C++

TP

Réalisation d'une application de traitement d'images

rahima.zaouche@u-bourgogne.fr

charles.meunier@u-bourgogne.fr

OBJECTIFS

Pendant de longues heures de cours et de TD, vous avez ingurgité moultes connaissances en lien avec la programmation orientée objets et le C++. Il est temps de restituer tout cela à travers la création d'une application complète.

Profitez bien de chaque séance pour avancer et bénéficier de l'expérience de l'enseignant en lui posant des questions.

Vous trouverez, ci-dessous, une liste des contraintes et des fonctionnalités attendues.

N'oubliez pas que votre enseignant est présent pour répondre à vos questions.

CONTRAINTES

CONDITIONS DE TRAVAIL

Vous travaillerez par groupe de 2.

Vous disposerez de quatre séances pour réaliser cette application, soit 8h x 2 étudiants = 16h.

ENVIRONNEMENT DE TRAVAIL ET TECHNOLOGIES

Vous utiliserez Visual Studio 2017 comme environnement de développement.

Vous utiliserez le C++ et la bibliothèque OpenCV pour développer votre application.

Votre code sera compréhensible et documenté.

Vous utiliserez GitHub pour versionner votre code et travailler de manière collaborative.

Vous pouvez utiliser votre ordinateur personnel.

LIVRABLES

Sont attendus les éléments suivants :

- La solution Visual Studio 2017 de votre projet.
- Le diagramme des classes UML de votre projet.
- Le manuel utilisateur de votre application.
- Tout élément de documentation que vous jugerez utile.
- Un lien vers votre projet GitHub.

Vous restituerez votre travail à la fin de la quatrième séance. Aucun délai supplémentaire ne sera accordé.

Toute duplication de code d'un autre groupe de travail sera sanctionnée par un zéro.

FONCTIONNALITES ATTENDUES

IHM

L'application développée fonctionnera en mode "console" et proposera un menu permettant d'accéder aux différentes fonctionnalités.

ARCHITECTURE MVC

Votre application sera conçue à partir d'une architecture MVC.

CHARGEMENT D'UNE IMAGE

L'application permettra de charger une image à partir de son chemin absolu (ex : C:\mon-image.jpg).

TRAITEMENTS D'IMAGES

L'application permettra de réaliser plusieurs traitements sur l'image chargée, grâce à la bibliothèque OpenCV :

- Filtrage :
  - Filtre médian
  - Filtre gaussien
- Dérivation
  - Calcul du gradient dans une image (Sobel)
- Opérations de morphologie mathématique :
  - Dilatation
  - Erosion
- Détection de contours :
  - Application d'un détecteur de contours « Canny »
- Segmentations d'images :
  - Opérations de seuillages
  - Segmentation par croissance de région

OPENCV

INSTALLATION DE LA BIBLIOTHEQUE OPENCV

OpenCV est la libraire de référence dédiée au domaine du traitement d'images, c'est une boite à outil qui intègre plusieurs algorithmes en C++, permettant de réaliser différents types d'opérations dans une image ou dans une vidéo. Ces opérations peuvent être simples telles qu'un changement de contraste, des rotations, mais elles peuvent être aussi plus complexes comme la détection d'objets ainsi que de nombreuses autres fonctions.

L'installation d'OpenCV dépend de l'environnement que vous utilisez pour développer, dans ce TP il s'agit d'installer OpenCV sous windows, en utilisant Microsoft Visual Studio. Pour ce faire, nous allons suivre les étapes décrites ci-dessous :

Etape 1 : Installation de la bibliothèque OpenCV et modification des variables d'environnements

- Tapez l'url : https://opencv.org sur votre navigateur, choisissez la dernière version proposée et télécharger l'exécutable sous windows ,
- Extraction du fichier « .exe » dans le répertoire : c :\opencv .
- Après l'extraction du fichier « .exe » dans le répertoire c:\opencv, nous allons revoir les paramètres d'environnement sur notre ordinateur. Il faudra rajouter dans PATH, le chemin correspondant au « bin » de notre libraire :
- Aller sur Ordinateur
- Propriétés système
- Paramètres système avancés
- Cliquer sur « Variables d'environnement » et dans la liste qui se trouve sur les Variables système, cherchez : Path et cliquer sur « Modifier »
- Ajouter le Path au répertoire « bin » du dossier « opencv » comme suit :
- Nouveau → ajouter ce chemin : C:\opencv\build\x64\vc14\bin
- Cliquer sur ok.

Etape 2 : Configuration de Visual studio

- Ouvrir Visual studio à partir de la barre de recherche
- Nous allons créer un nouveau projet appelé : opencvTry
- File → New → Project
- Une fois le projet créé, aller sur l'arborescence de votre projet et Cliquer sur « Sources files »
- Add –> New Item
- Créer un fichier appelé « try.cpp »

Avant de passer aux prochaines étapes, nous allons tenter d'exécuter un petit programme qui permet d'afficher une image !

Exemple : Charger une image soit : «lena.jpg » à partir de votre navigateur. Enregistrer cette image dans votre répertoire « opencvTry ». L'image doit être dans le même répertoire ou se trouve votre fichier « try.cpp »

- Taper le code suivant :

```
# include <opencv2/opencv .hpp>
# include <iostream>

using namespace std;
using namespace cv;

int main ()
```

```
{
	Mat img = imread (“Lena.jpg”);
	namedWindow (“image”, WINDOW_NORMAL);
	imshow (“image”, img);
	waitKey (0);
	return 0;
}
```

On peut directement s'apercevoir qu'il y a un souci avec la première ligne du code, à savoir la bibliothèque OpenCV, ceci est du au fait que nous n'avons pas encore créé de lien entre Visual studio et la bibliothèque d'OpenCV. C'est ce qu'on va faire dans l'étape qui suit, mais une dernière vérification est nécessaire, il faut s'assurer que vous êtes bien en mode « Debug ».

Etape 3 : Raccorder les librairies d'OpenCV vers visual studio

Le répertoire « include » du dossier « opencv » contient les différents fichiers d'en-tête, il est donc indispensable de rajouter le chemin de ce répertoire dans les paramètres du compilateur :

- Clique droit sur le répertoire « opencvTry »
- Aller sur Propriété : C/C++ → General
- Copier le path C:\opencv\build\include dans le champ associé à ***Additional include Directories*** .
- Quant aux bibilothèques, elles se retrouvent dans le sous répertoire lib, on doit indiquer le chemin de ce sous-répertoire comme suit :
- Choisir C/C++ → Linker → General
- Copier le path C:\opencv\build\x64 \vc14\lib dans le champ associé ***à Additonnal Library Directories.***
- Enfin vérifier que la configuration est bien en mode Debug !
- Choisir C/C++ → Linker → Input
- Additionnal depencencies → Edit :
- Copier dans la zone de texte le nom de ce fichier : opencv_world320d_lib
- Confirmer toutes ces modifications en appuyant sur « Appliquer » et retenter de tester votre programme à présent.

Vous remarquerez qu'il n'a y a plus d'erreur sur visual studio, Maintenant vous pouvez exécuter votre programme ! Bravo, l'image s'est bien chargée !

INTRODUCTION A OPENCV

Dans cette partie, nous allons nous familiariser avec les fonctions essentielles d'OpenCV, qui permettent de traiter des images, les points suivants seront abordés :

- Type élémentaire : Images (matrices) et utilisation de la classe ***Mat***,
- Opérations d'entrées/sorties : Lecture d'une image, affichage d'une image,
- Changement de l'intensité dans une image,
- Sauvegarde d'une image.

Rappel :

La méthode ***imread( )*** permet de charger une image et de la lire,

La méthode ***imshow( )*** permet d'afficher une image,

La méthode ***imwrite ( )*** permet de sauvegarder une image.

Type élémentaire : Images (matrices)

Pour charger une image, nous faisons appel à la classe « **Mat** » et à la méthode **imread():**

- Créer un nouveau projet, appelé : LoadImage et taper le code suivant :

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
    /*
      Création de l'objet image et lecture de l'image à partir du
      répertoire associé en utilisant la méthode imread()
    */
    Mat image = imread("D:/photo.jpg");

    // Vérifier si l'image existe bien dans le répertoire
    if (image.empty())
    {
        cout << "Could not open or find the image" << endl;
        cin.get(); //wait for any key press
        return -1;
    }

    // Affichage de l'image dans une fenêtre (Image)
    string windowName = "Image";
    namedWindow(windowName);
    imshow(windowName, image);
    waitKey(0);
    destroyWindow(windowName);

    return 0;
}
```

Changement d'intensité d'une image

Modifier la luminosité dans une image I (i,j) est une opération très pratiquée en traitement d'images, l'objectif est de modifier les valeurs pixelliques, qui constituent les éléments ou les portions de notre Image en largeur comme en longueur. On peut définir cette opération comme suit :

$$I2(i,j) = I(i,j) + \mathbf{c}$$

$$I2(i,j) = I(i,j) - \mathbf{c}$$

Avec I2(i,j): la nouvelle image traitée et **c** : une constante

```
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
    // Lecture de l'image avec la méthode imread ()

    Mat image = imread("Image.jpg");

        if (image.empty())
    {
        cout << "Could not open or find the image" << endl;
        cin.get(); //wait for any key press
        return -1;
    }

// Changement de luminosité dans l'image en utilisant la méthode
convertTo()

    Mat imageBrighnessHigh100;
    image.convertTo(imageBrighnessHigh100, -1, 1, 100);
    Mat imageBrighnessLow100;
    image.convertTo(imageBrighnessLow100, -1, 1, -100);

    namedWindow(windowNameBrightnessLow100, WINDOW_NORMAL);
```

```
/*
Affichage de l'image de l'image originale et des deux nouvelles
images dans trois fenêtres différentes avec la méthode imshow ()
*/

    String windowNameOriginalImage = "Original Image";
    String windowNameBrightnessHigh100 = "Brightness Increased by
100";
    String windowNameBrightnessLow100 = "Brightness Decreased by
100";
    namedWindow(windowNameOriginalImage, WINDOW_NORMAL);
    namedWindow(windowNameBrightnessHigh100, WINDOW_NORMAL);

 imshow(windowNameOriginalImage, image);
    imshow(windowNameBrightnessHigh100, imageBrighnessHigh100);
    imshow(windowNameBrightnessLow100, imageBrighnessLow100);


// Sauvegarde d'une image traitée à l'aide de la méthode imwrite ()

        imwrite("E:\My_new_image.jpg", imageBrighnessHigh100);

    waitKey(0);
    destroyAllWindows();
    return 0;
}
```