MODULE 2

Manipulations de base avec les images

Objectifs de ce module :

- ✓ ***Ouvrir un fichier d'image***
- ✓ ***Sauvegarder un fichier d'image***
- ✓ ***Utiliser adéquatement la classe d'image Mat.***

Table des matières

Sujets ***Page***

Introduction

Ce module nous permettra de mettre en place les éléments essentiels pour pouvoir manipuler les images avec la bibliothèque d'OpenCV. Nous utiliserons les fonctions qui permettent d'ouvrir et de sauvegarder les images. Nous nous attarderons également à l'étude de la classe d'image Mat qui représente une image en mémoire.

Manipulations de base avec les images

Ouvrir un fichier d'image

But : Ouvrir un fichier d'image avec imread, et afficher l'image à l'écran.

Fonction : Mat **imread**(const string& **filename**, int **flags**=1)

Paramètres:

const string &filename: Le nom du fichier d'image à ouvrir. Ce peut être directement une chaîne de caractère ou une chaîne de type « string ».

int flags = 1 Ce paramètre est égal à 1 par défaut ce qui est équivalent à la constante CV_LOAD_IMAGE_COLOR qui permet de charger l'image en couleur (BGR). Les constantes possibles sont les suivantes :

- CV_LOAD_IMAGE_UNCHANGED: Valeur < 0 par défaut
  Charge l'image telle quelle (incluant le canal alpha si présent)
- CV_LOAD_IMAGE_GRAYSCALE :Valeur égale à 0
  Charge l'image en utilisant un canal (ton de gris).
- CV_LOAD_IMAGE_COLOR: Valeur >0 par défaut
  Charge l'image en utilisant 3 canaux de format BGR (couleur)

Exemple:

```
Mat  ImgSource
ImgSource = imread("monimage.jpg", CV_LOAD_IMAGE_COLOR);
```

Ce qui est équivalent à avoir appeler la fonction de la façon suivante :

```
ImgSource = imread("monimage.jpg");
```

Charge, dans l'objet ImgSource, l'image nommée "monimage.jpg" située dans le répertoire courant avec le format RBG (couleur).

Note:
OpenCV supporte différents formats d'image. Les formats les plus courants sont les suivants:

- Windows bitmap (bmp)
- Jpeg
- PNG
- TIFF

Création d'une fenêtre d'affichage

But : Créer une fenêtre pour permettre d'afficher une image à l'écran.

Fonction : void namedWindow(const string& **winname**, int **flags**=CV_WINDOW_AUTOSIZE)

Paramètres:

const string& **winname**: Le nom de la fenêtre qui devient le titre et l'identificateur de la fenêtre.

int flags = CV_WINDOW_AUTOSIZE Ce paramètre est égal à CV_WINDOW_AUTOSIZE par défaut et permet d'ajuster automatiquement la dimension de la fenêtre d'affichage en fonction des dimensions de l'image à afficher. Il est cependant important de noter que la fenêtre ne peut pas être redimensionnée par la souris.

Les autres valeurs :

CV_WINDOW_NORMAL Ce paramètre affiche la fenêtre en fonction de la dimension de l'image mais permet un redimensionnement avec la souris.

Exemple:

```
namedWindow("Image Source");
namedWindow("Image Source", CV_WINDOW_NORMAL);
```

Sauvegarde d'un fichier d'image sur disque

But : Sauvegarder sur le disque une image.

Fonction : bool imwrite (const string& **filename**,
InputArray **img**,
const vector<int>& **params**=vector<int>())

Paramètres:

const string& **filename**: Le nom de l'identificateur de la fenêtre tel que défini par l'appel à la fonction namedWindow.

InputArray img Ce paramètre représente l'image en mémoire. C'est un objet de la classe Mat défini précédemment.

const vector<int>& **params**=vector<int>()

- Pour JPEG, ce paramètre est une valeur de 0 à 100. Le défaut vaut 95.
- Pour une format de type PNG, cette valeur est une valeur du niveau de compression utilise de 0 à 9. Une valeur plus élevée signifie une grosseur de fichier plus petite mais un temps plus élevé pour effectuer la compression. La valeur de défaut vaut 3.

La function imwrite sauvegarde l'image avec le nom du fichier spécifié. Le format de l'image est déterminé par l'extension du fichier. Seulement des images ayant une représentation des pixels sur 8 bits (ou 16 bits non signés (CV_16U) dans le cas des PNG, JPEG 2000 ou TIFF) sur un canal ou 3 canaux peuvent être sauvegardées avec cette fonction. Si le format, la profondeur ou l'ordre des valeur RGB est différent, il faut alors utiliser la méthode « **convertTo()** » de la classe Mat ou la fonction **cvtColor()** pour convertir l'image avant de la sauvegarder.

Exemple :

```
imwrite( "ImageResultat.jpg", ImgResultat );
```

On sauvegarde donc l'image qui est dans le tampon ImgResultat dans un fichier que l'on appelle ImageResultat.jpg. L'image sur disque aura donc un format Jpeg puisque l'extension du fichier est de ce type.

Exemple complet

```
#include <opencv2\core\core.hpp>                //Module pour les structures de base et la classe d'image Mat
#include <opencv2\highgui\highgui.hpp>          // Module pour les fonctions d'affichage à l'écran
#include <opencv2\imgcodecs\imgcodecs.hpp>  //Module pour l'écriture d'une image sur disque

 using namespace cv;          // Pour éviter de "trainer" le cv:: devant les noms.

 int main( int argc, char** argv )          // argc et argv uniquement nécessaire si vous prévoyez envoyer des
                                            // paramètres à la ligne de commande.
                                            // Exemple : MonProgramme.exe   Param1  Param2
                                            // Param1 serait disponible dans argv[1] et Param2 dans argv[2].
{

            Mat ImgSource;                  // Déclaration d'un objet de la classe Mat.  C'est un objet qui représente
                                            // l'image en mémoire.
            Mat ImgResultat;                // Objet Mat pour l'image résultante.  Ici, elle sera en ton de gris.

            ImgSource = imread("Image.jpg", CV_LOAD_IMAGE_COLOR );

            if (! ImgSource.data )
            {
                        cout <<  "Erreur dans l'ouverture du fichier" << endl;
                        return  -1;
            }

            cvtColor( ImgSource, ImgResultat, CV_BGR2GRAY );

            imwrite( "../../images/Image_Gris.jpg", ImgResultat );

            namedWindow( "Image source", CV_WINDOW_AUTOSIZE );
            namedWindow( "Image Resultat", CV_WINDOW_AUTOSIZE );

            imshow( "Image source", ImgSource );
            imshow( "Image Resultat", ImgResultat );

            waitKey(0);

            return 0;
 }
```

Explication du code:

- On commence par :
    - Créer un objet de la classe Mat pour représenter l'image source en mémoire.
      Mat ImgSource;
    - On fait la même chose pour l'image résultante.
      Mat ImgResultat;

- Charger l'image source

  - Charger une image avec la fonction imread. L'image se trouve dans le répertoire donné par le deuxième paramètre à la fonction. On suppose, dans l'exemple, que l'on charge une image RGB (couleur) d'où l'usage de la constante CV_LOAD_IMAGE_COLOR. On aurait pu la charger directement en ton de gris avec la constante CV_LOAD_IMAGE_GRAYSCALE

    ```
    ImgSource = imread( "Nom_du_fichier", CV_LOAD_IMAGE_COLOR );
    ```

- Traitement de l'image source

  - On convertit l'image de format RGB en ton de gris (Grayscale). OpenCV possède ce qu'il faut pour effectuer ce genre de transformation facilement avec la fonction cvtColor :

    ```
    cvtColor(ImgSource, ImgResultat, CV_BGR2GRAY);
    ```

    cvtColor prend 3 paramètres:

    - L'image source (ImgSource)
    - L'image de destination (ImgResultat)
    - Le troisième paramètre indique le type de transformation qui sera effectuée sur l'image source

- Sauvegarde de l'image résultante.

  - On sauvegarde l'image que l'on veut sur disque avec la fonction **imwrite**.

    ```
    imwrite( "../../images/Image_Gris.jpg", ImgResultat );
    ```

- Visualisation à l'écran

  - On crée 2 fenêtres d'affichage et on associe les images à afficher à ces fenêtres :

    ```
    namedWindow( "Image source", CV_WINDOW_AUTOSIZE );
    namedWindow( "Image Resultat", CV_WINDOW_AUTOSIZE );

    imshow( "Image source", ImgSource );
    imshow( "Image Resultat", ImgResultat );
    ```

- On utilise ensuite la fonction waitKey(0) pour attendre que l'usager presse une touche pour quitter. En fait, le paramètre indique le temps qu'il faut attendre avant de passer à l'instruction suivante. Ce temps est exprimé en milliseconde. Un temps à 0 indique une attente infinie.