Robotics: Principles and Practice

Module 5: Robot Vision

Lecture 3: Introduction to OpenCV

David Vernon
Carnegie Mellon University Africa

www.vernon.eu

OpenCV

There are many versions of OpenCV; we will use Ubuntu 16.04 using ROS with OpenCV 3.3

For documentation on OpenCV data structures, functions, and methods, see

https://docs.opencv.org/3.3.0/

Demo

The following code is taken from the imageAcquisition example application

See:

```
imageAcquisition.h
imageAcquisitionImplementation.cpp
imageAcquisitionApplication.cpp
```

To run the example:

```
rosrun module5 imageAcquisition
```

```
/*
  Example use of openCV to acquire and display images
  ---------------------------------------------------
  Interface file

  David Vernon
  14 December 2016

  Audit Trail
  -----------
  6 May 2017 Updated to process multiple images and read image filenames from an input file (imageAcquisitionInput.txt)

*/

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include <ctype.h>
#include <iostream>
#include <string>
#include <conio.h>

//opencv
#include <cv.h>
#include <highgui.h>
#include <opencv2/opencv.hpp>

#define TRUE  1
#define FALSE 0
#define MAX_STRING_LENGTH   80
#define MAX_FILENAME_LENGTH 80

using namespace std;
using namespace cv;

/* function prototypes go here */

void display_image_from_file(char *filename);
void display_image_from_video(char *filename);
void display_image_from_camera(int camera_number);
void prompt_and_exit(int status);
void prompt_and_continue();
```

```
/*=======================================================*/
/* display images from a video file in an openCV window  */
/* pass the filename of the video as a parameter         */
/*=======================================================*/

void display_image_from_video(char *filename) {

   VideoCapture video;     //  the video device
   Mat frame;              //  an image read from a camera
   Mat processedImage;     //  a processed image
   string inputWindowName  = "Input Image";

   namedWindow(inputWindowName, CV_WINDOW_AUTOSIZE); // create the window

   video.open(filename);                    // open the video input
   if (video.isOpened()){
      printf("Press any key to stop image display\n");

      do {
         video >> frame;                    // read a frame from the video

         if (!frame.empty()) {
            imshow(inputWindowName, frame);  // show our image inside it.
            waitKey(30);                     // this is essential as it allows openCV to handle the display event ...
                                             // the argument is the number of milliseconds to wait
         }
      } while ((!_kbhit()) && (!frame.empty()));

      getchar(); // flush the buffer from the keyboard hit
      destroyWindow(inputWindowName);
   }
   else {
      printf("Failed to open video file\n");
      prompt_and_continue();
      return;
   }
}
```

```
/*=============================================*/
/* Display images from a file in an openCV window */
/* pass the filename as a parameter               */
/*=============================================*/

void display_image_from_file(char *filename) {

   string inputWindowName     = "Input Image";

   Mat image;
   Mat processedImage;

   namedWindow(inputWindowName, CV_WINDOW_AUTOSIZE);// create the window

   image = imread(filename, CV_LOAD_IMAGE_COLOR);   // Read the file

   if (!image.data) {                               // Check for invalid input
      printf("Error: failed to read image\n");
      prompt_and_exit(-1);
   }

   printf("Press any key to stop image display\n");

   imshow(inputWindowName, image );                 // show our image inside it.

   do{
      waitKey(30);                                  // Must call this to allow openCV to display the images
   } while (!_kbhit());                             // We call it repeatedly to allow the user to move the windows
                                                    // (if we don't the window process hangs when you try to click and drag)

   getchar(); // flush the buffer from the keyboard hit

   destroyWindow(inputWindowName);
}
```

```
/*=====================================================*/
/* display images from a camera in an openCV window  */
/* pass the index of the camera as a parameter       */
/*=====================================================*/

void display_image_from_camera(int cameraNum) {

   VideoCapture camera;            //  the camera device
   Mat frame;                      //  save an image read from a camera
   vector<int> compressionParams;  // parameters for image write

   char windowName[MAX_STRING_LENGTH];
   char cameraNumber[MAX_STRING_LENGTH];

   strcpy(windowName,"Camera");
   sprintf(cameraNumber, " %d", cameraNum);

   namedWindow(windowName,     CV_WINDOW_AUTOSIZE); // create the window

   if (camera.open(cameraNum) == true) {          // open the camera input

      printf("Press any key to stop image display\n");

      camera >> frame;                            // read a frame from the camera to get the image size ... this is actually C++

      // printf("Camera image is %d x %d\n", frame.cols, frame.rows);

      do {
         camera >> frame;                         // read a frame from the camera
         imshow(windowName, frame);
         cvWaitKey(30); // this is essential as it allows openCV to handle the display event ...
                        // the argument is the number of milliseconds to wait
      } while (!_kbhit());

      getchar(); // flush the buffer from the keyboard hit
```

```
            compressionParams.push_back(CV_IMWRITE_PNG_COMPRESSION);
            compressionParams.push_back(9);                                  // 9 implies maximum compression

            imwrite("../data/camera_image.png", frame, compressionParams);   // write the image to a file just for fun

            destroyWindow(windowName);
        }
        else {
            printf("Failed to open camera %d\n",cameraNum);
            prompt_and_continue();
        }
    }
```

Demo

The following code is taken from the colourToGreyscale example application

See:

```
colourToGreyscale.h
colourToGreyscaleImplementation.cpp
colourToGreyscaleApplicatation.cpp
```

To run the example:

```
rosrun module5 colourToGreyscale
```

```
/*
  Example use of openCV to convert a colour image to greyscale
  -------------------------------------------------------------
  Implementation file

  David Vernon
  8 May 2017

*/

#include "colourToGreyscale.h"

void colourToGreyscale(char *filename) {

   char inputWindowName[MAX_STRING_LENGTH]   = "Input Image";
   char outputWindowName[MAX_STRING_LENGTH]  = "Greyscale Image";

   Mat colourImage;
   Mat greyscaleImage;

   int row;
   int col;
   int channel;
   int temp;

   namedWindow(inputWindowName,   CV_WINDOW_AUTOSIZE);
   namedWindow(outputWindowName,  CV_WINDOW_AUTOSIZE);

   colourImage = imread(filename, CV_LOAD_IMAGE_COLOR);        // Read the file
   // colourImage = imread(filename, CV_LOAD_IMAGE_GRAYSCALE); // just for testing

   printf("number of channels %d\n", colourImage.channels());
```

```
if (!colourImage.data) {                            // Check for invalid input
   printf("Error: failed to read image %s\n",filename);
   prompt_and_exit(-1);
}

printf("Press any key to continue ...\n");

/* test image */
/*
for (row=0; row < colourImage.rows; row++) {
   for (col=0; col < colourImage.cols; col++) {
      for (channel=0; channel < colourImage.channels(); channel++) {
         if (colourImage.channels() == 1) { // failsafe just in case the colourImage is not a colour image or a multichannel image
            colourImage.at<uchar>(row,col) = 80 * (col % 3);             // test image comprises bands of greyscale 0, 80, and 160
         }
         else {
            colourImage.at<Vec3b>(row,col)[channel] = 80 * (col % 3);    // test image comprises bands of greyscale 0, 80, and 160
         }
      }
   }
}
*/
imshow(inputWindowName, colourImage);
```

```
    //CV_Assert(colourImage.type() == CV_8UC3);

    // convert to greyscale by explicit access to colour image pixels
    // we do this simply as an example of one way to access individual pixels
    // see changeQuantisation() for a more efficient method that accesses pixels using pointers

    greyscaleImage.create(colourImage.size(), CV_8UC1);

    for (row=0; row < colourImage.rows; row++) {
       for (col=0; col < colourImage.cols; col++) {
          temp = 0;
          for (channel=0; channel < colourImage.channels(); channel++) {
             if (colourImage.channels()== 1) {
                // failsafe just in case the colour image is not a colour image or a multichannel image
                //temp += colourImage.at<Vec3b>(row,col)[channel]; // don't use this
                temp += colourImage.at<uchar>(row,col);            // use this
             }
             else {
                temp += colourImage.at<Vec3b>(row,col)[channel];
             }
          }
          greyscaleImage.at<uchar>(row,col) = (uchar) (temp / colourImage.channels());
       }
    }

    // alternative ... use OpenCV!!!
    // cvtColor(colourImage, greyscaleImage, CV_BGR2GRAY);

    imshow(outputWindowName, greyscaleImage);

    do{
       waitKey(30);                                  // Must call this to allow openCV to display the images
    } while (!_kbhit());                             // We call it repeatedly to allow the user to move the windows
                                                     // (if we don't the window process hangs when you try to click and drag

    getchar(); // flush the buffer from the keyboard hit

    destroyWindow(inputWindowName);
    destroyWindow(outputWindowName);
}
```

Demo

The following code is taken from the colourToHIS example application

See:

```
colourToHIS.h
colourToHISImplementation.cpp
colourToHISApplicationation.cpp
```

To run the example:

```
rosrun module5 colourToHIS
```

```
/*
  Example use of openCV to convert a colour image to hue, intensity, and saturation
  ---------------------------------------------------------------------------------
  Implementation file

  David Vernon
  8 May 2017
*/

#include "colourToHIS.h"

void colourToHIS(char *filename) {

   char inputWindowName[MAX_STRING_LENGTH]         = "Input Image";
   char hueWindowName[MAX_STRING_LENGTH]           = "Hue Image";
   char intensityWindowName[MAX_STRING_LENGTH]     = "Intensity Image";
   char saturationWindowName[MAX_STRING_LENGTH]    = "Saturation Image";

   Mat colourImage;
   Mat hueImage;
   Mat intensityImage;
   Mat saturationImage;

   int row;
   int col;
   unsigned char red;
   unsigned char green;
   unsigned char blue;
   float hue;
   float saturation;
   float intensity;
```

```
namedWindow(inputWindowName,      CV_WINDOW_AUTOSIZE);
namedWindow(hueWindowName,        CV_WINDOW_AUTOSIZE);
namedWindow(intensityWindowName,  CV_WINDOW_AUTOSIZE);
namedWindow(saturationWindowName, CV_WINDOW_AUTOSIZE);

colourImage = imread(filename, CV_LOAD_IMAGE_COLOR); // Read the file

if (!colourImage.data) {                            // Check for invalid input
   printf("Error: failed to read image %s\n",filename);
   prompt_and_exit(-1);
}

printf("Press any key to continue ...\n");

imshow(inputWindowName, colourImage );

CV_Assert(colourImage.type() == CV_8UC3 );

// convert to HIS by explicit access to colour image pixels
// we do this simply as an example of one way to access individual pixels
// see changeQuantisation() for a more efficient method that accesses pixel using pointers

hueImage.create(colourImage.size(), CV_8UC1);
saturationImage.create(colourImage.size(), CV_8UC1);
intensityImage.create(colourImage.size(), CV_8UC1);
```

```
    intensityImage.create(colourImage.size(), CV_8UC1);

    for (row=0; row < colourImage.rows; row++) {
       for (col=0; col < colourImage.cols; col++) {
          blue  = colourImage.at<Vec3b>(row,col)[0];
          green = colourImage.at<Vec3b>(row,col)[1];
          red   = colourImage.at<Vec3b>(row,col)[2];

          rgb2hsi(red, green, blue, &hue, &saturation, &intensity);
          hueImage.at<uchar>(row,col)        = (char)  (255.0  * (hue/360.0));
          saturationImage.at<uchar>(row,col) = (char)  (saturation * 255);
          intensityImage.at<uchar>(row,col)  = (char)  (intensity * 255);

       }
    }

    imshow(hueWindowName,       hueImage);
    imshow(intensityWindowName, intensityImage);
    imshow(saturationWindowName,saturationImage);

    do{
       waitKey(30);                                 // Must call this to allow openCV to display the images
    } while (!_kbhit());                            // We call it repeatedly to allow the user to move the windows
                                                    // (if we don't the window process hangs when you try to click and drag

    getchar(); // flush the buffer from the keyboard hit

    destroyWindow(inputWindowName);
    destroyWindow(hueWindowName);
    destroyWindow(intensityWindowName);
    destroyWindow(saturationWindowName);
}
```

Demo

The following code is taken from the gaussianFiltering example application

See:

```
gaussianFiltering.h
gaussianFilteringImplementation.cpp
gaussianFilteringApplication.cpp
```

To run the example:

```
rosrun module5 gaussianFiltering
```

```
/*
 * function processNoiseAndAveraging
 * Trackbar callback - add Gaussian noise with standard deviation input from user
 * Trackbar callback - remove noise with local averaging using filter size input from user
*/

void processNoiseAndAveraging(int, void*) {

   extern Mat src;
   extern int noise_std_dev;
   extern int gaussian_std_dev;
   extern char* processed_window_name;

   Mat noisy_image;
   Mat filtered_image;

   int filter_size;

   filter_size = gaussian_std_dev * 4 + 1;
   noisy_image = src.clone();

   addGaussianNoise(noisy_image, 0.0, (double)noise_std_dev);

   GaussianBlur(noisy_image,filtered_image,Size(filter_size,filter_size),gaussian_std_dev);

   imshow(processed_window_name, filtered_image);
 }
```