Introduction à OpenCV

1. Introduction

OpenCV (**Open** Source for **C**omputer **V**ision) est une librairie en C/C++ regroupant de nombreuses bibliothèques permettant le traitement d'images et de vidéos. A l'heure de l'écriture de ce sujet de travaux pratiques, la version stable courante est le 2.4.1. Des milliers d'algorithmes y sont implémentés, ainsi que des structures de données facilitant l'accès et la manipulation des données brutes de l'image ou de la vidéo.

OpenCV regroupe plusieurs modules :

- Core module : contient les fonctionnalités de base, notamment l'accès aux pixels, le changement de luminosité, de contraste, le changement d'espace couleur, la possibilité de « dessiner » sur les images etc.
- Imgproc module : permet d'appliquer différents filtres (moyen, gaussien, médian…), permet d'appliquer différentes opérations morphologiques (dilatation, érosion, ouverture, fermeture, gradient morphologique etc…), il permet de créer des pyramides d'images…, contient différents algorithmes de seuillage et de détection de contours (Sobel, opérateur de Laplace, Canny Edge Detector…), mais aussi des détecteur de droites ou d'ellipse (Hough Line/Cercle Transform).
- Highgui module : permet l'ajout de composants graphiques de base, mais aussi avancés.
- Calib3D module : permet la calibration des caméras et la reconstruction 3D.
- Feature2D module : contient des descripteurs 2D souvent utilisés basés sur les couleurs, la forme, la texture, les points d'intérêt… et des algorithmes de mises en correspondance.
- Video module : contient les fonctionnalités de base de traitement de vidéos (algorithmes de détection de mouvement, de suivi, d'extraction de plan principal etc…).
- Objdetect module : contient des algorithmes de détection d'objet, notamment la détection de visages…
- ML module : contient les algorithmes d'apprentissage et de classification.
- GPU module : contient les algorithmes permettant l'utilisation de la carte graphique afin d'accélérer le temps d'exécution grâce au parallélisme des GPU.

2. Utilisation d'OpenCV avec Eclipse sous noyau Linux.

Les tps se feront avec Eclipse et OpenCV sous linux. L'installation de la bibliothèque OpenCV sous windows nécessite la recompilation de la bibliothèque avec MinGW et CMake. Référez-vous à l'aide en ligne pour le faire.

- Créer un projet C/C++ sous Eclipse nommé master1.
- Inclure la bibliothèque. Pour connaître l'emplacement exécutez la commande : ***pkg-config --cflags opencv***

- Intégrer les bibliothèques au Linker. Pour connaître l'emplacement, taper la commande ***pkg-config --libs opencv***.

3. Lire une Image

Pour lire une image utiliser la fonction imread.

Mat img = imread(filename)

La classe Mat (pour matrix) permet de stocker l'image sous forme matricielle sous la forme BGR donc avec 3 canaux à 8 bits chacun.

Pour obtenir une image en niveaux de gris donc sur un canal de 8 bits, écrire :

Mat img = imread(filename, 0);

Pour afficher une image écrire :

```
namedWindow( "Display Image", CV_WINDOW_AUTOSIZE );
imshow( "Display Image", img );
waitKey(0) ;
```

La classe Mat contient un header de l'image et les données brutes.

```
Mat A, C;
A = imread(argv[1], CV_LOAD_IMAGE_COLOR);
Mat B(A);
C = A;
```

Dans l'exemple précédent, une copie des headers est effectuée, par contre toute modification des données dans l'un se retrouvera dans les autres.

```
Mat F = A.clone();
Mat G;
A.copyTo(G);
```

Pour faire une copie, il faut utiliser les fonctions ci-dessus.

4. Conversion en niveaux de gris (manuellement)

Écrire une fonction toGrayScale qui retourne une Mat en niveaux de gris. Tester la méthode.

```
Vec3b intensity = img.at<Vec3b>(x, y);
uchar blue = intensity.val[0];
uchar green = intensity.val[1];
uchar red = intensity.val[2];
```

On obtient ainsi les 3 canaux qui nous intéressent pour le pixel de coordonnées x et y.

Mat result ;
result.create(img.size(),CV_8U);

Pour créer une Mat avec la même taille que img contenant des octets non signés uchar.

Essayez la fonctions ***cvtColor(src, src_gray, CV_RGB2GRAY);***

5. Extraction d'histogramme

Écrire une fonction getGrayScaleHistogram permettant d'obtenir l'histogramme d'une image Mat img.

CV_Assert(img.depth() == CV_8U); permet de vérifier que img est bien une image en niveaux de gris.

6. Utilisations de filtres

Mat kern = (Mat_<char>(3,3) << 0, -1, 0,
-1, 5, -1,
0, -1, 0);

Ci-dessus nous avons créé la matrice suivante $\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$

En s'inspirant de cet exemple et en utilisant la méthode filter2D

(http://opencv.itseez.com/modules/imgproc/doc/filtering.html#filter2d), créez les fonctions meanFilter et gaussianFilter à partir des filtres respectifs suivant :

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} 1 & 9 & 18 & 9 & 1 \\ 9 & 81 & 162 & 81 & 9 \\ 18 & 162 & 324 & 162 & 18 \\ 9 & 81 & 162 & 81 & 9 \\ 1 & 9 & 18 & 9 & 1 \end{pmatrix}$$

Il existe des fonctions prédéfinies capables d'effectuer ces traitements, trouvez-les et appliquez sur les images.

7. Projet : Collecte automatique des réponses d'un QCM ou questionnaire

Le but du projet du semestre est d'automatiser la collecte de réponses d'un qcm ou d'un questionnaire.

Ecrire une fonction qui permet de faire la différence de NdG pixel à pixel de deux images. La fonction retournera une image en NdG.

Tester avec les images de la base « questionnaire 01 ».