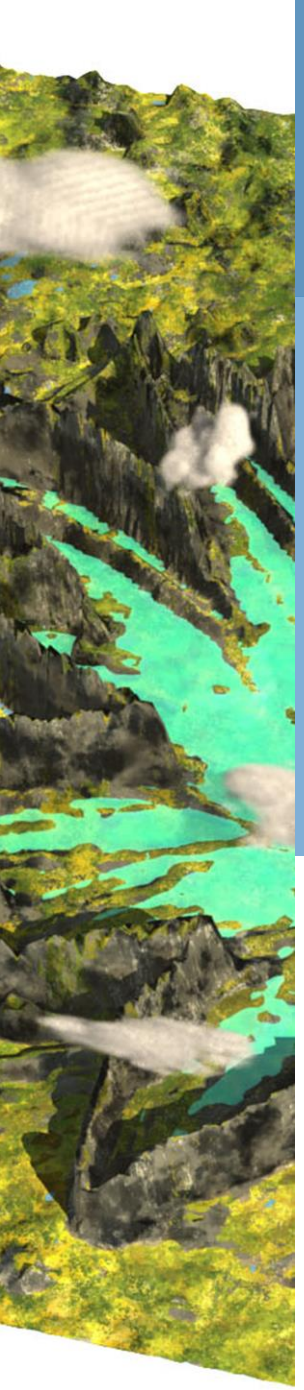

Cours de segmentation d'images

TP1 – Prise en main

Master M2TI - Paris V

John Chaussard
LAGA – Université Paris 13
chaussard@math.univ-paris13.fr

Les outils que l'on utilisera

Langage de programmation

Tout au long de ces TPs, nous utiliserons comme langage de programmation

Plusieurs versions existent, il faudrait utiliser la **version 2.7** de préférence (la plus stable)…

Python peut être installé sous Linux, Windows et Mac sans (trop de) soucis (pour Mac, il faut passer par un outil appelé MacPorts)

Editeur de code

Pour programmer en Python, l'éditeur de code PyCharm est très pratique

L'éditeur existe pour Linux, Windows et Mac.

Les modules supplémentaires

Enfin, il est possible de télécharger des modules supplémentaires pour Python, pour gérer des matrices comme dans Matlab, ou manipuler des images…

Nous utiliserons **deux modules importants** :

Pour les installer, il suffit d'utiliser le gestionnaire de modules **pip**, qui est généralement installé avec Python (et accessible depuis l'interface graphique de PyCharm).

Exemple de code

Les bases en Python

Tout programme Python s'écrit dans un fichier .py.

Par exemple, **créez le fichier test.py**

`test.py`

Le fichier ne doit contenir **aucun accent**.

Pour **écrire un commentaire**, commencez la ligne avec un dièse (#).

Un hello world en Python

Tout programme Python doit contenir, pour être exécuté, une fonction main qui s'écrit de la sorte :

```
if __name__ == "__main__":
    #Insérer le code de la fonction après
```

Après chaque déclaration de fonction, il faut nécessairement écrire une tabulation. Pareil après un if, for ou while : **l'indentation du code est primordiale**.

Votre code ne s'exécutera pas s'il n'est pas correctement indenté.

Nous allons demander, dans la fonction, à Python d'afficher « Bonjour ».

```
if __name__ == "__main__":
    print('Bonjour')
```

Une fois le code écrit, sauvegardez-le, puis dans le terminal, dans le dossier où se trouve le fichier, écrivez

```
python test.py
```

Manipuler des images

Importer les bons package

Pour manipuler des images, il faudra importer dans votre code les bons packages : OpenCV et NumPy.

Pour ce faire, en début de fichier, écrivez

```
import cv2
import cv
import numpy as np
```

Si vous êtes sous Mac, il faudra probablement écrire, avant ces import :

```
import sys
sys.path.append('/usr/local/lib/python2.7/site-packages')
```

Ouvrir et afficher une image

Récupérez, parmi les fichiers du TP1, l'image **perr.jpg**

Pour l'ouvrir, écrivez dans la fonction principale :

```
image = cv2.imread('perr.jpg')
```

Pour afficher l'image, écrivez ensuite :

```
#On affiche l'image dans une fenetre
cv2.imshow("Image", image)
#On attend que l'utilisateur appuie sur une touche
key = cv2.waitKey(0)
#On detruit la fenetre
cv2.destroyAllWindows()
```

Une fenêtre s'affiche avec une image, et le programme se met en pause. Il faudra activer la fenêtre en cliquant dessus et appuyer sur une touche du clavier pour fermer la fenêtre et reprendre l'exécution du code.

Ecrire une image, générer des images

Pour écrire une image sur le disque, dans le fichier **out.png** (par exemple), on fera :

```
cv2.imwrite('out.png', image)
```

Enfin, pour générer une image vide, on fera

```
imagevide = np.zeros((300, 200, 3), np.uint8)
```

Ici, on créé une image de 200 pixels de large sur 300 de haut, avec trois canaux couleur, chacun codé sur 8 bits, remplie de zéros.

```
imageun = np.ones ((300, 200, 3), np.uint8)
```

Si on souhaite générer une image remplie de 1:

Ma première fonction Python

Il sera utile d'afficher souvent des images. On va donc faire une fonction qui prendre en paramètre une image et s'occupera de l'afficher:

```
#Fonction permettant d'afficher une image
def afficher_image(img):
    #On affiche l'image dans une fenetre
    cv2.imshow("Image", img)
    #On attend que l'utilisateur appuie sur une touche
    key = cv2.waitKey(0)
    #On detruit la fenetre
    cv2.destroyAllWindows()
```

Ce qui fait que la fonction principale se réduit à

```
if __name__ == "__main__":
    image = cv2.imread('perr.jpg')
    afficher_image(image)
```

Dimensions de l'image

On peut récupérer les dimensions de l'image avec

```
largeur = image.shape[1]
print ('Largeur image : ' + str(largeur) + ' pixels')

hauteur = image.shape[0]
print('Hauteur image : ' + str(hauteur) + ' pixels')
```

Le symbole + sert à coller ensemble les chaînes des caractères.

A votre tour : affichez le nombre de pixels total de l'image

Lire un pixel

On peut lire la valeur d'un pixel en faisant :

```
pix = image[200, 300]
print ('Valeur du pixel de coordonnee (300, 200) (format BVR) : ')
print(pix)
```

Que vous affiche le résultat ? Qu'en déduisez-vous concernant le nombre total de valeurs stockées dans le tableau image ?

Ecrire un pixel

On peut modifier la valeur d'un pixel en lui attribuant directement le triplet de valeurs qui doit le remplacer (dans le cas des images couleur) :

```
#On modifie la valeur du pixel (0,0)
image[0,0] = [0,0,255]
```

On peut aussi ne modifier que la valeur d'une composante de couleur :

```
#On modifie la composante bleue du pixel (3,11)
image[11,3,0] = 255
```

A votre tour : Modifiez en vert le pixel de coordonnées (7,8), puis placez à 255 la composante rouge du pixel (8,8).

Affichez l'image : voyez-vous vos modifications ?

Ecrire plusieurs pixel

On peut, comme en Matlab, agir sur toute une partie de l'image à la fois. Par exemple pour colorier en rouge la ligne 30 de l'image, du pixel 50 au pixel 90, on fera :

```
#On modifie en rouge une partie de la ligne 30 de pixels
image[30,50:91] = [0,0,255]
```

Tiens ! Pourquoi a-t-on marqué 91 ?

On peut modifier l'ensemble d'une ligne :

```
#On modifie en rouge toute la ligne 80 de pixels
image[80,:] = [0,0,255]
```

A votre tour : Modifiez en bleu la colonne 60, du pixel 20 au pixel 90 exclu.

Affichez l'image : voyez-vous vos modifications ?

Modifiez le carré de coin (250,130) au coin (290,170) en y multipliant toutes les valeurs par 2/5.

Que voyez-vous sur l'image ?