Université de Marne-La-Vallée
Institut Gaspard MONGE

Master d'Informatique **Module de Traitement d'Images**

ITI

CORRIGÉ

EXAMEN

Année 2012-2013

On répondra directement sur les feuilles d'examen en indiquant en pied de page ses NOM et Prénom. L'usage de documents n'est pas autorisé.

1. Stretching par saturation

Soit une image en entrée [r(i,j), i=0..(M-1), j=0..(N-1)] de 16 bits non signés et possédant un background valant 0. On désire produire en sortie une image en 8 bits non signés r'(i,j), étirée (« *stretchée* ») linéairement entre les valeurs a et b correspondant respectivement à 1% des données saturées par défaut et 1% des données saturées par excès.

1

a. Quelle est l'intervalle des valeurs permises dans l'image en entrée ?

$\forall$ i=0..(M-1), j=0..(N-1), r(i,j) $\in [0,2^{16}-1]$ = **[0 , 65535]**

Quelle est l'intervalle des valeurs des pixels image dans l'image en entrée ?

[1 , 65535]

1

b. Quelle est l'intervalle des valeurs permises dans l'image en sortie ?

$\forall$ i=0..(M-1), j=0..(N-1), r'(i,j) $\in [0,2^{8}-1]$ = **[0 , 255]**

Quelle est l'intervalle des valeurs des pixels image dans l'image en sortie ?

[1 , 255]

Combien y-a-t'il de pixels de background dans l'image de sortie ?

On trouvera le même nombre de pixels de background dans l'image en sortie que dans l'image en entrée.
H(0) = H'(0)

1

c. Calcul de l'histogramme.

Soit la déclaration des variables qu'on complètera

```
unsigned char    *origin_image[3];   /* image array: ORIGIN IMAGE */
unsigned char    *processed_image[3];/* image array: PROCESSED IMAGE */
int              nliin;              /* input line number */
int              npxin;              /* input pixel number */
int              ichannel;           /* index among channels */
int              ili;                /* index among lines */
int              ipx;                /* index among pixels */
int              histo[3][65536] ;
```

1

Compléter le programme en langage C pour calculer l'histogramme des trois images rouge, verte et bleue.

```
...
for (ichannel=0; ichannel<channel_number; ichannel++)
   for (i=0 ; i<65536 ; i++)
      histo[ichannel][i] = 0 ;
...
for (ichannel=0; ichannel<channel_number; ichannel++)
   for (ili=0 ; ili<nliin ; ili++)
      for (ipx=0 ; ipx<npxin ; ipx++)
         histo[ichannel][origin_image[ichannel][ili*npixin+ipx]] =
            histo[ichannel][origin_image[ichannel][ili*npixin+ipx]] + 1 ;
```

NOM : …… Prénom : ……

d. Ecrire l'extrait du programme en langage C permettant de calculer pour chaque bande (channel) le nombre de pixels saturés à 1% à partir de l'histogramme calculé en question c. **1**

On n'oubliera pas de déclarer les variables utilisées.

```
int              nb_total[3];        /* nombre total de pixels image */
int              nb_satur[3];        /* nombre de pixels saturés */
...
for (ichannel=0; ichannel<channel_number; ichannel++)
{
   nb_total[ichannel] = 0 ;
   for (i=1 ; i<65536 ; i++)
      nb_total[ichannel] = nb_total[ichannel] + histo[ichannel][i];
   nb_satur[ichannel] = 0.01 * nb_total[ichannel] ;
}
```

e. Ecrire l'extrait du programme en langage C permettant de calculer pour chaque bande (channel) la borne gauche du stretching pour une saturation par défaut à 1%. On n'oubliera pas de déclarer les variables utilisées. **1**

```
int              a[3];               /* borne gauche du stretching */
...
for (ichannel=0; ichannel<channel_number; ichannel++)
{
   a[ichannel] = histo[ichannel][1];
   i = 2;
   while (a[ichannel] < nb_satur[ichannel])
   {
      a[ichannel] = a[ichannel] + histo[ichannel][i];
      i = i + 1 ;
   ]
}
```

f. Ecrire l'extrait du programme en langage C permettant de calculer pour chaque bande (channel) la borne droite du stretching pour une saturation par défaut à 1%. On n'oubliera pas de déclarer les variables utilisées. **1**

```
int              b[3];               /* borne droite du stretching */
for (ichannel=0; ichannel<channel_number; ichannel++)
{
   b[ichannel] = histo[ichannel][65535];
   i = 65534;
   while (b[ichannel] < nb_satur[ichannel])
   {
      b[ichannel] = b[ichannel] + histo[ichannel][i];
      i = i - 1 ;
   ]
}
```

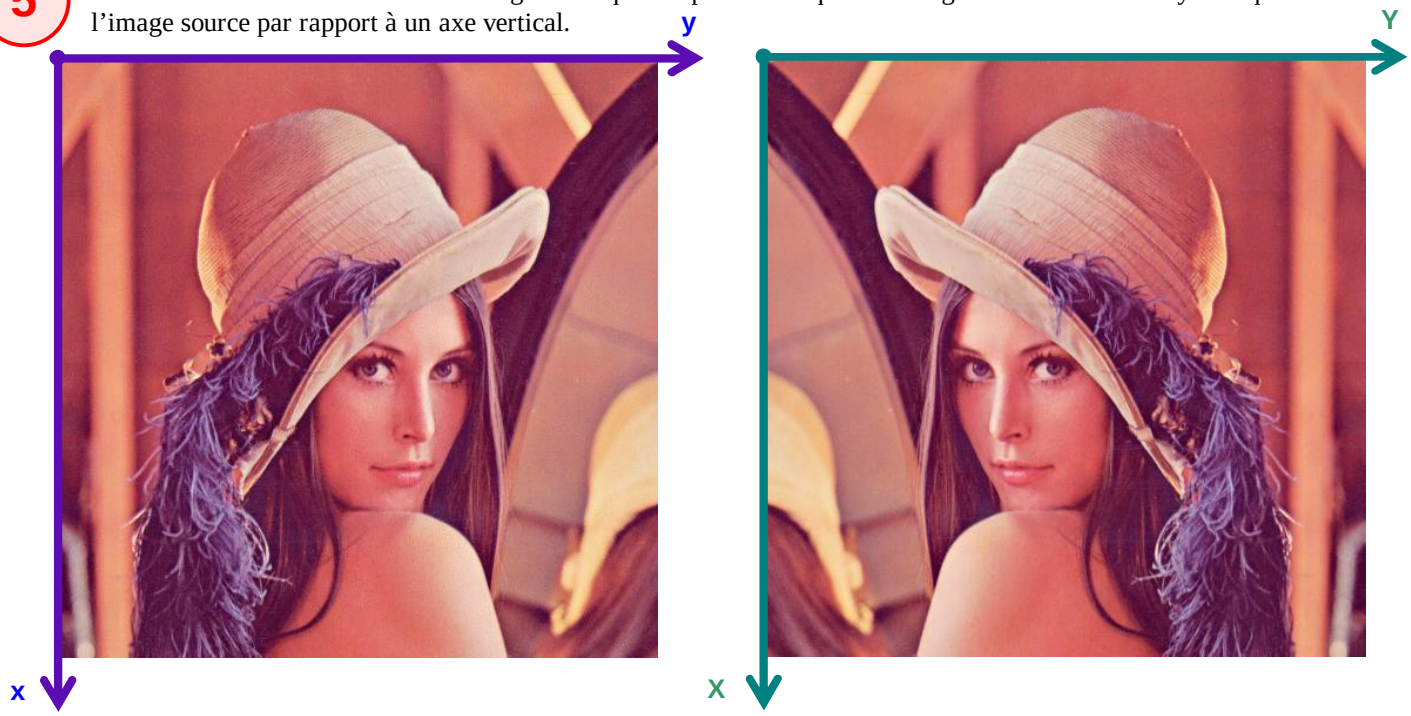
g. Ecrire l'extrait du programme en langage C permettant d'initialiser une LUT pour réaliser le stretching entre les deux bornes calculées précedemment. On n'oubliera pas de déclarer les variables utilisées. **1**

```
int              lut[3][65536];      /* LUT du stretching */
for (ichannel=0; ichannel<channel_number; ichannel++)
{
   lut[ichannel][0] = 0 ;
   for (i=1 ; i<65536 ; i++)
   {
      lut[ichannel][i] =
         1 + round(255. * (i - a[ichannel]) / (b[ichannel] - a[ichannel]);
      if (lut[ichannel][i] < 1)
         lut[ichannel][i] = 1;
      if (lut[ichannel][i] > 255)
         lut[ichannel][i] = 255;
}
```

2. Miroir vertical

5

On désire réaliser une transformation géométrique simple dans laquelle l'image destination est le symétrique de l'image source par rapport à un axe vertical.



Ecrire le code en langage C réalisant cette transformation. On se limitera à la partie des boucles de calcul de la processed_image (partie appelée « PROCESSING SECTION » dans le code de « skelet.c »).

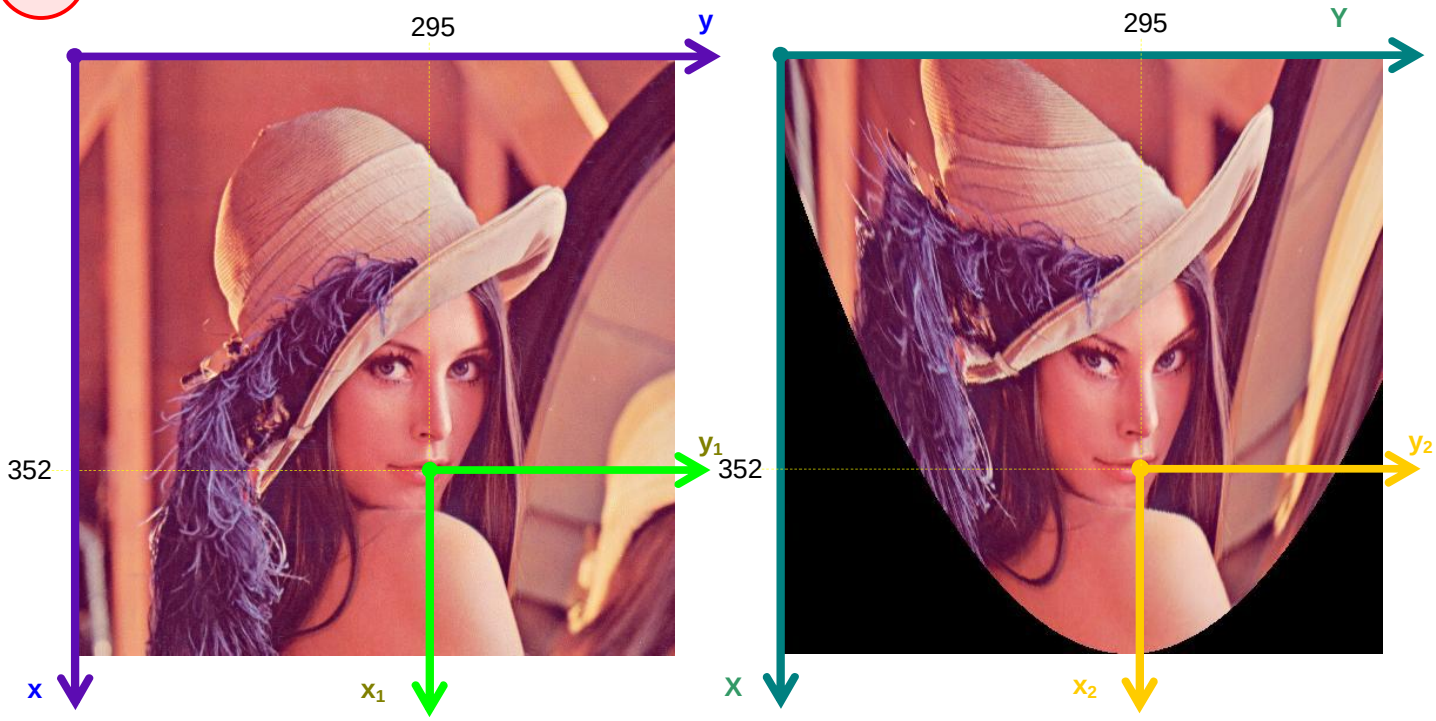
On veillera à tester la position de l'antécédent dans l'image source pour affecter la valeur de background en vert.

```
for (ichannel=0; ichannel<channel_number; ichannel++)
{
   for (ili=0; ili<nliin; ili++)
   {
      for (ipx=0; ipx<npxin; ipx++)
      {
         x = ili;
         y = 511 - ipx;

         iround = (int)(x + 0.5);
         jround = (int)(y + 0.5);

         if ((iround < 0)          ||
             (iround >= nliin)     ||
             (jround < 0)          ||
             (jround >= npxin))
         {
            if (ichannel == 1)
               processed_image[ichannel][ili*npxin+ipx] = 255;
            else
               processed_image[ichannel][ili*npxin+ipx] = 0;
         }
         else
         {
            processed_image[ichannel][ili*npxin+ipx] =
               origin_image[ichannel][iround*npxin+jround];
         }
      } /* Loop on pixels */
   } /* Loop on lines */
} /* Loop on channels */
```

3. Léna parabolique

7

Pour forcer le sourire de Léna, on applique une transformation parabolique centrée sur la bouche de Léna au point de coordonnées (352,295).



Montrer la séquence des trois transformations géométriques permettant de –centrer sur la bouche, -calculer la parabole, -repositionner l'origine de l'image.

Pour les transformations, on décrira les trois repères, les équations des modèles de déformation directs et inverses, puis on illustrera le schéma de déformation par un croquis.

On rappelle que l'expression canonique de la parabole est donnée par l'équation : $Y = A.x^2$. Dans les équations, on retrouvera la valeur la plus précise possible du gain A.

MDD	$\begin{cases} x_1 = x - 352 \\ y_1 = y - 295 \end{cases}$	$\begin{cases} x_2 = x_1 - 0.005 \times y_1^2 \\ y_2 = y_1 \end{cases}$	$\begin{cases} X = x_2 + 352 \\ Y = y_2 + 295 \end{cases}$
MDI	$\begin{cases} x = x_1 + 352 \\ y = y_1 + 295 \end{cases}$	$\begin{cases} x_1 = x_2 + 0.005 \times y_2^2 \\ y_1 = y_2 \end{cases}$	$\begin{cases} x_2 = X - 352 \\ y_2 = Y - 295 \end{cases}$