TP 2: Images numériques

Juillet 2019
1.0

Légende

 Entrée du glossaire

 Abréviation

 Référence Bibliographique

 Référence générale

Table des matières

Objectifs 4

Introduction 5

I - **Utilisation rapides des fonctions de base de traitement d'images avec OpenCv** 6

1. La représentation d'une image en C/C++ 6

2. Le type IplImage défini par OpenCV 7

3. La création d'image 7

4. Chargement et affichage d'un fichier image 8

5. Exercice 9

6. Exercice 10

7. Manipuler des images en couleur 10

8. Exercice 11

II - **Énoncé de TP2** 12

III - **Conclusion** 13

Solutions des exercices 14

Webographie 15

Objectifs

A l'issu de ce TP, l'étudiant sera capable de :

- Connaître comment représenter une image numérique en C/C++.
- Comprendre les caractéristiques de base d"une image numérique avec la bibliothèque OpenCV en langage C/C++.
- Savoir manipuler les images numériques couleurs avec OpenCV.

Pré-requis

- Pour pouvoir suivre ce TP avec succès il faut au préalable savoir:
    - Les notions de base d'images numériques.
    - Des notions de base en programmation en langage C/C++.

Introduction

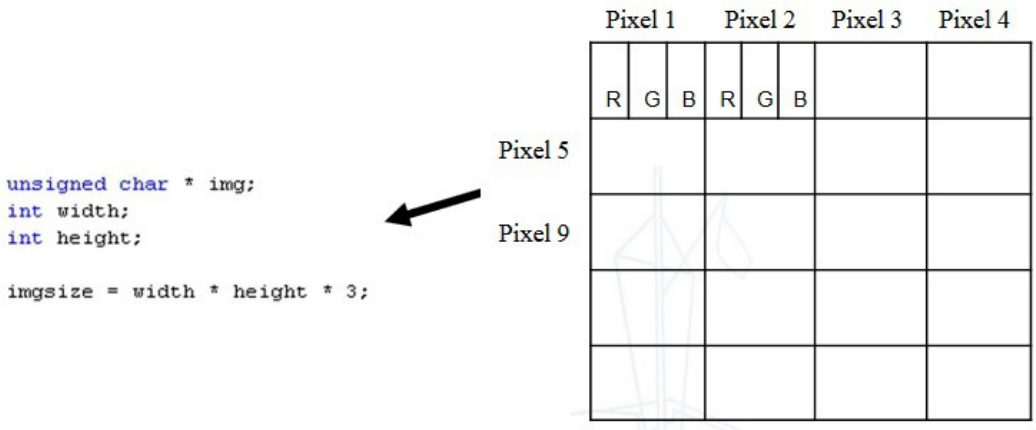
- Ce TP a pour but d'illustrer certaines notions de base concernant le traitement d'images. En particulier, on abordera les fonctions de base pour utiliser les images numériques en utilisant la bibliothèque OpenCV.
- OpenCV est une Bibliothèque de traitement d'images en langageC/C++, optimisée proposée par Intel pour Windows et Linux.

Utilisation rapides des fonctions de base de traitement d'images avec OpenCv

La représentation d'une image en C/C++	6
Le type IplImage défini par OpenCV	7
La création d'image	7
Chargement et affichage d'un fichier image	8
Exercice	9
Exercice	10
Manipuler des images en couleur	10
Exercice	11

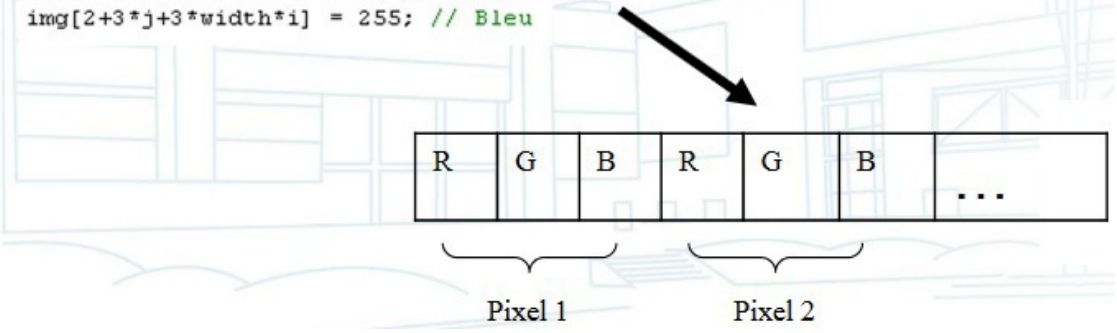
1. La représentation d'une image en C/C++

- En langage C/C++, une image numérique peut représenter par un Tableau à deux dimensions.



- On peut aussi représenter par un tableau un seul dimension.

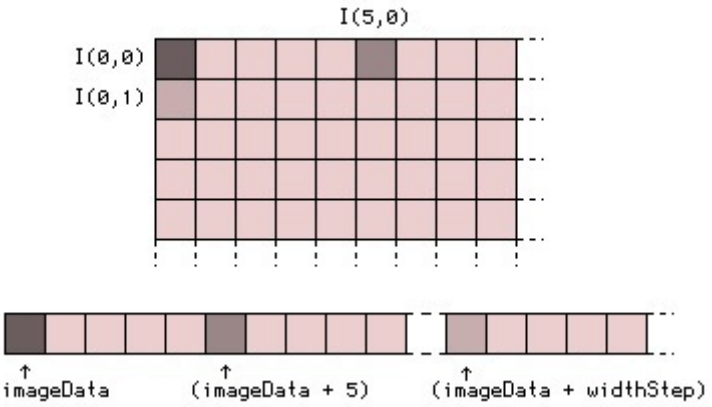
```
// Pixel à ligne i, colonne j
img[0+3*j+3*width*i] = 0; // Rouge
img[1+3*j+3*width*i] = 0; // Vert
img[2+3*j+3*width*i] = 255; // Bleu
```



2. Le type IplImage défini par OpenCV

- Le type IplImage défini avec la bibliothèque OpenCV avec :
  - La taille d'une image peut être connue avec les champs height (nombre de colonnes, donc largeur de l'image) et width (nombre de lignes, donc hauteur de l'image):

```
typedef struct _IplImage
{
    int  nChannels;     /* 3 composantes pour une image couleur (rouge, vert, bleu) */
    int  depth;         /* 8 bits (1 octet, taille d'un char) pour chaque composante */
    int  width;         /* Largeur de l'image en pixels */
    int  height;        /* Hauteur de l'image en pixels */
    char *imageData;  /* Pointeur vers les données de l'image */
    ...
}
IplImage;
```



3. La création d'image

Pour la création d'une image, on doit premièrement inclure la bibliothèque *cv.h*

- Déclaration et création d'une image

Il est possible que vous ayez déjà entendu parler d'« images 8, 16 ou 32 bits ». Dans cette expression, le nombre de bits désigne la taille d'un pixel en mémoire. On appelle cela la profondeur (depth) de l'image. En toute rigueur, la profondeur d'une image décrit le nombre de valeurs sur lequel l'intensité lumineuse est échantillonnée. Par exemple, sur une image en niveaux de gris 8 bits (non signés), l'intensité lumineuse sera échantillonnée sur 256 valeurs, avec 0 pour le noir et 255 pour le blanc. Cette profondeur, dans OpenCV, est décrite par le

- champ depth de la structure IplImage.

p.15

- La structure *« CvSize »* permet de décrire la largeur et la hauteur de l'image.

```
#include <cv.h>
IplImage* im = cvCreateImage(
    cvSize(320, 240), // Taille de l'image
    8, // 8 bits (1 octet, taille d'un char) pour chaque composante
    3 // 3 composantes pour une image couleur (rouge, vert, bleu)
    );
```

- Accéder et remplir les éléments d'une image: (codage par défaut= BGR)

```
IplImage* frame;

...

unsigned char* data = reinterpret_cast<unsigned char*>(frame->imageData);

for (int i = 0; i < frame->height; i++)
{
    for (int j = 0; j < frame->width; j++)
    {
        data[0+3*j+3*frame->width*i] = 0; // Bleu
        data[1+3*j+3*frame->width*i] = 0; // Vert
        data[2+3*j+3*frame->width*i] = 255; // Rouge
    }
}
```

4. Chargement et affichage d'un fichier image

On doit utiliser la bibliothèque *« highgui.h »* pour la récupération et affichage d'images : lecture /enregistrement de fichiers images et videos, gestion des webcams, affichage dans des interfaces graphiques...

- Lire une image

```
#include <highgui.h>
IplImage *im;
im = cvLoadImage("mon_image.jpg", 1); /* (1) */
/* 1 => 3 canaux (0 => 1 seul, -1 => automatique) */
im = cvLoadImage("mon_image.jpg", 0); /* (2) */
```

- Afficher une image

```
cvNamedWindow("Ma fenetre", CV_WINDOW_AUTOSIZE);
cvShowImage("Ma fenetre", im);
cvWaitKey(0);     /* Attendre qu'une touche soit pressée */
```

- Sauvegarder une image : voici maintenant la fonction qui nous permettra d'enregistrer une image sur le disque dur, les argument de cette fonction sont :
  - filename: chemin du fichier à sauvegarder
  - image: image à sauvegarder
  - params: paramètres optionnels (retourne : 1 si le fichier a bien été sauvegardé, 0 sinon).
  - CvArr signifie « array » (tableau).

```
int cvSaveImage (const char* filename, const CvArr* image, const int* params);
```

- Libération de l'espace mémoire

```
cvReleaseImage(&im);
```

5. Exercice

[solution n°1 p.14]

Quelle est la fonction qui permet d'afficher une image en OpenCV

- cvCreateImage
- cvNamedWindow
- cvShowImage

6. Exercice

[solution n°2 p.14]

La fonction *« cvReleaseImage »* permet de créer une image:

- Vrai
- Faux

7. Manipuler des images en couleur

- Séparer les canaux d'une image couleur RGB 8 bits en Troix canaux mono-canal 8 bits (rouge, vert et bleu)

```
void my_split (const IplImage* src, IplImage* blue, IplImage* green, IplImage* red);
```

- Assemble une image BGR 8 bits à partir de trois images mono-canal 8 bits.

```
void my_merge (const IplImage* blue, const IplImage* green, const IplImage* red, IplImage* dst);
```

- Convertir l'image d'un espace vers un autre espace :
  - RGB->HSV

```
img_rgb = imread("pic.png",1);
cvtColor(img_rgb,img_hsv,CV_RGB2HSV);
```

- RGB->Niveau de gris

```
cvtColor(img, grayImg, CV_BGR2GRAY);
```

8. Exercice

[solution n°3 p.14]

La fonction *« my_split »* permet de

- ☐ Convertir l'image RGB en HSV
- ☐ Assembler une image BGR 8 bits à partir de trois images mono-canal 8 bits.
- ☐ Séparer les canaux d'une image couleur RGB 8 bits en Troix canaux mono-canal 8 bits

Énoncé de TP2

II

Tester les commandes précédentes (sur différentes types d'images).

On utilisera l'image couleur Lena.jpg

Écrire un code en C en utilisant la bibliothèque OpenCV qui permet de :

1. Chargez l'image Lena et visualisez la dans une fenêtre.
2. Afficher la taille de l'image (nombre de pixels).
3. Visualisez la représentation matricielle de cette image.
4. Convertir l'image vers l'espace HSV, sauvegarder et afficher la nouvelle image.
5. Convertir l'image en niveaux de gris, sauvegarder afficher la nouvelle image.

Compiler et exécuter le code.

Conclusion

II

- Dans ce deuxième TP, nous avons vu :
  - L'utilité de les bibliothèques « *cv.h* » et « *highgui.h* »
  - Comment représenter une image en C/C++
  - Comment créer, déclarer, afficher, remplir , charger, convertir et supprimer une image.
- Il existe encore de nombreuses fonctions, structures de données,. .
- Toutes les fonctions matricielles fonctionnent aussi sur les images.
- La documentation de Open CV (papier et en ligne) décrit pratiquement tous les algorithmes implémentés et présentée quelques références bibliographiques, des exemples de code et des fichiers d'exemples de code pour les problèmes assez complexes

Solutions des exercices

> **Solution** n°1

Exercice p. 9

Quelle est la fonction qui permet d'afficher une image en OpenCV

- ○ cvCreateImage
- ○ cvNamedWindow
- ◉ cvShowImage

> **Solution** n°2

Exercice p. 10

La fonction *« cvReleaseImage »* permet de créer une image:

- ○ Vrai
- ◉ Faux

> **Solution** n°3

Exercice p. 11

La fonction *« my_split »* permet de

- ☐ Convertir l'image RGB en HSV
- ☐ Assembler une image BGR 8 bits à partir de trois images mono-canal 8 bits.
- ☑ Séparer les canaux d'une image couleur RGB 8 bits en Troix canaux mono-canal 8 bits

Webographie

https://perso.esiee.fr/~perretb/I5FM/TAI/histogramme/index.html#inversion-et-introduction-a-opencv

sdz.tdct.org/sdz/introduction-a-la-vision-par-ordinateur.html