Introduction to Cognitive Robotics

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 20.04 using ROS with OpenCV 4.2

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

https://docs.opencv.org/4.2.0/

Note: the example code in these notes was written for OpenCV 3.2;
the changes required when porting to OpenCV 4.2 are highlighted

Demo

The following code is taken from the imageAcquisitionFromImageFile example application

See:

```
imageAcquisitionFromImageFile.h
imageAcquisitionFromImageFileImplementation.cpp
imageAcquisitionFromImageFileApplication.cpp
```

To run the example:

```
rosrun module5 imageAcquisitionFromImageFile
```

```
#include "module5/imageAcquisitionFromImageFile.h"

/*================================================*/
/* 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);
}
```

This should be WINDOW_AUTOSIZE for OpenCV 4

This should be IMREAD_COLOR for OpenCV 4

```cpp
#include "module5/imageAcquisitionFromImageFile.h"

int main() {

   #ifdef ROS
      // Turn off canonical terminal mode and character echoing
      static const int STDIN = 0;
      termios term, old_term;
      tcgetattr(STDIN, &old_term);
      tcgetattr(STDIN, &term);
      term.c_lflag &= ~(ICANON | ECHO);
      tcsetattr(STDIN, TCSANOW, &term);
   #endif

   const char input_filename[MAX_FILENAME_LENGTH] = "imageAcquisitionFromImageFileInput.txt";
   char input_path_and_filename[MAX_FILENAME_LENGTH];
   char data_dir[MAX_FILENAME_LENGTH];
   char file_path_and_filename[MAX_FILENAME_LENGTH];


   int end_of_file;
   bool debug = true;
   char filename[MAX_FILENAME_LENGTH];
   int  camera_number;

   FILE *fp_in;
```

```
#ifdef ROS
   strcpy(data_dir, ros::package::getPath(ROS_PACKAGE_NAME).c_str()); // get the package directory
#else
   strcpy(data_dir, "..");
#endif

strcat(data_dir, "/data/");
strcpy(input_path_and_filename, data_dir);
strcat(input_path_and_filename, input_filename);


if ((fp_in = fopen(input_path_and_filename,"r")) == 0) {
  printf("Error can't open input imageAcquisitionFromImageFileInput.txt\n");
  prompt_and_exit(1);
}

printf("Example of how to use openCV to acquire and display images from file\n");
printf("from image file.\n\n");
```

```
    do {

       end_of_file = fscanf(fp_in, "%s", filename);

       if (end_of_file != EOF) {
          printf("\nDisplaying image from image file %s \n",filename);

          strcpy(file_path_and_filename, data_dir);
          strcat(file_path_and_filename, filename);
          strcpy(filename, file_path_and_filename);

          display_image_from_file(filename);
       }
    } while (end_of_file != EOF);

    fclose(fp_in);

    #ifdef ROS
       // Reset terminal
       tcsetattr(STDIN, TCSANOW, &old_term);
    #endif
    return 0;
}
```

Demo

The following code is taken from the imageAcquisitionFromVideoFile example application

See:

```
imageAcquisitionFromVideoFile.h
imageAcquisitionFromVideoFileImplementation.cpp
imageAcquisitionFromVideoFileApplication.cpp
```

To run the example:

```
rosrun module5 imageAcquisitionFromVideoFile
```

```
#include "module5/imageAcquisitionFromVideoFile.h"
/*=========================================================*/
/* 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;
   }
}
```

This should be `WINDOW_AUTOSIZE` for OpenCV 4

Demo

The following code is taken from the imageAcquisitionFromUSBCamera example application

See:

```
imageAcquisitionFromUSBCameraFile.h
imageAcquisitionFromUSBCameraImplementation.cpp
imageAcquisitionFromUSBCameraApplication.cpp
```

To run the example:

```
rosrun module5 imageAcquisitionFromUSBCamera
```

```
#include "module5/imageAcquisitionFromUSBCamera.h"

/*=====================================================*/
/* 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];
   char outputFile[MAX_FILENAME_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 C++

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

      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
```

This is missing in the software provided

This should be WINDOW_AUTOSIZE for OpenCV 4

This is missing in the software provided

```
      #ifdef ROS
         strcpy(outputFile, ros::package::getPath(ROS_PACKAGE_NAME).c_str()); // get the package directory
      #else
         strcpy(outputFile, "..");
      #endif

      strcat(outputFile, "/data/camera_image.png");

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

      imwrite(outputFile, 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
```

```
#include "module5/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);
   }
```

This should be WINDOW_AUTOSIZE for OpenCV 4

This should be IMREAD_COLOR for OpenCV 4

This should be IMREAD_GRAYSCALE for OpenCV 4

```
/* 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) { // defensive: in case 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, 160
         }
         else {
            colourImage.at<Vec3b>(row,col)[channel] = 80 * (col % 3); // test image comprises bands of greyscale 0, 80, 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) {  // defensive: in case the colour image is not a colour image or 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);

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

    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);
}
```

This should be COLOR_BGR2GRAY for OpenCV 4

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
```

```
#include "module5/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 );
```

This should be `WINDOW_AUTOSIZE` for OpenCV 4

This should be `IMREAD_COLOR` for OpenCV 4

```
    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);

    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
```

```cpp
#include "module5/gaussianFiltering.h"

/*
 * 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);
}
```

```
#include "module5/gaussianFiltering.h"

// Global variables to allow access by the display window callback functions

Mat src;
int noise_std_dev             = 0; // default standard deviation for additive Gaussian noise
int gaussian_std_dev          = 0; // default standard deviation for Gaussian filter: filter size = value * 4 + 1

const char* input_window_name     = "Input Image";
const char* processed_window_name = "Gaussian Image";

int view;

int main() {

   const char input_filename[MAX_FILENAME_LENGTH] = "gaussianFilteringInput.txt";
   char input_path_and_filename[MAX_FILENAME_LENGTH];
   char data_dir[MAX_FILENAME_LENGTH];
   char file_path_and_filename[MAX_FILENAME_LENGTH];


   int end_of_file;
   bool debug = true;
   char filename[MAX_FILENAME_LENGTH];

   int const max_noise_std_dev     = 50;
   int const max_gaussian_std_dev  = 5;

   FILE *fp_in;
```

```
printf("Example use of openCV to remove noise using Gaussian filtering.\n\n");

#ifdef ROS
   strcpy(data_dir, ros::package::getPath(ROS_PACKAGE_NAME).c_str()); // get the package directory
#else
   strcpy(data_dir, "..");
#endif

strcat(data_dir, "/data/");
strcpy(input_path_and_filename, data_dir);
strcat(input_path_and_filename, input_filename);

if ((fp_in = fopen(input_path_and_filename,"r")) == 0) {
       printf("Error can't open input file gaussianFilteringInput.txt\n");
  prompt_and_exit(1);
}

do {

   end_of_file = fscanf(fp_in, "%s", filename);

   if (end_of_file != EOF) {
      strcpy(file_path_and_filename, data_dir);
      strcat(file_path_and_filename, filename);
      strcpy(filename, file_path_and_filename);

      src = imread(filename, CV_LOAD_IMAGE_UNCHANGED);
      if (src.empty()) {
         cout << "can not open " << filename << endl;
         prompt_and_exit(-1);
      }
```

This should be IMREAD_UNCHANGED for OpenCV 4

```
        // Create a window for input and display it
        namedWindow(input_window_name, CV_WINDOW_AUTOSIZE );
        imshow(input_window_name, src);

        // Create a window
        namedWindow(processed_window_name, CV_WINDOW_AUTOSIZE );
        resizeWindow(processed_window_name,0,0); // this forces the trackbar to be as small as possible (and to fit in the window)

        // create trackbars; same callback for both
        createTrackbar("Noise",  processed_window_name, &noise_std_dev,    max_noise_std_dev,    processNoiseAndAveraging);
        createTrackbar("Std Dev",processed_window_name, &gaussian_std_dev, max_gaussian_std_dev, processNoiseAndAveraging);

        // Show the image
        processNoiseAndAveraging(0, 0);

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

        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(input_window_name);
        destroyWindow(processed_window_name);
      }
   } while (end_of_file != EOF);

   fclose(fp_in);

   return 0;
}
```

This should be WINDOW_AUTOSIZE for OpenCV 4