Department of Computer Science
University of Bristol

COMS30121 - Image Processing and Computer Vision

www.ole.bris.ac.uk/bbcswebdav/courses/COMS30121_2019_TB-1/index.html

Lab Sheet 01 - Part 1

Introduction to OpenCV Basics

Lab Setup and Getting Started

Tilo Burghardt | tilo@cs.bris.ac.uk

What is OpenCV?

- OpenCV is a library framework for developing computer vision solutions. It is widely used in industry and research.
- It is (mainly) free for both academic and commercial use.
- It has C++, C, Python and Java interfaces and supports Linux, Windows, Mac OS, iOS and Android. (We will support C++ on lab machines only, but you may opt to use your own setup and machine. In either case, note that you will have to present your coursework in the lab or on your own laptop.)
- OpenCV is designed for computational efficiency and with a strong focus on real-time applications.
- Written in optimized C/C++, the library can take advantage of multi-core processing and hardware acceleration.

Objectives of the First Lab Session

- Your first task is to form pairs and register your team on the unit website.
- Once you have done this, setup OpenCV on the lab machines and/or your own machine.
- Finally, start familiarising yourselves with the basics of OpenCV, that is after the lab you should be able to compile and run OpenCV programs, to create, draw into, load, save and display images, and to manipulate pixels.
- The following lab sheets will help you achieve this. Code and scripts for this lab are also available on the unit website.

Setting up OpenCV in the Lab MVB2.11

- First, open a terminal and enter the bash shell by typing:

```
bash
```

- Make sure you type the following four lines before you start using OpenCV or, to avoid this, make sure your .bashrc script in your home directory contains them:

```
export LD_LIBRARY_PATH=/usr/lib64:$LD_LIBRARY_PATH

export CPLUS_INCLUDE_PATH=$CPLUS_INCLUDE_PATH:
/usr/include/opencv2/:/usr/include/opencv

export OPENCV_CFLAGS=-I/usr/include/opencv2/

export O_LIBS="-L/usr/lib64/ -lopencv_core
-lopencv_imgproc -lopencv_highgui -lopencv_ml
-lopencv_video -lopencv_features2d -lopencv_calib3d
-lopencv_objdetect -lopencv_contrib -lopencv_legacy
-lopencv_flann"
```

Setting up OpenCV in the Lab MVB2.11

- If you edited your bashrc script, make sure you refresh your environment via:

```
source ~/.bashrc
```

- Now create a project directory and change to it:

```
mkdir mydir
cd mydir
```

- Now download the HelloOpenCV program hello.cpp from the course website.

A look inside the hello.cpp program...

hello.cpp

```
#include <opencv/cv.h>          //you may need to
#include <opencv/highgui.h>     //adjust import locations
#include <opencv/cxcore.h>      //depending on your machine setup
using namespace cv;

int main() {
  //create a black 256x256, 8bit, gray scale image in a matrix container
  Mat image(256, 256, CV_8UC1, Scalar(0));
  //draw white text HelloOpenCV!
  putText(image, "HelloOpenCV!", Point(70, 70),
    FONT_HERSHEY_COMPLEX_SMALL, 0.8, cvScalar(255), 1, CV_AA);
  //save image to file
  imwrite("myimage.jpg", image);
  //construct a window for image display
  namedWindow("Display window", CV_WINDOW_AUTOSIZE);
  //visualise the loaded image in the window
  imshow("Display window", image);
  //wait for a key press until returning from the program
  waitKey(0);
  //free memory occupied by image
  image.release();
  return 0;
}
```

Compiling OpenCV Code in the Lab MVB2.11

- Now you are ready to compile your program using:

```
g++ hello.cpp /usr/lib64/libopencv_core.so.2.4
/usr/lib64/libopencv_highgui.so.2.4
```

- Consider using make. You can run your program via:

```
./a.out
```

- The program will create and display an image called myimage.jpg:
- Finally, have a look at the following simple sample programs provided. View, compile and run them...

First Steps in OpenCV(C++): Load and Display an Image

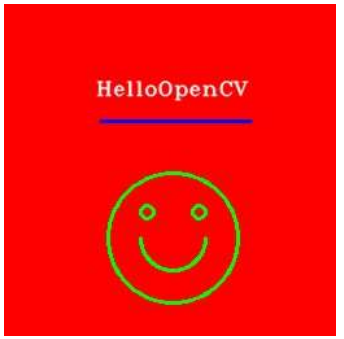
display.cpp

```
#include <opencv/cv.h>          //you may need to
#include <opencv/highgui.h>     //adjust import locations
#include <opencv/cxcore.h>      //depending on your machine setup

using namespace cv;             //make available OpenCV namespace

int main() {
  //declare a matrix container to hold an image
  Mat image;
  //load image from a file into the container_
  image = imread("myimage.jpg", CV_LOAD_IMAGE_UNCHANGED);
  //construct a window for image display
  namedWindow("Display window", CV_WINDOW_AUTOSIZE);
  //visualise the loaded image in the window
  imshow("Display window", image);
  //wait for a key press until returning from the program
  waitKey(0);
  //free memory occupied by image
  image.release();
  return 0;
}
```

First Steps in OpenCV(C++): Create, Draw and Save

draw.cpp

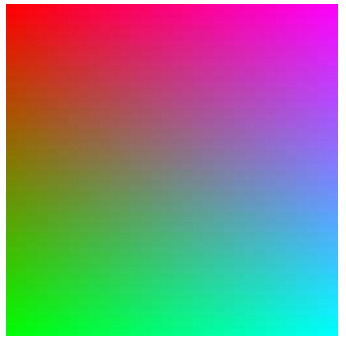
```
#include [...]
using namespace cv;

int main() {
  //create a red 256x256, 8bit, 3channel BGR image in a matrix container
  Mat image(256, 256, CV_8UC3, Scalar(0, 0, 255));
  //put white text HelloOpenCV
  putText(image, "HelloOpenCV", Point(70, 70),
    FONT_HERSHEY_COMPLEX_SMALL, 0.8, cvScalar(255, 255, 255), 1, CV_AA);
  //draw blue line under text
  line(image, Point(74, 90), Point(190, 90), cvScalar(255, 0, 0),2);
  //draw a green smile
  ellipse(image, Point(130, 180), Size(25,25), 180, 180, 360,
    cvScalar(0, 255, 0), 2);
  circle(image, Point(130, 180), 50, cvScalar(0, 255, 0), 2);
  circle(image, Point(110, 160), 5, cvScalar(0, 255, 0), 2);
  circle(image, Point(150, 160), 5, cvScalar(0, 255, 0), 2);
  //save image to file
  imwrite("myimage.jpg", image);
  //free memory occupied by image
  image.release();
  return 0;
}
```



First Steps in OpenCV(C++): Access and Set Pixel Values

pixels.cpp

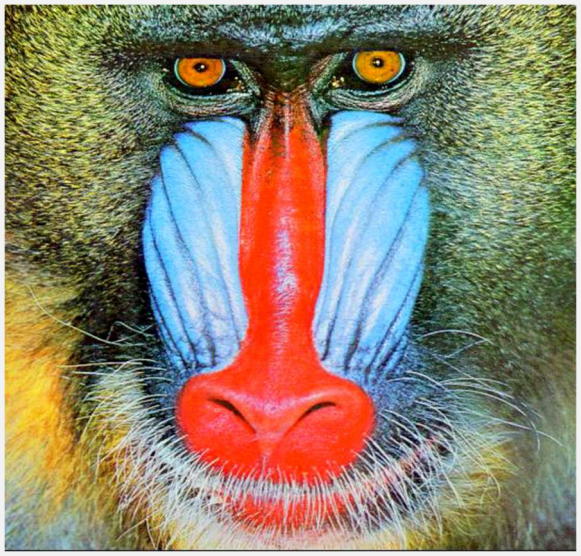
```
#include [...]
using namespace cv;

int main() {
  //create a black 256x256, 8bit, 3channel BGR image in a matrix container
  Mat image(256, 256, CV_8UC3, Scalar(0, 0, 0));
  //set pixels to create colour pattern
  for(int y = 0; y < image.rows; y++) //go through all rows (or scanlines)
    for (int x = 0; x < image.cols; x++) { //go through all columns
      image.at<Vec3b>(y, x)[0] = x; //blue
      image.at<Vec3b>(y, x)[1] = y; //green
      image.at<Vec3b>(y, x)[2] = 255 - image.at<Vec3b>(y, x)[1]; //red
    }
  //construct a window for image display
  namedWindow("Display window", CV_WINDOW_AUTOSIZE);
  //visualise the loaded image in the window
  imshow("Display window", image);
  //wait for a key press until returning from the program
  waitKey(0);
  //free memory occupied by image
  image.release();
  return 0;
}
```



Department of Computer Science
University of Bristol

COMS30121 - Image Processing and Computer Vision

www.ole.bris.ac.uk/bbcswebdav/courses/COMS30121_2019_TB-1/index.html



Lab Sheet 01 - Part 2

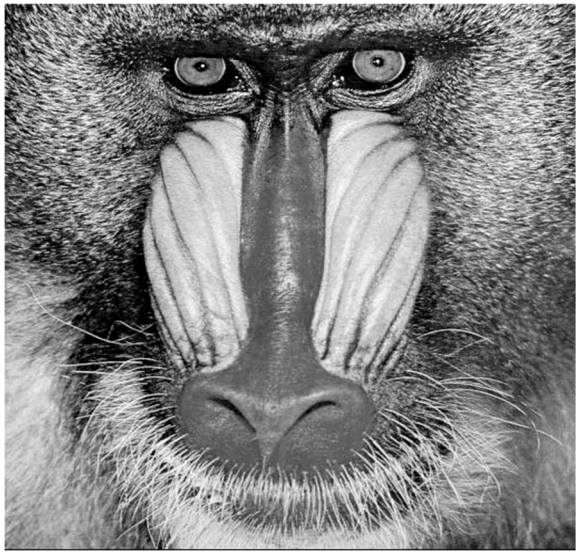
Introduction to OpenCV Basics

Pixel Manipulation and Thresholding

Tilo Burghardt | tilo@cs.bris.ac.uk

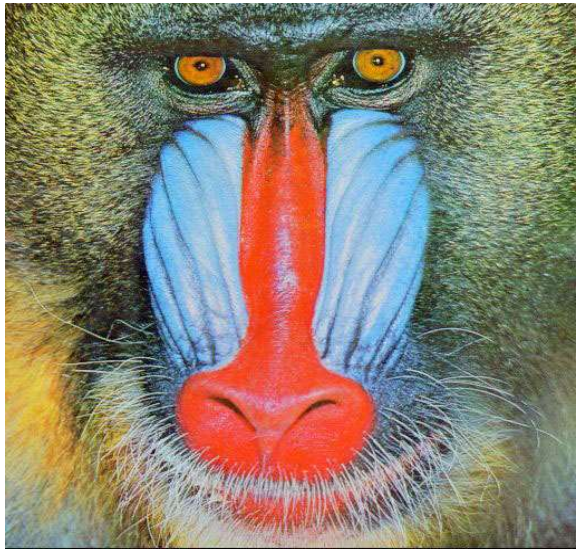
Implementing Thresholding

- Now that you are able to handle images, your next task is to write an OpenCV-based program that loads the mandrill.jpg greyscale image and that, pixel by pixel, sets all pixels above a certain value (maybe start with 128) to white (255) and all pixels equal or below the value to black (0).
- Experiment with different thresholding values and examine the resulting images. Can you highlight certain parts of the face (e.g. the nose, the eyes) with one or more specific thresholds?
- Compare your results to the output of the inbuilt OpenCV function **`threshold`**.



Optional: Thresholding Colour Images

- Whilst in greyscale images the brightness of a pixel is usually represented as a single byte (`unsigned char`), colour images use three bytes to store information for one pixel. Bytes represent the BLUE, GREEN and RED channel in this order.
- If you have time left in this lab (or just for interest), try to implement thresholding of the red, green and/or blue channels to highlight facial components in mandrillRGB.jpg, which now contains colour information.



- Sample answers are available at thr.cpp and colourthr.cpp if you are stuck.
- Also check the OpenCV function `inRange`.

First Steps in OpenCV(C++): Basic Thresholding

thr.cpp

```
#include [...]
using namespace cv;

int main() {

  // Read image from file
  Mat image = imread("mandrill.jpg", 1);

  // Convert to grey scale
  Mat gray_image;
  cvtColor(image, gray_image, CV_BGR2GRAY);

  // Threshold by looping through all pixels
  for (int y = 0; y<gray_image.rows; y++) {
    for (int x = 0; x<gray_image.cols; x++) {
      uchar pixel = gray_image.at<uchar>(y, x);
      if (pixel>128) gray_image.at<uchar>(y, x) = 255;
      else gray_image.at<uchar>(y, x) = 0;
  } }

  //Save thresholded image
  imwrite("thr.jpg", gray_image);

  return 0;
}
```

First Steps in OpenCV(C++): RGB Thresholding

colourthr.cpp

```
#include [...]
using namespace cv;

int main() {

  // Read image from file
  Mat image = imread("mandrillRGB.jpg", 1);

  // Threshold by looping through all pixels
  for (int y = 0; y<image.rows; y++) {
    for (int x = 0; x<image.cols; x++) {
      uchar pixelBlue = image.at<Vec3b>(y, x)[0];
      uchar pixelGreen = image.at<Vec3b>(y, x)[1];
      uchar pixelRed = image.at<Vec3b>(y, x)[2];
      if (pixelBlue>200) {
        image.at<Vec3b>(y, x)[0] = 255;
        image.at<Vec3b>(y, x)[1] = 255;
        image.at<Vec3b>(y, x)[2] = 255;
      } else {
        image.at<Vec3b>(y, x)[0] = 0;
        image.at<Vec3b>(y, x)[1] = 0;
        image.at<Vec3b>(y, x)[2] = 0;
  } } }

  //Save thresholded image
  imwrite("colourthr.jpg", image);

  return 0;
}
```