Thème Image - TP2 - Utilisation de l'histogramme pour la correction et le traitement d'images

Instructions pour la rédaction du compte-rendu

- Les travaux doivent être remis sur la plateforme chamilo, au plus tard dans une semaine.
- Le travail comprend : un compte-rendu au format pdf, ainsi que les codes scilab.
- Vous déposerez UN UNIQUE FICHIER au format zip qui contiendra le compte rendu et les codes.
- Le nom du fichier doit être de la forme : `NOM1_NOM2_groupe_XX_TP2.zip`
- Soyez attentifs à correctement commenter et indenter le code pour faciliter la lecture.
- De manière générale, soyez attentifs à la présentation des résultats et à la rédaction. Le thème de ce TP est l'image, il est donc obligatoire d'illustrer votre propos avec les images produites.

1 - Histogramme et histogramme cumulé

1.1 - Histogramme

`Scilab` dispose des fonctions `histc(classes, data)` et `histplot(classes, data)` pour calculer et tracer l'histogramme d'un ensemble de données `data` par rapport à des classes de valeurs `classes`. Dans le cas des images, nous voulons compter le nombre de pixels dont la valeur est p, $\forall p \in \{0, \ldots, 255\}$ et il faut l'indiquer à la fonction `histc` en spécifiant correctement `classes` (consultez l'aide de `histc` pour plus de détails). Pour le tester, vous pouvez exécuter les commandes suivantes directement dans la console ou à partir d'un script :

```
exec('init_tp_image.sce');
im = lire_imageBMPgris('papillon.bmp');
afficher_image(im);
// classes pour les valeurs de pixels
classes = [-0.1, linspace(0,255,256)];
// h est un tableau à 256 entrées qui contient l'histogramme
// h(p) = nb de pixels valant p-1
h = histc(im, classes)
// attention cette syntaxe correspond aux versions récentes de scilab
// la syntaxe ancienne est
// h = histc(classes, im, normalization=%f);

// tracé de l'histogramme
scf();
histplot(classes, im, normalization=%f, strf='021');
```

Ici, `h` est un tableau scilab qui contient les valeurs de l'histogramme h ; les tableaux scilab étant indexés à partir de 1, le 1er élément du tableau est `h(1)` qui contient le nombre de pixels de valeur 0. De manière générale, `h(p)` = nombre de pixels valant `p-1`. L'argument optionnel `normalization=%f` indique à la routine `histplot` de ne pas normaliser par le nombre total de pixels de l'image, et `strf='021'` sert à

paramétrer l'affichage.

Exercice 1 :

Tracés d'histogrammes. En vous inspirant de l'exemple précédent, affichez les images suivantes et tracez leurs histogrammes :
`papillon.bmp, fruits.bmp, manoir.bmp, desert.bmp`
Commentez l'aspect général des images et l'allure de leurs histogrammes.

Exercice 2 :

Effet d'une transformation sur l'histogramme.

- Ouvrez et exécutez le fichier `exemple_transformation.sce` pour voir un exemple de transformation et son effet sur l'histogramme d'une image.
- On considère les fonctions de transformation suivantes :

$$f_1(p) = p + 50 \quad ; \quad f_3(p) = 0.5(p - 127) + 127 \quad ; \quad f_5(p) = 2(p - 127) + 127.$$

Modifez le fichier `exemple_transformation.sce` pour appliquer ces transformations à l'image `papillon.bmp`. Affichez les images transformées et leurs histogrammes, et commentez la différence par rapport à l'original. Dans quels cas y a-t-il perte d'information par saturation des valeurs à 0 ou 255 ?

1.2 - Histogramme cumulé

Etant donné l'histogramme $h(p)$ d'une image, l'histogramme cumulé est défini par la formule suivante :

$$H(p) = \sum_{q=0}^{p} h(q), \ \forall p \in \{0, \ldots, 255\}.$$

Il peut également se définir par récurrence :

- $H(0) = h(0)$
- Pour $p \in 1, \ldots, 255, H(p) = H(p-1) + h(p)$.

En pratique, l'histogramme cumulé H est stocké sous la forme d'un tableau à 256 entrées, comme l'histogramme h que vous avez déjà calculé, il faut faire attention aux indices !

Exercice 3 :

Dans le fichier `transformations.sci`, complétez la fonction `hist_cumul` pour qu'elle réalise le calcul de l'histogramme cumulé, puis testez votre fonction comme ci-dessous :

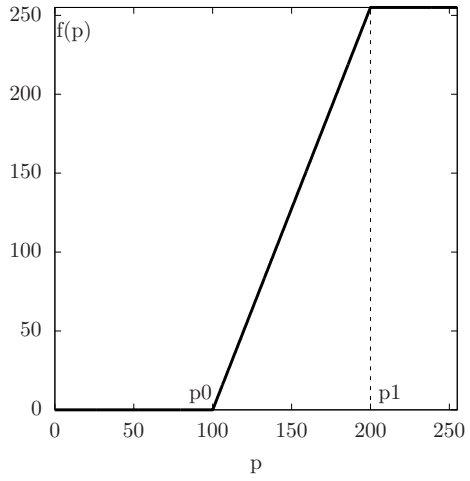
```
exec('transformations.sci', -1)

im = lire_imageBMPgris('fruits.bmp');
// calcul de l'histogramme cumulé
Hist = hist_cumul(im);
// affichage de l'histogramme cumulé
plot2d3(Hist);
```

Quelle est la valeur du dernier élément du tableau `Hist` ? A quoi correspond-elle ?

2 - Histogramme et correction du contraste

Dans cette partie, nous allons voir comment l'histogramme permet de choisir la transformation affine à appliquer à une image donnée.
L'idée est d'identifier l'intervalle $[p_0, p_1]$ de valeurs dans lequel se trouvent la majorité des pixels de l'image puis de répartir ces valeurs sur l'intervalle $[0, 255]$; la transformation correspondante est donnée dans la figure ci-contre.



Exercice 4 :

- Tracez l'histogramme de l'image `desert.bmp`.
- Dans quel intervalle se trouvent la majorité des pixels de l'image ?
- Quelle est la transformation affine qui permet de répartir ces valeurs sur l'intervalle $[0, 255]$?
- Dans le fichier `transformations.sci`, complétez le code de la fonction `f_affine`, puis appliquez cette transformation à l'image grâce aux commandes suivantes :

```
// chargement des transformations
exec('transformations.sci');
im = lire_imageBMPgris('desert.bmp');
p0 = 0;   // à modifier
p1 = 255; // à modifier
im2 = T_affine(im, p0, p1);
```

  Commentez les différences au niveau des images et des histogrammes.

Exercice 5 :

Détermination automatique de p_0 et p_1. Soit I une image de taille MN. Soit $s \in [0, 100]$ une valeur choisie par l'utilisateur. On choisit p_0 (respectivement p_1) de telle manière que $s\%$ des pixels de l'image I aient une valeur plus petite que p_0 (respectivement plus grande que p_1). Autrement dit, soit H l'histogramme cumulé de I, p_0 est la plus petite valeur p telle que $\frac{H(p)}{MN} \geq s/100$, et p_1 est la plus grande valeur p telle que $\frac{H(p)}{MN} \leq 1 - s/100$. Voir la figure ci-dessous pour une illustration de cette méthode.

- Dans le fichier `transformations.sci`, modifiez la fonction `calcul_p0p1` pour déterminer p_0 et p_1.
  Indication : pour p_0, parcourir l'histogramme cumulé H à partir de la valeur de pixel 0 (donc l'indice 1 dans le tableau scilab) ; pour p_1, parcourir H dans le sens inverse (donc à partir de l'indice 256).
- Testez votre code avec l'image `desert.bmp`, par exemple avec un seuil de 1% :

```
// chargement des transformations
exec('transformations.sci');
im = lire_imageBMPgris('desert.bmp');
s = 1;
[p0, p1] = calcul_p0p1(im, s);
```

  Les valeurs de p_0 et p_1 sont-elles proches de celles que vous avez choisies dans l'exercice précédent ? Essayez avec d'autres valeurs du seuil et affichez les images correspondantes.
- Faites de même pour les images `manoir.bmp` et `fruits.bmp`.

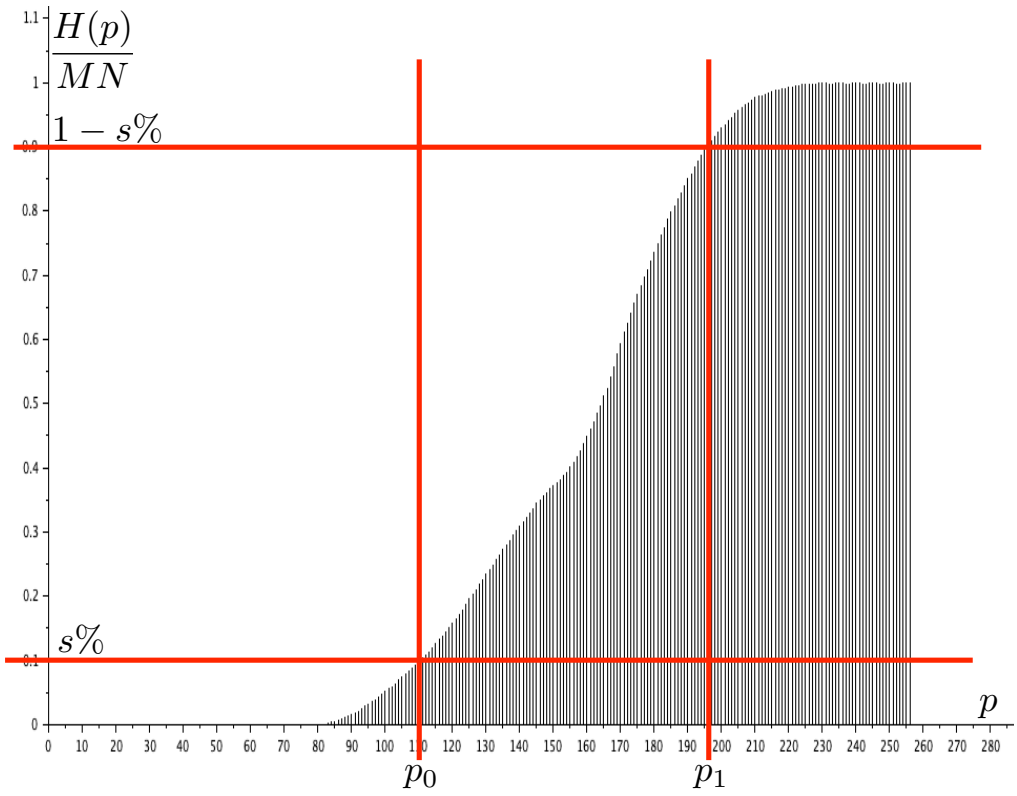


Illustration de la méthode proposée dans l'exercice 5 pour $s = 10$.

3 - Egalisation d'histogramme

Pour une image $I_1(u, v)$ de taille $M \times N$ dont l'histogramme cumulé est $H_1(p)$, l'égalisation d'histogramme consiste à transformer $I_1(u, v)$ en une image $I_2(u, v) = T(I_2(u, v))$ de manière à ce que le nouvel histogramme cumulé H_2 soit le plus linéaire possible. Pour cela, on applique la transformation suivante aux pixels de l'image :

$$T(p) = \frac{255}{MN} H_1(p).$$

Exercice 6 :

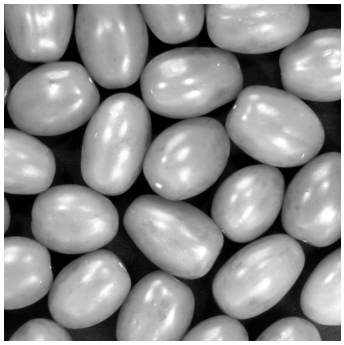
- Complétez le fichier `transformations.sci` pour que la fonction `hist_egal` mette en oeuvre l'égalisation d'histogramme, puis testez-là sur `fruits.bmp` :

```
exec('transformations.sci', -1)
im = lire_imageBMPgris('fruits.bmp');
// calcul de l'image égalisée
im_egal = hist_egal(im);
// affichage
afficher_image(im_egal);
```

  Attention : en SCILAB, les tableaux sont indexés à partir de 1, ce qui fait que si `Hist` est le tableau contenant l'histogramme cumulé $H(p), p \in \{0, \ldots, 255\}$, alors `Hist(1)` désigne son premier élément, c'est-à-dire la valeur $H(0)$.
- Commentez les allures de l'histogramme et de l'histogramme cumulé de la nouvelle image par rapport à ceux de l'image originale. Commentez l'aspect visuel de la nouvelle image. Que pensez-vous du résultat par rapport à la correction affine ?
- Appliquez une deuxième fois l'égalisation d'histogramme. Qu'observez-vous ? Comment l'expliquer ?
- Que pensez-vous de l'égalisation appliquée à l'image `cellules.bmp` ?

4 - Segmentation par seuillage

La segmentation d'une image consiste à identifier des classes de pixels suivant certains critères.



Par exemple dans l'image `tomates.bmp` ci-contre, on constate que le fond est nettement plus sombre que les tomates, et on peut choisir comme critère un seuil d'intensité `s` : un pixel donné appartient à la classe "fond" si sa valeur est inférieure à `s`, et à la classe "tomate" sinon.

Exercice 7 :

- Visualisez l'histogramme de l'image `tomates.bmp` et identifiez une valeur plausible du seuil `s` qui permette de séparer le fond des tomates.
- Dans le fichier `transformations.sci`, modifiez la fonction `seuillage` pour qu'elle applique la transformation suivante :

$$T(p) = \begin{cases} 255 & \text{si} \quad p > s, \\ 0 & \text{sinon.} \end{cases}$$

- Vous pouvez la tester de la façon suivante, en faisant varier le seuil `s` autour de la valeur que vous avez évaluée précédemment :

```
im = lire_imageBMPgris('tomates.bmp');
afficher_image(seuillage(im, s));
```

  Dans le même esprit que le TP précédent, vous pouvez créer un masque puis utiliser la fonction `produit` (qui vous est fournie dans `transformations.sci`) pour afficher les tomates sans le fond, ou le fond sans les tomates :

```
masque = seuillage(im, s);
afficher_image(produit(masque, im));
afficher_image(produit(255 - masque, im));
```

- Il y a un message caché dans le fond de l'image `tomates.bmp`, saurez-vous le rendre clairement visible ?