**ESTIMATION DE LA LUMINOSITÉ D'UNE ÉTOILE VARIABLE**
à partir d'une photographie.

Exercices préparatoires : additionner les pixels d'une étoile avant et après filtrage et application d'un masque. Explications sans code Python puis avec code Python.

L'image utilisée pour les manipulations est la même que celle utilisée dans les deux supports précédents (estimations06.pdf et estimation07.pdf)

## *1. Intention de la manipulation*

Dans l'idéal, je voudrais :
- Compter combien d'étoiles sont présentes sur le cliché.
- Additionner toutes les valeurs des pixels, étoile par étoile, (ce que je désignerai par la suite par "le poids" et elle "pèse") de façon à pouvoir dire :
Si A "pèse" 1240 et B "pèse" 960, alors la magnitude de A est moins grande que celle de B.

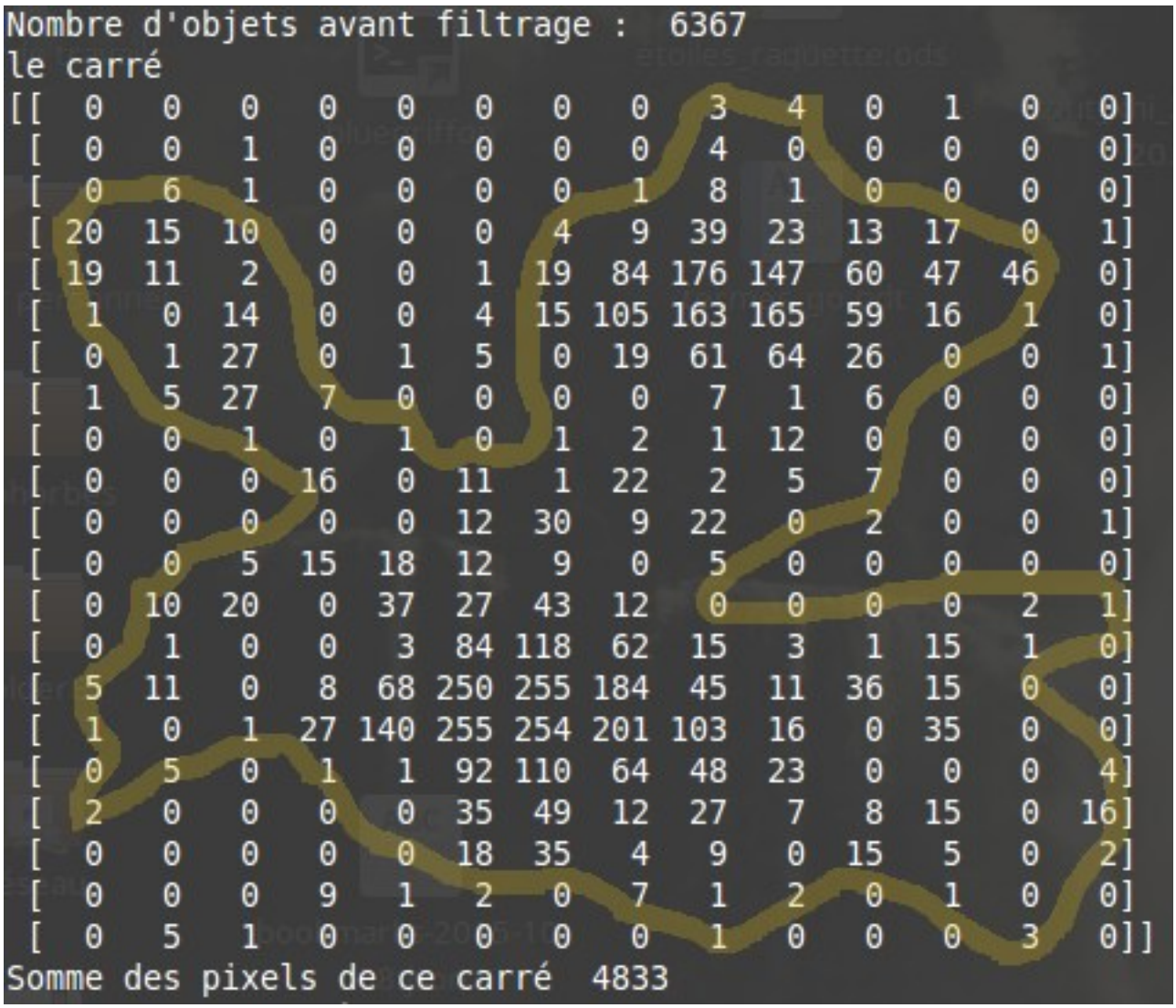
## *2. Première manipulation "magique"*

Voici une capture du résultat.
Le programme a identifié 6367 objets (potentiellement étoiles) dans le cliché.
Les nombres inscrits dans la zone délimitée par des crochets sont les valeurs numériques des pixels de l'objet numéro 7.
La somme des pixels de cet objet est 4833. En d'autres termes, l'objet pèse 4833.

```
Nombre d'objets avant filtrage :  6367
le carré
[[  0   0   0   0   0   0   0   0   3   4   0   1   0   0]
 [  0   0   1   0   0   0   0   0   4   0   0   0   0   0]
 [  0   6   1   0   0   0   0   1   8   1   0   0   0   0]
 [ 20  15  10   0   0   0   4   9  39  23  13  17   0   1]
 [ 19  11   2   0   0   1  19  84 176 147  60  47  46   0]
 [  1   0  14   0   0   4  15 105 163 165  59  16   1   0]
 [  0   1  27   0   1   5   0  19  61  64  26   0   0   1]
 [  1   5  27   7   0   0   0   0   7   1   6   0   0   0]
 [  0   0   1   0   1   0   1   2   1  12   0   0   0   0]
 [  0   0   0  16   0  11   1  22   2   5   7   0   0   0]
 [  0   0   0   0   0  12  30   9  22   0   2   0   0   1]
 [  0   0   5  15  18  12   9   0   5   0   0   0   0   0]
 [  0  10  20   0  37  27  43  12   0   0   0   0   2   1]
 [  0   1   0   0   3  84 118  62  15   3   1  15   1   0]
 [  5  11   0   8  68 250 255 184  45  11  36  15   0   0]
 [  1   0   1  27 140 255 254 201 103  16   0  35   0   0]
 [  0   5   0   1   1  92 110  64  48  23   0   0   0   4]
 [  2   0   0   0   0  35  49  12  27   7   8  15   0  16]
 [  0   0   0   0   0  18  35   4   9   0  15   5   0   2]
 [  0   0   0   9   1   2   0   7   1   2   0   1   0   0]
 [  0   5   1   0   0   0   0   0   1   0   0   0   3   0]]
Somme des pixels de ce carré  4833
```

Critique du résultat : la forme qui se dessine ne ressemble pas à une étoile.

J'ai dessiné une forme qui englobe les pixels de valeur > 0, cela donne :



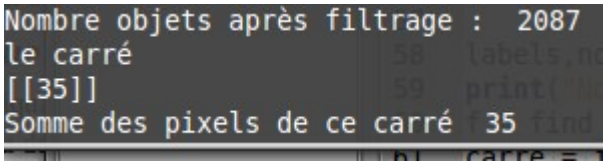

En fait l'objet identifié ne se limite pas à une seule étoile. Par l'intermédiaire de pixels de faible intensité, les étoiles se sont trouvées reliées et confondues en un seul objet.
Si l'on veut identifier chacune d'elles, il faut éliminer ces ponts plus ou moins ténus et donc filtrer les valeurs faibles.
Cela va être l'occasion d'utiliser à nouveau un masque.

### *3. Le principe du masque*

Un masque est obtenu en appliquant un seuil sur un tableau de données.
Ensuite on multiplie le tableau de données par le masque.
Le résultat attendu étant : tous les pixels dont la valeur est inférieure à un seuil donné sont mis à zéro (noir).
Reprenant notre illustration précédente, on pourrait fixer 30 comme seuil.

### *4. Application du seuil*

Un grand nombre d'objets faibles a été éliminé.
L'objet numéro 7 a été ramené à pas grand chose.
Mais rien ne prouve que c'est le même objet.



## *5. Le code Python*

J'ai trouvé les explications nécessaires à partir de cette page, en anglais malheureusement :
http://docs.scipy.org/doc/scipy/reference/tutorial/ndimage.html
*L'intention de mes notes successives étant d'ailleurs de présenter, en français, l'usage de certaines fonctions.*
Je vais insister sur les points délicats de la syntaxe,  c'est à dire sur ce qui m'a demandé plus de temps d'expérimentation et de lecture, l'ensemble du code étant fourni à la fin du document.

Pour pouvoir trouver, et additionner les pixels, il faut charger les outils :

```
from scipy.ndimage import find_objects,label,sumfrom scipy.ndimage import
find_objects,label,sum
from numpy import *
```

La fonction label() retourne le nombre d'objet ainsi qu'une liste permettant de localiser les objets.
f =find(labels) crée un tableau **exploitable** des positions.

```
labels,nombre_objets = label(im)
print("Nombre d'objets avant filtrage : ",nombre_objets)
f = find_objects(labels)
```

Pour récupérer tous les pixels de l'objet 7 de l'image im, il faut utiliser cette syntaxe :
im[f[7]]  à partir du tableau f créé précédemment.
**Ce point de syntaxe est extrêmement important** : j'ai passé des heures à le comprendre.
Ayant récupéré les pixels, on peut les additionner. Ce code n'est pas optimisé mais il se veut didactique.

```
carre = im[f[7]]
print("le carré ")
print(carre)
print("Somme des pixels de ce carré ",sum(carre))
```

Pour mémoire (déjà présenté dans un autre support) : la création du masque par :
seuil =30
msq = 1*(im>seuil)
Puis la multiplication du tableau de nombres par le masque :i m2 = im * msq

```
seuil =30
msq = 1*(im>seuil)
im2 = im * msq
```

Et maintenant le code entier. Les remarques sont en rouge.

```
# -*- coding:Utf-8 -*-
"""
Notes prises à partir du tutorial
file:///media/alain/9016-4EF8/documentations/scipy-html-0.16.1/tutorial/ndimage.html  à
la rubrique "Objects measurements"
Calculer la somme des pixels pour chaque objet identifié.
Idem après filtrage.  """

from PIL import Image
from scipy.ndimage import find_objects,label,sum
from numpy import *
im = array(Image.open("TCas_a_cliquer2.jpg").convert("L"))

labels,nombre_objets = label(im)
print("Nombre d'objets avant filtrage : ",nombre_objets)
f = find_objects(labels)

"""
find_objects returns slices for all objects, unless the max_label parameter is
larger then zero, in which case only the first max_label objects are returned.
exemple : f = find_objects(liste,max_label=3)
A partir de ceci, on peut récupérer les valeurs de l'objet num par
un_objet =im[f[num]]
Et ainsi, obtenir la somme par sum(un_objet) """

carre = im[f[7]]
print("le carré ")
print(carre)
print("Somme des pixels de ce carré ",sum(carre))

"""Nombre d'objets avant filtrage :  6367
le carré
[[  0   0   0   0   0   0   0   0   3   4   0   1   0   0]
 [  0   0   1   0   0   0   0   0   4   0   0   0   0   0]
 [  0   6   1   0   0   0   0   1   8   1   0   0   0   0]
 [ 20  15  10   0   0   0   4   9  39  23  13  17   0   1]
 [ 19  11   2   0   0   1  19  84 176 147  60  47  46   0]
 [  1   0  14   0   0   4  15 105 163 165  59  16   1   0]
 [  0   1  27   0   1   5   0  19  61  64  26   0   0   1]
 [  1   5  27   7   0   0   0   0   7   1   6   0   0   0]
 [  0   0   1   0   1   0   1   2   1  12   0   0   0   0]
 [  0   0   0  16   0  11   1  22   2   5   7   0   0   0]
 [  0   0   0   0   0  12  30   9  22   0   2   0   0   1]
 [  0   0   5  15  18  12   9   0   5   0   0   0   0   0]
 [  0  10  20   0  37  27  43  12   0   0   0   0   2   1]
 [  0   1   0   0   3  84 118  62  15   3   1  15   1   0]
 [  5  11   0   8  68 250 255 184  45  11  36  15   0   0]
 [  1   0   1  27 140 255 254 201 103  16   0  35   0   0]
```

```
 [  0   5   0   1   1  92 110  64  48  23   0   0   0   4]
 [  2   0   0   0   0  35  49  12  27   7   8  15   0  16]
 [  0   0   0   0   0  18  35   4   9   0  15   5   0   2]
 [  0   0   0   9   1   2   0   7   1   2   0   1   0   0]
 [  0   5   1   0   0   0   0   0   1   0   0   0   3   0]]
Somme des pixels de ce carré  4833
"""
```

```
seuil =30
msq = 1*(im>seuil)
im2 = im * msq

labels,nombre_objets = label(im2)
print("Nombre objets après filtrage : ",nombre_objets)
f = find_objects(labels)
carre = im2[f[7]]
print("le carré ")
print(carre)
print("Somme des pixels de ce carré ",sum(carre))
```