Télécom Physique Strasbourg - Université de Strasbourg - TIC Santé 2A

TP1 : Images et Vision

Introduction

Objectifs du TP :

- Compilation avec CMake
- Utilisation de la librairie OpenCV
- Premières opérations sur des images

Pour la compilation des programmes, on utilisera cmake et le compilateur `g++` sous le système Linux. On pourra trouver la documentation d'OpenCV sur le site `http://docs.opencv.org` et un bref résumé de certaines fonctions dans le fichier `http://docs.opencv.org/2.4/opencv_cheatsheet.pdf`. Il est conseillé de créer un répertoire par exercice. On pourra récupérer l'image de test avec la commande `cp /adhome/n/np/npadoy/lena.jpg ./`

Prise en mains

1. **Un peu de lecture**
   Parcourir la documentation pour se faire une idée des possibilités offertes par la librairie OpenCV.
2. **Affichage d'une image** [fichier image.cpp]
   Copier le code suivant qui lit une image et l'affiche dans une fenêtre :

```
#include <stdio.h>
#include <opencv2/opencv.hpp>

int main( int argc, char** argv )
{
    cv::Mat img;
    img = cv::imread( "lena.jpg");
    if( !img.data ) {
        printf( "No image data \n" );
        return -1;
    }
    cv::namedWindow( "Display Image", CV_WINDOW_AUTOSIZE );
    cv::imshow( "Display Image", img );
    cv::waitKey(0);
    return 0;
}
```

   Compiler et tester le code à l'aide de CMake, en utilisant le fichier `CMakeLists.txt` suivant :

```
cmake_minimum_required(VERSION 2.8)
project(image)
find_package( OpenCV REQUIRED )
include_directories( ${OpenCV_INCLUDE_DIRS} )
add_executable( ${PROJECT_NAME} image.cpp )
target_link_libraries( ${PROJECT_NAME} ${OpenCV_LIBS} )
```

Remarque : le programme précédent recherche l'image dans le répertoire courant. Il peut être nécessaire de changer le chemin dans `imread`.

3. **Manipulation des matrices** [fichier matrices.cpp]
   (a) Créer une matrice `R` de taille 2x2 de type `CV_32FC1`. Regarder la signification des constantes `CV_8UC1`, `CV_32FC1`, et `CV_8UC3`, `CV_32FC3`.
   (b) Ecrire le code qui permet à l'utilisateur de la remplir au clavier (utiliser l'opérateur >>). Pour accéder aux éléments d'un objet de type `cv::Mat`, on pourra utiliser la fonction membre `at<T>(i,j)`, où T correspond au type des éléments.
   (c) Afficher l'inverse de la matrice `R`.
   (d) Afficher les résultats de la multiplication par la matrice `R` de points `p` lus au clavier (utiliser le type `cv::Mat` pour les points).
4. **Utilisation des images** [fichier points.cpp]
   (a) Créer une nouvelle image appelée `gray` (objet de type `cv::Mat`) de la même taille que l'image `img`, mais contenant un seul canal de couleur (image en niveaux de gris).
   (b) Utilisez la fonction `cvtColor` pour convertir l'image `img` en niveau de gris et la stocker dans `gray`. Afficher l'image `gray` dans une nouvelle fenêtre.
   (c) Ecrire une fonction `DrawCrossC1` qui trace une croix *blanche* dans une image en niveaux de gris, en vérifiant que le point est bien dans l'image.
   (d) Afficher 4 points lus au clavier sur l'image du premier exercice ainsi que leur barycentre.
   (e) Ecrire une fonction `DrawCrossC3` qui trace une croix *rouge* dans une image couleur.
   (f) Afficher 4 points lus au clavier sur l'image du premier exercice. Utiliser la fonction `cv::line` pour tracer dans l'image le quadrilatère correspondant.

Opérations sur les images

1. **Lissage de l'image en niveaux de gris**
   (a) Utiliser la fonction `cv::GaussianBlur` pour lisser l'image. Afficher le résultat et expérimenter pour différentes valeurs des paramètres `ksize` et `sigma`.
   (b) Intégrer une barre de défilement dans la fenêtre d'affichage pour régler la valeur de `sigma` à la souris, en utilisant la fonction `createTrackbar`.
2. **Detection de contours**
   (a) Utiliser la fonction `Canny` pour afficher les gradients de l'image.
   (b) Intégrer deux barres de défilement dans la fenêtre d'affichage pour régler les valeurs des deux seuils requis par la fonction.

Interactions avec la souris

1. Utiliser la fonction `setMouseCallback` pour définir une fonction `OnClick` qui sera appelée à chaque clic de souris dans la fenêtre d'affichage. Afficher une croix sur chaque point cliqué avec le bouton gauche.
2. Une fois le bouton droit cliqué, effacer de l'image le polygone défini par les points cliqués avec le bouton gauche depuis le dernier clic droit, en utilisant la fonction `fillPoly`.