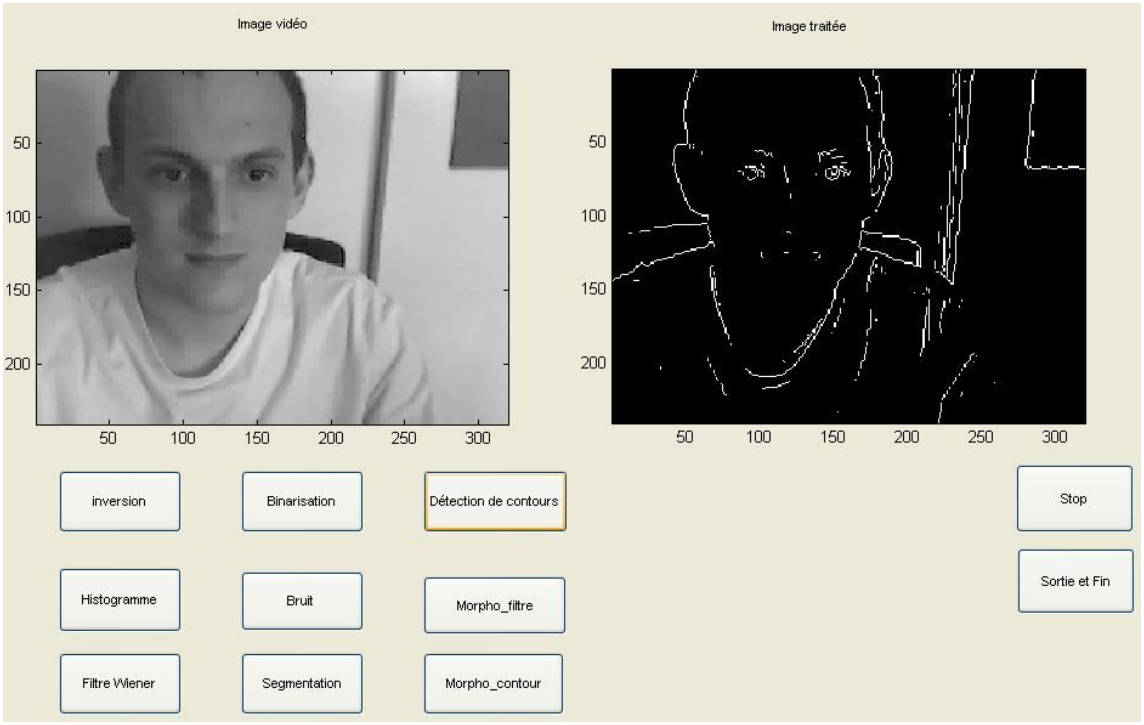
Le filtrage de Wiener permet de partir d'une images floutée et d'arriver à une image plus ou moins nette mais cela est possible car on connaît le masque qui donne le flou de l'image de départ.

Programme :

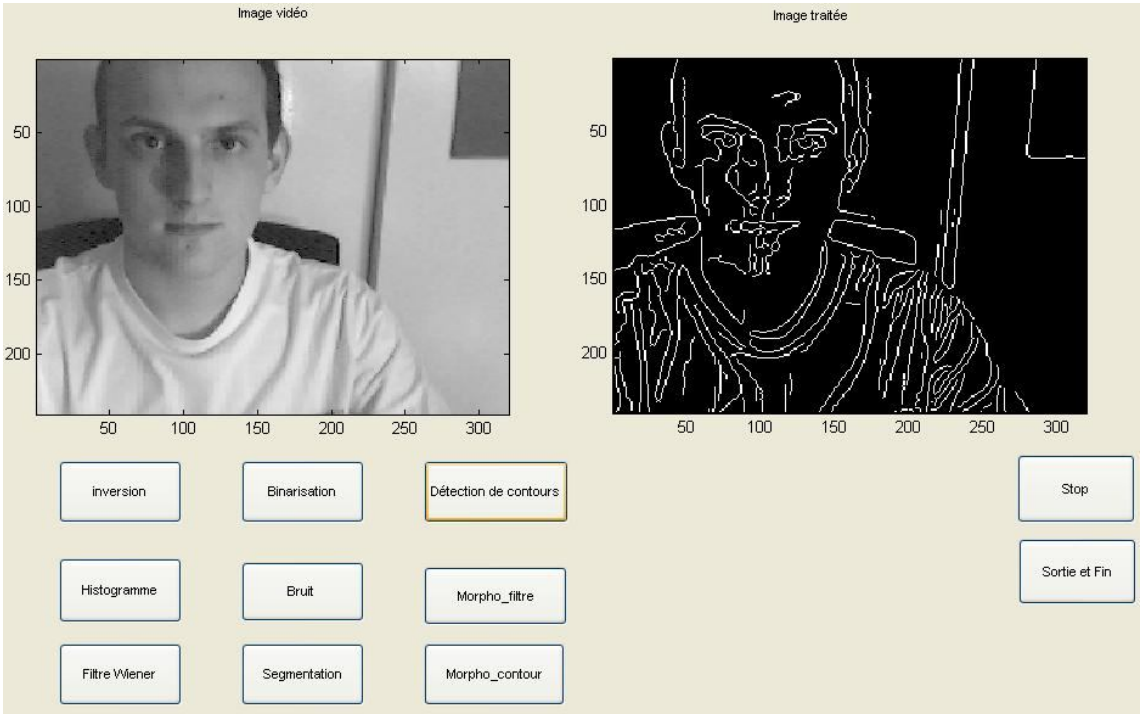
```
y = (getdata(handles.vid,1,'uint8'));   %traitement
      I4 = rgb2gray(y);

      L = imfilter(I4,h,'circular','conv');
      wnr1 = deconvwnr(L,h);
      axes(handles.axes1);
      subimage(L);

      axes(handles.axes2);
      subimage(wnr1);
```

TP 3 : Détection de contours

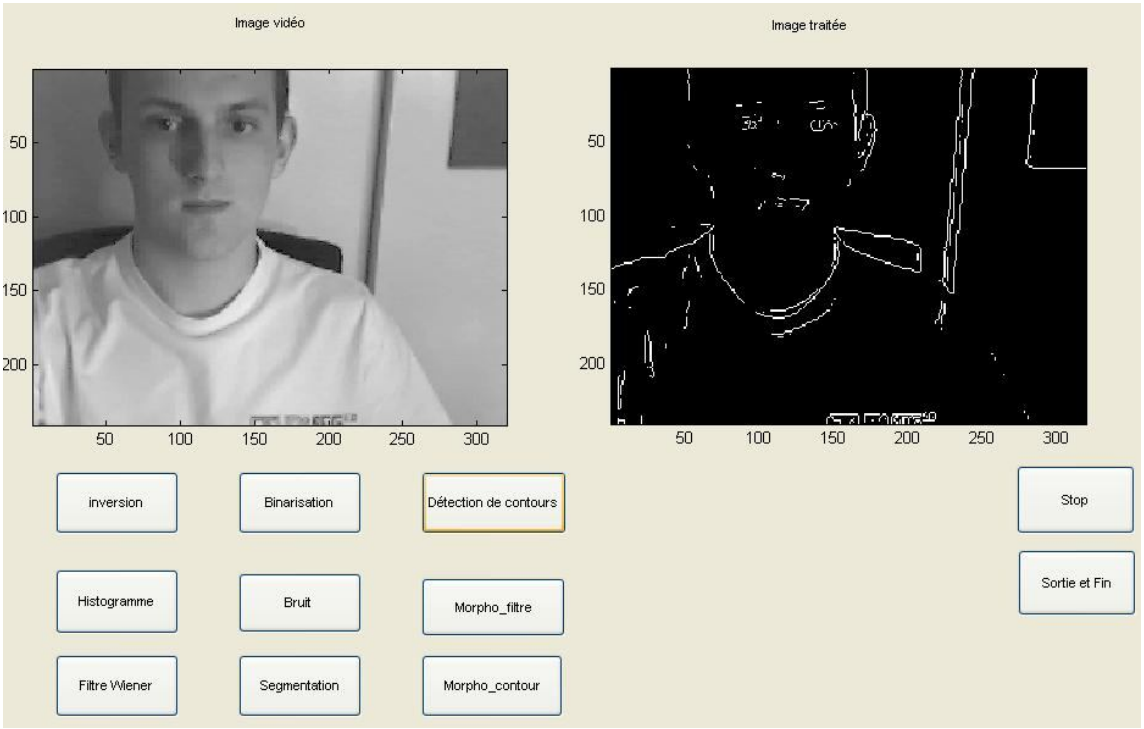
Images sans bruit gaussien :



Prewitt : On n'obtient pas les contours au complet mais on a l'essentiel des contours et on reconnaître facilement les formes.

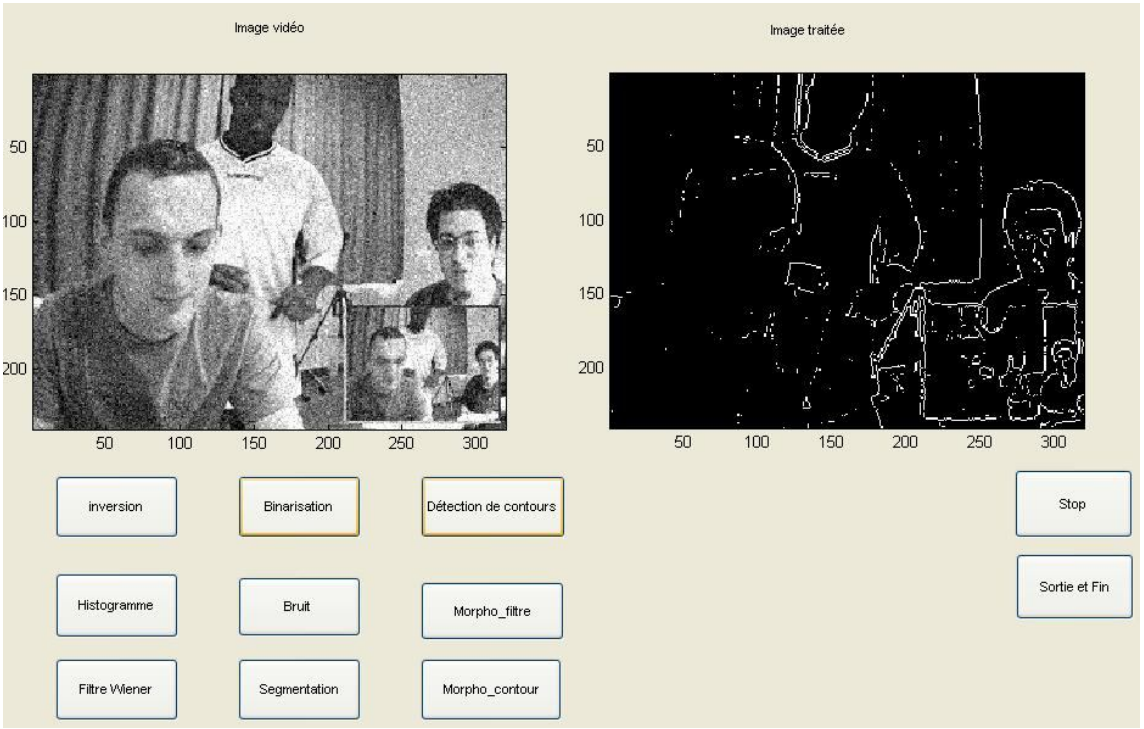


Canny : L'image prend toutes les variations alors on a plein de contours dans l'image ce qui rend l'image presque inutile.

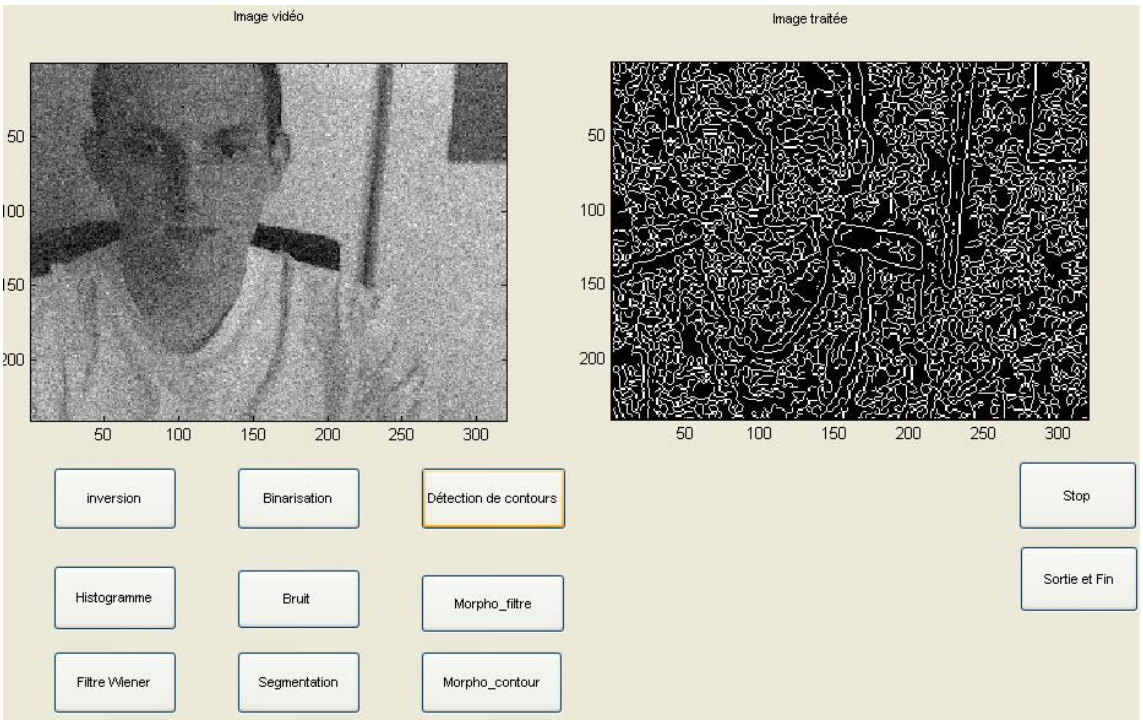


Sobel : On a très peu de contour détecté alors on reconnaît très peu la forme.

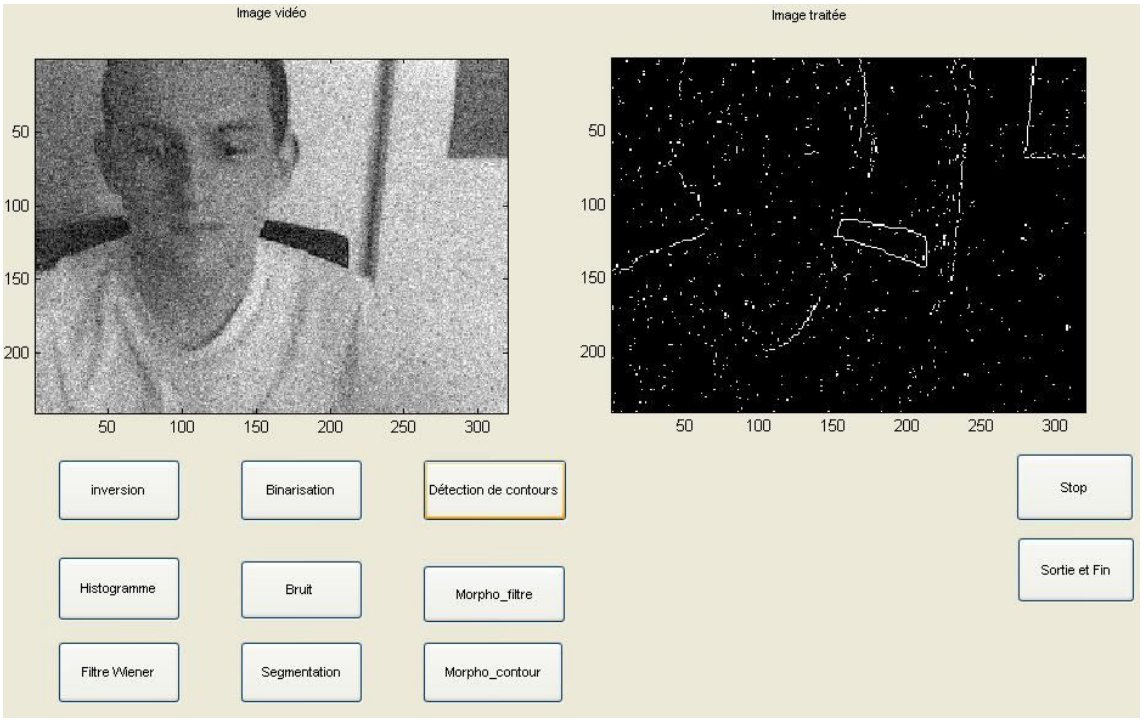
<u>Images avec du bruit (gaussien) :</u>



Prewitt : L'image est plutôt bonne mais on a pas la totalité des contours donc on élimine une partie du bruit en faisant la détection de contour.



Canny : On voit les contours du bruit ce qui rend l'image inutile.



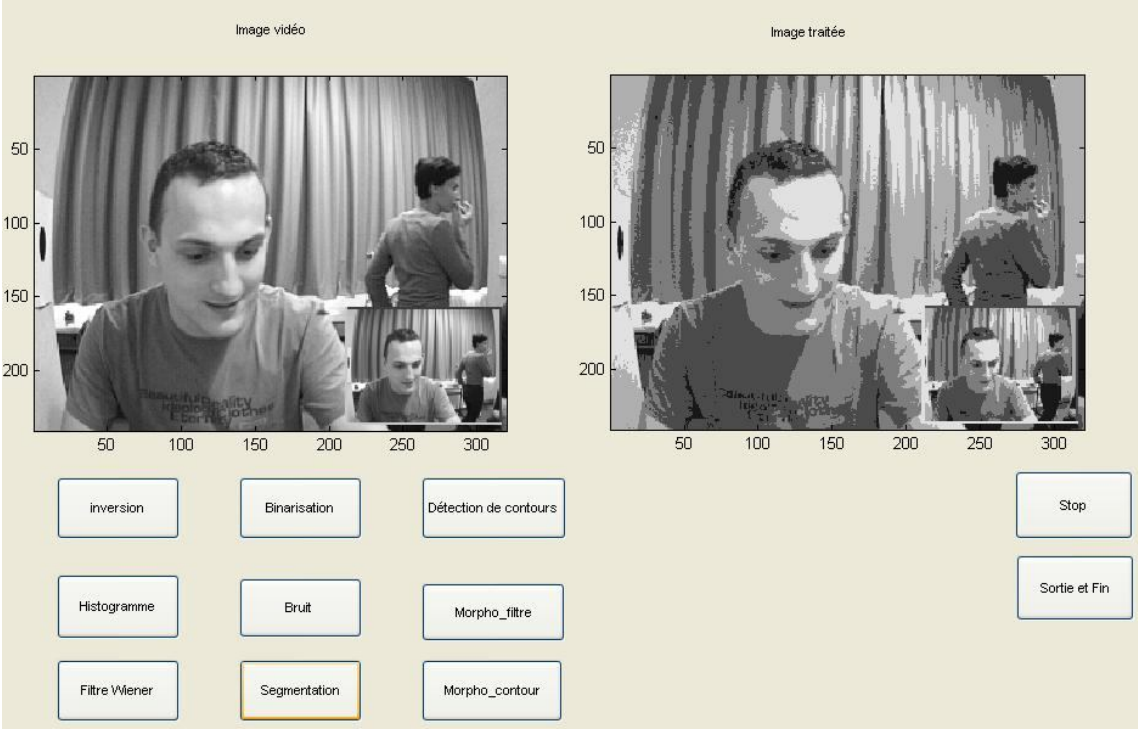
Sobel : Les contours sont mal détectés et on a le contour du bruit.

Programme :

```
y = (getdata(handles.vid,1,'uint8'));   %traitement
```

```
I4 = rgb2gray(y);
blanc = imnoise(I4,'gaussian');
BW1 = edge(blanc,'prewitt');
axes(handles.axes1);
subimage(blanc);

axes(handles.axes2);
subimage(BW1);
```

TP 4 : Segmentation d'image par seuillage



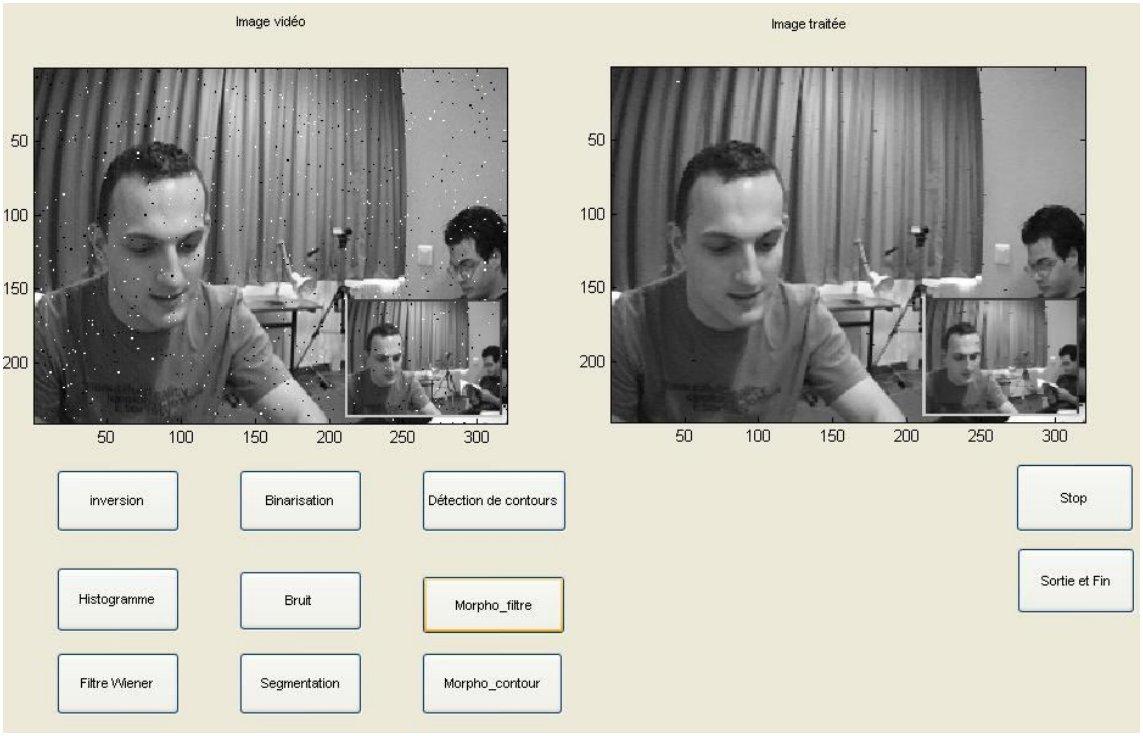
La segmentation consiste a prend des pixels qui sont proche de couleur pour les regrouper et de leur donner par la suite la même valeur suivant la valeur moyen du groupe.

Programme :

```
y = (getdata(handles.vid,1,'uint8'));   %traitement
      I4 = rgb2gray(y);
      blanc = I4 ;%imnoise(I4,'gaussian');
      bin1=(blanc<50)*25;
      bin2=((49<blanc) & (blanc<100))*75;
      bin3=((99<blanc) & (blanc<150))*125;
      bin4=((149<blanc) & (blanc<200))*175;
      bin5=(199<blanc)*225;

      seg=uint8(bin1+bin2+bin3+bin4+bin5);


      axes(handles.axes1);
      subimage(blanc);

      axes(handles.axes2);
      subimage(seg);
```

TP 5 : Morphologie mathématique

Filtrage par morphologie mathématique :

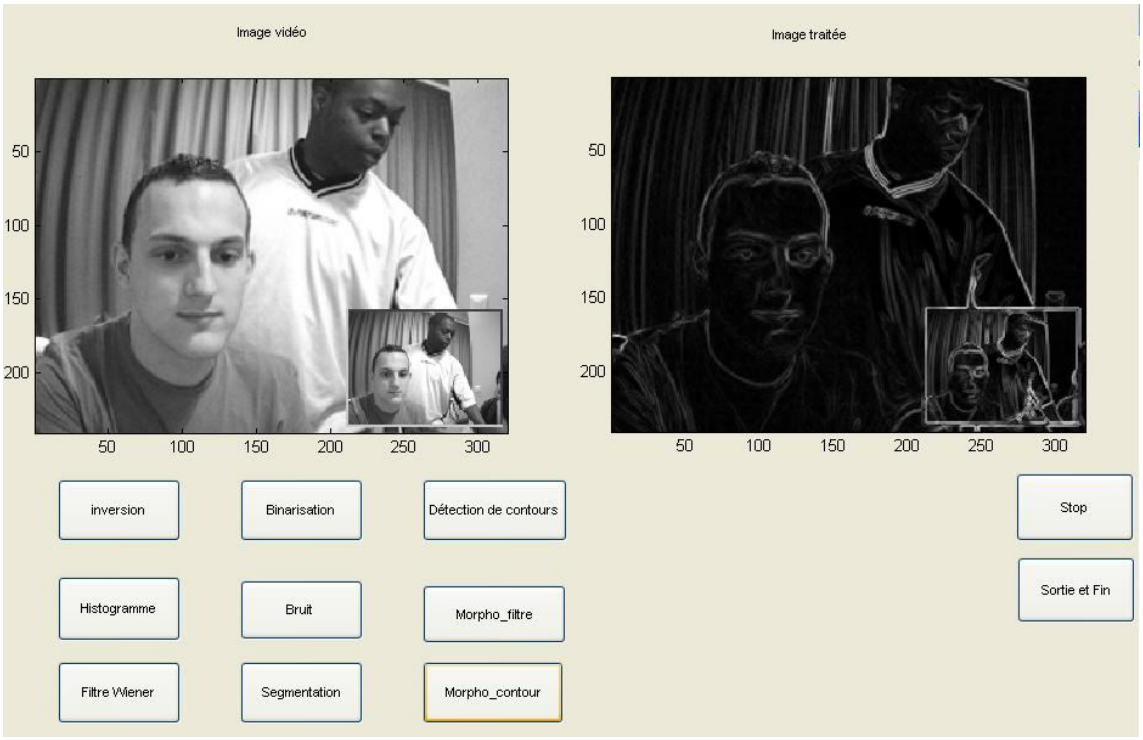


On par de l'image d'entré on lui fait un érosion et ensuite une dilation se qui correspond a se moment la une ouverture puis on refait un dilatation et la on commence obtenir un résultat interessant.

Programme :

```
y = (getdata(handles.vid,1,'uint8'));   %traitement
      I4 = rgb2gray(y);
      h1=[0 0 0;
          0 1 1;
          0 0 0];
       img= imnoise(I4,'salt & pepper',0.01);
      img1 = imopen(img,h1);
      img2 = imclose(img1,h1);

      axes(handles.axes1);
      subimage(img);

      axes(handles.axes2);
      subimage(img2);
```

Détection de contour par morphologie mathématique :



Pour obtenir la détection de contour on calcul en valeur absolue et on fait la soustraction de l'image dilaté moins l'image érodé.

Programme :

```
y = (getdata(handles.vid,1,'uint8'));   %traitement
      I4 = rgb2gray(y);
      h1=[0 1 0;
          1 1 1;
          0 1 0];
       img= imnoise(I4,'salt & pepper',0.01);
      img1 = imerode(I4,h1);
      img2 = imdilate(I4,h1);
      img5= abs(img2-img1);



      axes(handles.axes1);
      subimage(I4);

      axes(handles.axes2);
      subimage(img5);
```