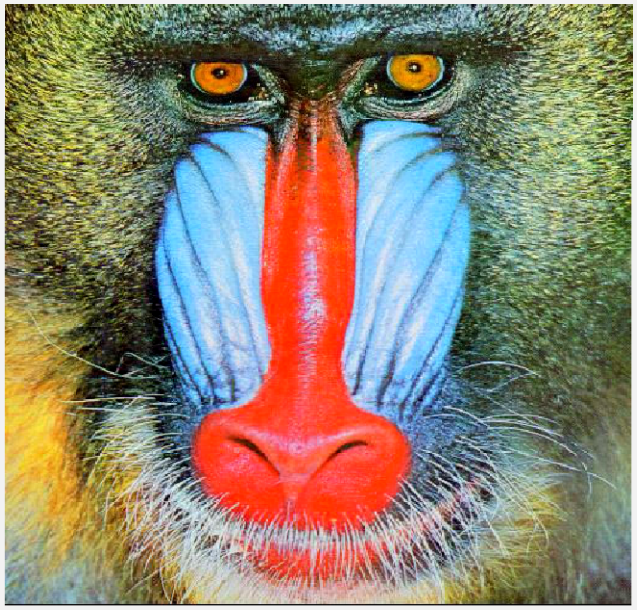
Department of Computer Science
University of Bristol

COMS30121 - Image Processing and Computer Vision

www.cs.bris.ac.uk/Teaching/Resources/COMS30121



Lab Sheet 01 - Part 2

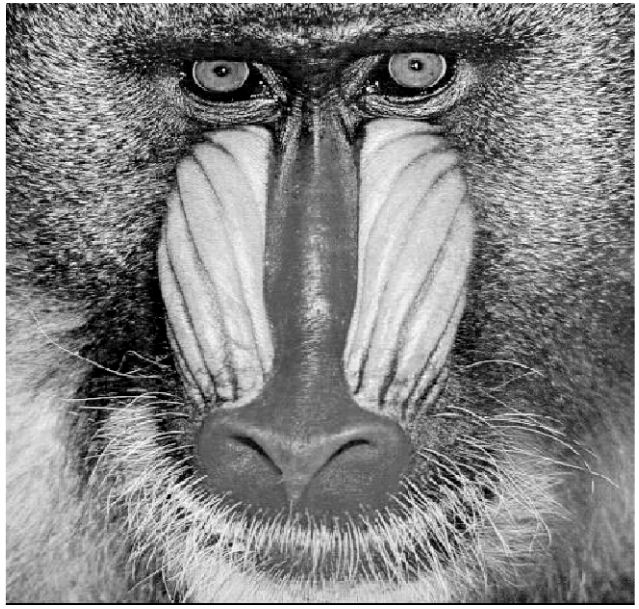
Introduction to OpenCV Basics

Pixel Manipulation and Thresholding

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

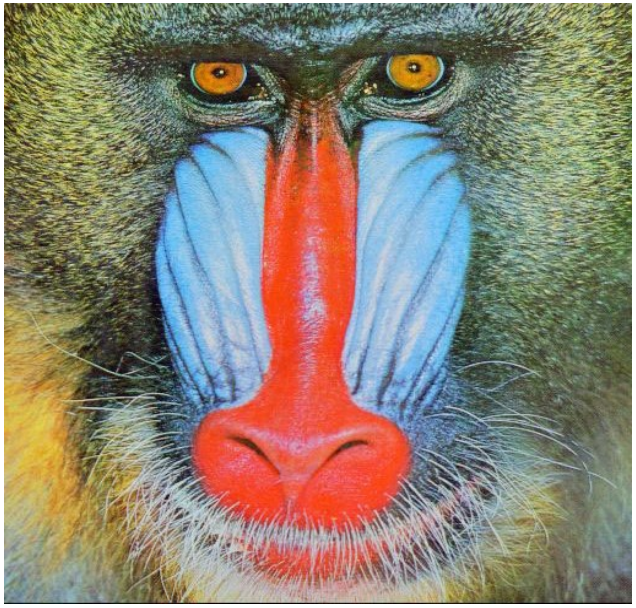
Implementing Thresholding

- 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`.

Running OpenCV Code in the Lab MVB2.11

- Remember, you are ready to compile your program using:

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

- Note, that depending on the functionality used you may need to add more libraries to the above call and include further headers into your program
- Explore the directories `/usr/lib64` and `/usr/include/opencv` and `/usr/include/opencv2` to have a look at the different libraries and header files available
- As always, you can run your program via:

```
./a.out
```

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;
}
```