Traitement d'images TP2 : Histogramme
Segmentation d'une image, Etalement, Egalisation

Line POUVARET, (Hamdi BENAOUI)

2015-2016

Segmentation d'une image par seuillage des niveaux de gris

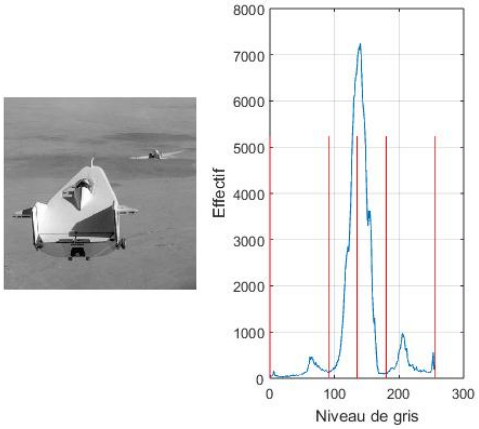
1°) Algorithme de segmentation (lignes 20-75)

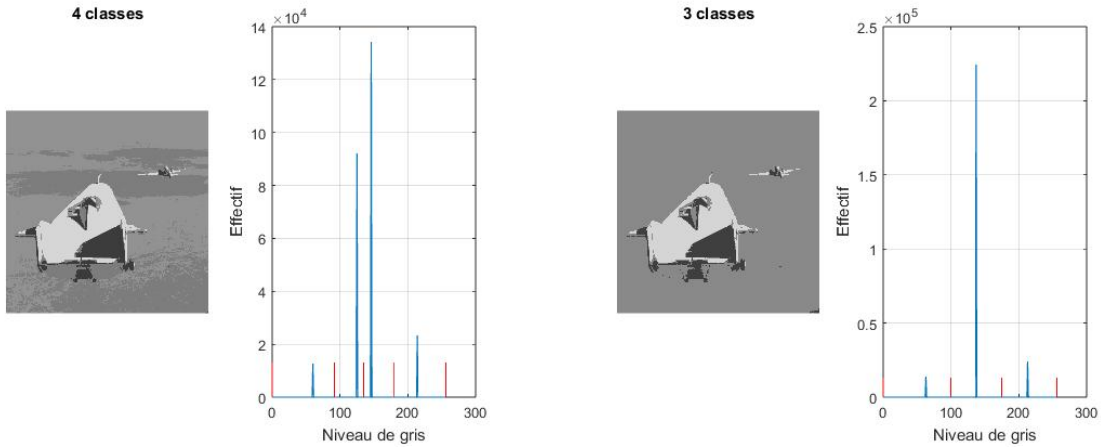
Cet algorithme correspond à l'algorithme des K-Moyennes. Il va itérer maximum 200 fois afin de trouver les seuils de niveau de gris de l'image qu'on choisit pour segmenter l'image par la suite. A partir de l'histogramme des niveaux de gris des pixels de l'image, on arrive à séparer les groupes de pixels ayant sensiblement le même niveau de gris et ces séparations vont constituer nos seuils.

2°) Instructions ajoutées dans le programme

```
1  Pour le mode 'thres' :
2  level =(v_threshold(k) + v_threshold(k+1))/2;
3
4  Pour le mode 'mean' :
5  if(N(k)~=0)
6                  current_X = [v_threshold(k):v_threshold(k+1)-1]';
7                  h2=histo(current_X+1);
8                  level = h2'*current_X/N(k);
9              else
10                 level = 0;
```

3°)



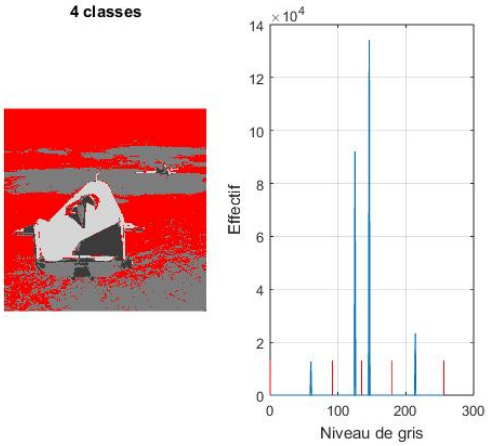


4°) Instructions ajoutées dans le programme

```
[nbMax,classeMax] = max(seg_histo(:));
map = gray(256);
map(classeMax,:) = [1 0 0];

figure
colormap(map);
subplot(1,2,1)
image(segmented_pic)
axis equal
axis off
title([int2str(nbr_class) ' classes'])
subplot(1,2,2)
plot(seg_bin,seg_histo);
hold on
for k = 1 : length(v_threshold)
    plot(v_threshold(k)*[1 1], [0 nbpix/20], 'r-')
end
grid on
xlabel('Niveau de gris')
ylabel('Effectif')

figure(n1)
subplot(1,2,2)
hold on
for k = 1 : length(v_threshold)
    plot(v_threshold(k)*[1 1], [0 nbpix/50], 'r-')
end
```



5°)

Modification d'histogramme sur une image en noir et blanc

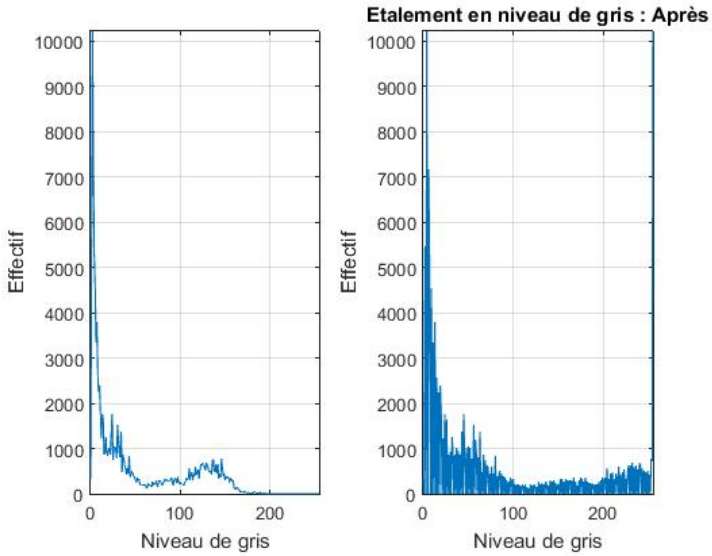
A°) Etalement d'un histogramme de niveaux de gris

1°) Cet appel permet de générer un histogramme à partir d'une image et d'un nombre de couleurs (niveaux de gris ici). h correspond à la hauteur des barres, et n l'indice des barres de l'histogramme. L'histogramme représente le nombre de pixels par niveau de gris.

2°) Instructions ajoutées dans la fonction histo_etalement :

```
1 pic = 255*((original_pic-pic_min)/(pic_max-pic_min));
2 pic = uint8(max(min(pic,255),0));
```

On remarque sur l'histogramme qu'on a un peu plus étalé nos niveaux de gris entre 0 et 255.



1°)
- Quand on met factor_min et factor_max à 1 tous les deux, on obtient l'image d'origine comme image de sortie et les deux histogrammes sont les mêmes (donc on n'a pas eu d'étalement).
- Quand on met factor_min et factor_max à 0.5 tous les deux, on obtient une image très sombre et l'histogramme de sortie est compressé vers les niveaux de gris sombres.

- Factor_max=1, factor_min=0 → image de sortie = image d'origine
- Plus on augmente sensiblement la valeur de factor_max (>1) plus on va étaler nos pixels et les éloigner des nuances très sombres.
- Dès que factor_max < 1, on compresse les pixels vers les niveaux de gris plus sombres.

2°) L'image d'origine est globalement très sombre au départ. La majorité des pixels se situe dans les niveaux de gris sombres mais il y a un certain nombre de pixels proches de nuances plus claires (ceux qui constituent les nuances du ciel).

Quand on va augmenter notre alpha (factor_max), > 1, on étale les pixels beaucoup plus vers les niveaux de gris sombres. Si on augmente trop alpha, on aura tendance à avoir une image beaucoup trop claire. Il faudrait arriver à étaler énormément pour les pixels très nombreux des nuances sombres et étaler moins pour les nuances claires (sinon le ciel apparaît beaucoup trop blanc).

B°) Egalisation d'histogramme de niveaux de gris

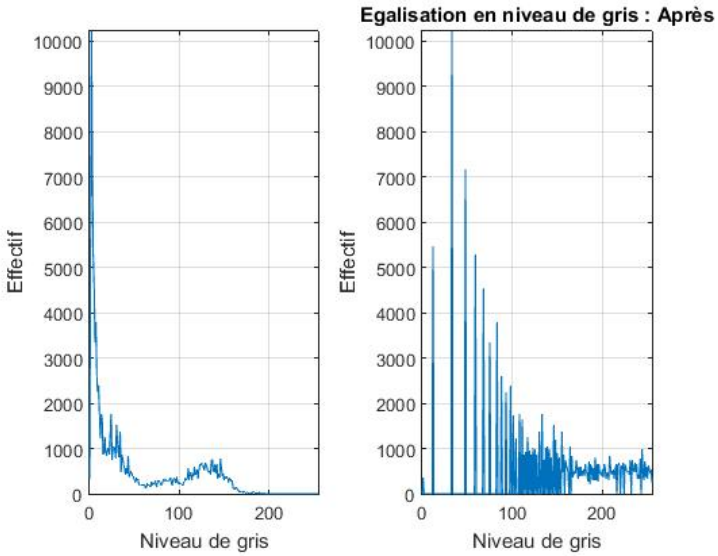
1°) `lut = 255/nbr\_pixels*cumsum(old\_histo);`

C'est cette ligne de code qui permet de calculer notre fonction de transformation.
$\phi(n_e)$ = (N_{max} / NT) * $ch_e(n_e)$ Avec :

- $\phi(n_e)$ = lut → la fonction de transformation
- N_{max} = 255 → Niveaux de gris max dans l'image
- NT=nbr_pixels → nombre de pixels total
- $ch_e(n_e$ = cumsum(old_histo) → cumul de l'histogramme
  La suite de la fonction permet de reshape l'image avec la fonction de transformation lut et on reconvertit en int.

2°) On voit que sur l'image, on a énormément diminué le contraste entre les nuances sombres et les nuances claires. L'image est globalement plus composée de nuances moyennes.

On constate bien que sur l'histogramme, les valeurs des pixels sont plus proches du centre qu'avant. On a essayé d'égaliser la répartition des pixels sur l'histogramme.



3°) On remarque que les parties qui étaient les plus sombres sur l'image sont devenues, après égalisation, moins précises.

On a perdu de l'information au niveau du détail et cela forme des paquets de pixels incohérents à certains endroits (notamment en bas à gauche).

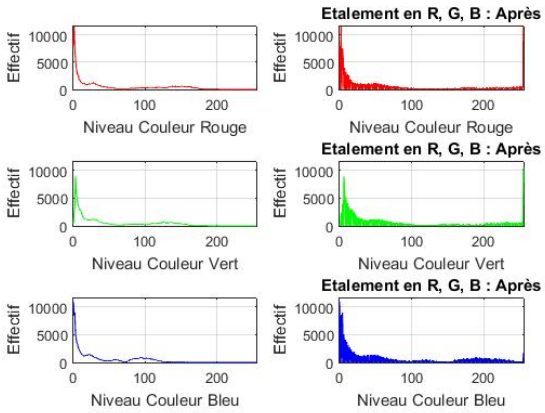
Modification d'histogramme sur une image en couleurs, approche naïve

C°) Etalement d'histogramme de couleurs

1°) Instructions ajoutées au programme :

```
[pic_RGB_linearized,h_r,h_r0] = histo_etalement(pic(:,:,1),factor_max,
    factor_min);
[pic_RGB_linearized,h_g,h_g0] = histo_etalement(pic(:,:,2),factor_max,
    factor_min);
[pic_RGB_linearized,h_b,h_b0] = histo_etalement(pic(:,:,3),factor_max,
    factor_min);
```

On obtient les résultats suivants :



On a généré de fausses couleurs en voulant étirer nos histogrammes de chaque composante. (Puisque chaque histogramme est différent)

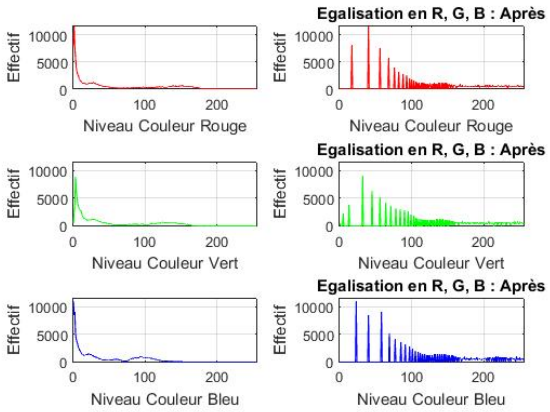
2°) En faisant varier les facteurs alpha et beta, on constate que les nuances sombres vont soit se rapprocher du bleu soit s'en séloigner et les nuances claires vont soit s'approcher du jaune soit s'en éloigner.

D°) Egalisation d'histogramme de couleurs

1°) Instructions ajoutées au programme :

```
[pic_RGB_equalized,h_r,h_r0] = histo_egalisation(pic(:,:,1));
[pic_RGB_equalized,h_g,h_g0] = histo_egalisation(pic(:,:,2));
[pic_RGB_equalized,h_b,h_b0] = histo_egalisation(pic(:,:,3));
```

On obtient les résultats suivants :



2°) L'image de Venise est devenue pratiquement entièrement jaune, les pixels ont tous été rapprochés vers la même nuance de couleur (ici le jaune). On voit bien sur les histogrammes que les pixels sont répartis de manière plus uniforme et plus vers des valeurs moyennes.

Changement de base « couleur » (NON FINI)

E°)

1°) Plus une couleur va contenir du rouge et plus Cr va être positive. Plus elle va contenir du vert, plus elle va avoir une valeur négative.

Une couleur jaune a un Cr de 0. Une couleur rouge aura un Cr maximum et une couleur verte aura un Cr minimum.

Cb code l'opposition de couleur bleu-jaune.

Cr code l'opposition de couleur rouge-vert.

2°) • instructions ajoutées à la fonction convert_RGB_to_YCbCr :

```
pic_YCbCr = reshape(double(pic_rgb)./255,[],3)*M;
pic_YCbCr = reshape(pic_YCbCr,size(pic_rgb));
```

• instructions ajoutées à la fonction convert_YCbCr_to_RGB :

```
M = [ [ 0.299 , 0.587 , 0.114];...
      [-0.168 ,-0.331 , 0.5];...
      [ 0.5 ,-0.4187 ,-0.0813] ];

pic_rgb = reshape(double(pic_YCbCr), [], 3)*inv(M);
pic_rgb = reshape(pic_rgb, size(pic_YCbCr));

pic_rgb = pic_rgb.*255;
pic_rgb = uint8(pic_rgb);
```

Etalement d'histogramme de couleurs (YCbCr)

F°)

1°) Instructions ajoutées dans le programme :

```
pic_Lum_linearized = uint8(zeros(size(pic)));
pic_Lum_linearized=convert_rgb_to_YCbCr(pic);
[pic_Lum_linearized_Y,h,h0] = histo_etalement(pic_Lum_linearized(:,:,1),
    factor_max,factor_min);
pic_Lum_linearized(:,:,1) = double(pic_Lum_linearized_Y)./255;
pic_Lum_linearized = convert_YCbCr_to_rgb(pic_Lum_linearized);
```

Egalisation d'histogramme de couleurs (YCbCr)

G°)

1°) Instructions ajoutées dans le programme :

```
pic_Lum_equalized = uint8(zeros(size(pic)));
pic_Lum_equalized = convert_rgb_to_YCbCr(pic);
[pic_Lum_equalized_Y,h,h0] = histo_egalisation(pic_Lum_equalized(:,:,1));
pic_Lum_equalized(:,:,1) = double(pic_Lum_equalized_Y)./255;
pic_Lum_equalized = convert_YCbCr_to_rgb(pic_Lum_equalized);
```