# ESTIMATION DE LA LUMINOSITÉ D'UNE ÉTOILE VARIABLE

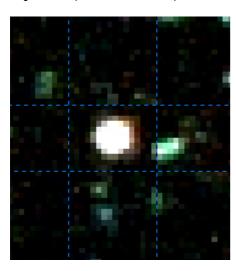
à partir d'une photographie.

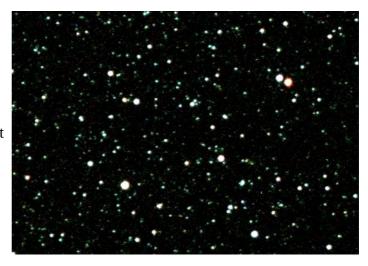
Exercices préparatoires : opérations numériques sur plusieurs tableaux de nombres

# 1. L'image de départ

L'image ci-contre est utilisée pour les exemples de traitements numériques qui suivent.

Dans cette image, une zone définie par ses coordonnées va être systématiquement explorée. (voir ci-dessous)





La zone concernée est limitée par le rectangle en pointillés. En utilisant la syntaxe du langage Python la zone est définie par : ligne du dessus, ligne du dessous, colonne de gauche, colonne de droite.

Ce qui amène aux variables suivantes : |i1,|i2,co1,co2| = 74,82,118,129

# 2. Bibliothèques et chargement de l'image sous forme de trois tableaux de nombres

from PIL import Image
from pylab import \*
im = array(Image.open("TCas a cliquer2.jpg")

im	Nom symbolique à utiliser par la suite pour désigner l'image chargée dans la mémoire
array	Transforme en tableau à deux dimensions (hauteur, largeur)
lmage.open	Tourne le robinet du "tonneau" qui contient l'image

Remarque : le code du programme est listé à la fin de cette note

### 3. Obtenir des informations sur la structure du fichier

L'information obtenue par programme est la suivante :

Dimensions (304, 433, 3) et type de données : uint8

Hauteur : 304 Largeur : 433 Nombre de couches : 3

#### **Commentaires:**

Les dimensions sont exprimées en nombre de pixels et non en millimètres ou toute autre mesure de longueur usuelle.

Nombre de couches : l'image couleur est contenue dans 3 tableaux de données à deux dimensions.

uint8 : vient de Unsigned Integer (nombre entier sans signe) 8 (sur 8 bits = 256 valeurs possibles).

# 4. Lister les valeurs des octets dans la zone |i1,|i2,co1,co2

Par rapport au support précédent (*estimation06.pdf*) on va lister la même zone pour trois couleurs : le rouge, le vert, le bleu.

-	-	100										
Pour le rouge												
]]	1	31	49	69	105	131	102	96	65	50	Θ]	
[	1	23	94	197	255	255	255	199	111	59	2]	
[	Θ	39	236	254	254	255	254	255	156	57	27]	
[	15	115	255	255	255	254	254	255	173	75	10]	
]	34	112	255	255	255	255	255	255	195	90	20]	
[	0	34	207	254	255	255	255	254	135	67	38]	
[	0	21	57	200	255	255	219	113	97	68	16]	
[	16	20	16	68	103	100	54	59	52	25	0]]	

Si vous regardez une même zone sur chacune des captures d'écran suivantes, vous remarquerez que les intensités (valeur des pixels) varient pour une même cellule.

40 36 1] 28 65 102 123 77 31 1] 19 1] 254 254 32 0] 255 255 255 255 159 0] 255 255 255 20] 255 255 209 209 98 51 2] 16 0] 72 105 97 51 50

La somme des pixels pour chacune de ces trois zones exprime aussi ces différences.

Rouge : 11402 Vert : 10776 Bleu : 9950

Pour le bleu 0] 33 56 87 100 46 33 12 0] 7 84 188 255 255 215 131 2] 29 240 254 254 255 255 255 0 255 254 254 13 255 255 255 255 255 33 18 10 254 255 255 255 252 192 255 255 197 91 71 44 55 84 80 36 41 11

En astronomie savante, il existe un mode de classification des étoiles appelé B -V ou indice de couleur.

L'ébauche d'un classement automatique ? Amusant.

# 5. Et si l'on additionnait les trois tableaux de valeur ?

Cela peut paraître idiot et la première question qui vient est : à quoi cela sert-il ? Pour répondre je vais rappeler deux choses :

• L'estimation de la magnitude de T Cas était peut-être erratique parce que l'indice de couleur

- venait fausser les choses. Si l'on réunit tous les pixels ensemble on aura au moins éliminé cette cause d'erreur.
- Pour individualiser chacune des étoiles, je vais avoir besoin d'un masque (voir estimation06.pdf). Mais un masque établi à partir de quel tableau de couleur : le rouge, le vert, le bleu ? Et si l'on choisit celui-ci, comment défendre ce choix ?
   Le choix fait ici est de créer le masque à partir de la somme des couleurs, avec l'hypothèse qu'ainsi on garde tout.

#### 6. Visualisation de la somme

La façon dont on parvient au résultat, via Python, est amusante (si on aime ce genre d'amusement). En attendant, voici le résultat :

```
[[ 3 72 122 190 294 354 225 196 122 93 1]
[ 3 53 269 577 765 765 719 499 234 97 3]
[ 5 119 718 762 762 765 764 765 365 76 30]
[ 38 347 765 765 765 762 762 764 417 120 10]
[ 74 329 763 765 765 765 765 765 491 175 21]
[ 0 88 632 762 765 765 765 760 334 124 68]
[ 0 56 171 601 765 765 625 302 246 163 19]
[ 41 50 41 195 292 277 141 150 133 52 0]]
```

La conclusion, le code Python et les commentaires qui s'y rapportent sont dans les pages suivantes.

#### Définition du codage et ouverture des bibliothèques nécessaires

```
# -*- coding:Utf-8 -*-
from PIL import Image
from pylab import *
```

Chargement de l'image couleur et définition de la zone témoin qui sera listée

```
im = array(Image.open("TCas_a_cliquer2.jpg"))
li1,li2,co1,co2 = 74,82,118,129 # coordonnées de la zone à lister
```

Définition d'une fonction qui affichera les informations sur l'image. Elle sera appelée plus loin par : dimensions=infos sur image(im)

```
def infos_sur_image(im):
    print("Dimensions {} et type de données : {} ".format(im.shape,im.dtype))
    h,l,n =im.shape[0:3] # façon pythonesque de récupérer les valeurs
    print("Hauteur : {} ".format(h))
    print("Largeur : {} ".format(l))
    print("Nombre de couches : {} ".format(n))
    nb = im.shape[0:2] #
    return nb
```

Définition d'une fonction qui imprime les valeurs des pixels contenus dans la zone li1,li2,co1,co2 Elle sera appelée par : imprime couche(li1,li2,co1,co2,i)

```
def imprime_couche(li1,li2,co1,co2,num):
    tcoul=("rouge","vert","bleu")
    print("Pour le {}".format(tcoul[num]))
    print(im[li1:li2,co1:co2,num])
    print("Somme {} et moyenne {}
".format(im[li1:li2,co1:co2,num].sum(),im[li1:li2,co1:co2,num].mean()))
```

Création du tableau qui contiendra la somme des pixels des trois couleurs. On récupère les dimensions de l'image puis on crée un tableau contenant des zéros.

```
dimensions=infos_sur_image(im)
cible = np.zeros((dimensions[0], dimensions[1]), dtype=np.int32)
```

Remarque importante : le tableau que l'on crée doit avoir des cellules assez grandes pour contenir la somme de trois pixels. Si l'on avait conservé la taille initiale uint8, qui ne peut pas dépasser le nombre 255, cela ne suffisait pas. J'ai choisi le type entier sur 32 bits qui dépasse de beaucoup les besoins.

Le tableau est rempli de zéros.

# La boucle principale.

Il y a trois couleurs. On va parcourir chacune successivement de façon à :

- lister la zone concernée (voir captures sur fond gris plus haut dans le support),
- additionner le tableau cible avec chacun des trois tableaux de pixels.

La syntaxe Python pour la boucle est for i in range(3):

```
for i in range(3):
   imprime_couche(li1,li2,co1,co2,i)
   cible = cible + im[:,:,i]
```

Listage des pixels de la zone li1,li2,co1,co2 pour le tableau qui contient la somme.

```
print(cible[li1:li2,co1:co2])
```

#### 7. Conclusion

Nous avons pu ouvrir une image JPEG couleur et accéder individuellement à chacun des tableaux à deux dimensions qui contiennent les valeurs des pixels.

La syntaxe im[:,:,i] est à retenir. (on y reviendra).

Il a été possible de créer un tableau qui contient la somme des valeurs des pixels.

Ce tableau va être utilisé pour créer un masque...

Mais cela, c'est la suite.