Answers for the exercises in Chapter 5

Andre L. C. Barczak

2015

1 Exercise 1

Write a program that shows three greyscale images representing the three components R, G and B of a colour image. (hint: just create three greyscale images and copy each component to one of them).

Listing 1: Three components

```
1  #include "opencv2/imgproc/imgproc.hpp"
2  #include "opencv2/highgui/highgui.hpp"
3  #include <stdio.h>
4
5  using namespace cv;
6  using namespace std;
7
8  #define Mpixel(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x)
       ]
9  #define MpixelB(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x
       )*((image).channels())]
10 #define MpixelG(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x
       )*((image).channels())+1]
11 #define MpixelR(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x
       )*((image).channels())+2]
12
13 Mat imageinput;
14
15 int main( int argc, char** argv )
16 {
17          namedWindow("Original",0);
18          namedWindow("Red channel",0);
19          namedWindow("Green channel",0);
20          namedWindow("Blue channel",0);
21          if (argc != 2) {printf("needs an input image\n");exit(0);}
22          imageinput = imread( argv[1], 1);
23          imshow("Original",imageinput);
24
25          Mat imagered(imageinput.rows,imageinput.cols,CV_8UC1, Scalar::all(0));
26          Mat imagegreen(imageinput.rows,imageinput.cols,CV_8UC1, Scalar::all(0));
```

```
27          Mat imageblue(imageinput.rows,imageinput.cols,CV_8UC1, Scalar::all(0));
28          for(int x=0;x<imageinput.cols;x++){
29            for(int y=0;y<imageinput.rows;y++){
30              Mpixel(imagered,x,y)=MpixelR(imageinput,x,y);
31              Mpixel(imagegreen,x,y)=MpixelG(imageinput,x,y);
32              Mpixel(imageblue,x,y)=MpixelB(imageinput,x,y);
33            }
34          }
35
36          imshow("Red channel",imagered);
37          imshow("Green channel",imagegreen);
38          imshow("Blue channel",imageblue);
39
40          waitKey(0);
41  }
```

Figure 1: Red, green and blue channels from peppers.bmp .

2 Exercise 2

Write a program that converts RGB to HSV and show the three components separately. Remember that the range of the HSV space is different than that for RGB. Adopt your own conversion table (e.g., for H, 0 is converted to 0, and 360 is converted to 255).

Answer: Using opencv, we need to use the convert function (`cvtColor(imageinput, image2, CV_BGR2HSV)`). Be very careful with the CV_BGR2HSV (NOT CV_RGB2HSV, which is one of the valid options, but the images by default are BGR).

If you compare the pixels with tools such as GIMP, you will note that V and S are given in percentage (so 255 = 100%), but H is given in angle (from 0 to 360 degrees). OpenCV uses H from 0 to 180 only, so one has to multiply the OpenCV H values by two to obtain the angle of the Hue. There values above 180 for H are invalid in the OpenCV HSV images.

To show how the colours can be classified under HSV, an image was produced with the following thresholds:

- Red if the Hue < 10 or > 160
- Green if $45 <$ Hue < 65
- Yellow if $25 <$ Hue < 45
- Orange if $10 <$ Hue < 25
- Black if saturation S < 15

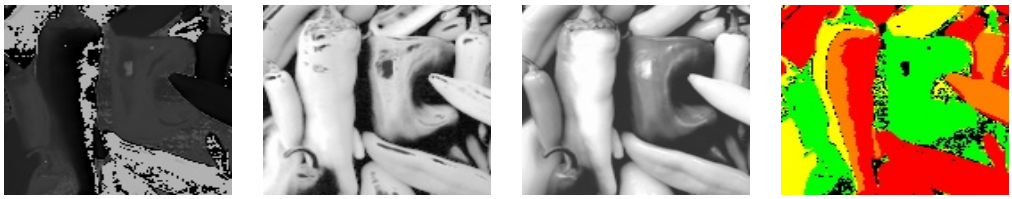

Figure 2: Hue, Saturation and Value in grayscale, colour classification based on Hue.

Listing 2: HSV components

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdio.h>

using namespace cv; using namespace std;

#define Mpixel(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x)
    ]
#define MpixelB(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x
    )*((image).channels())]
#define MpixelG(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x
    )*((image).channels())+1]
#define MpixelR(image,x,y) ( (uchar *) ( ((image).data) + (y)*((image).step) ) ) [(x
    )*((image).channels())+2]

Mat imageinput;

int main( int argc, char** argv )
{
        namedWindow("HSV",0);     namedWindow("H",0);
        namedWindow("S",0);       namedWindow("V",0);
        if (argc != 2) {printf("needs an input image\n");exit(0);}
        imageinput = imread( argv[1], 1);

        Mat image2(imageinput.rows,imageinput.cols,CV_8UC3, Scalar::all(0));
        cvtColor(imageinput, image2, CV_BGR2HSV);
        imshow("HSV",image2);
        waitKey(0);

        Mat imageH(imageinput.rows,imageinput.cols,CV_8UC1, Scalar::all(0));
        Mat imageS(imageinput.rows,imageinput.cols,CV_8UC1, Scalar::all(0));
        Mat imageV(imageinput.rows,imageinput.cols,CV_8UC1, Scalar::all(0));
        for(int x=0;x<imageinput.cols;x++){
          for(int y=0;y<imageinput.rows;y++){
            Mpixel(imageV,x,y)=MpixelR(image2,x,y);
            Mpixel(imageS,x,y)=MpixelG(image2,x,y);
```

```
33                  Mpixel(imageH,x,y)=MpixelB(image2,x,y);
34              }
35          }
36          imshow("H",imageH);
37          imshow("S",imageS);
38          imshow("V",imageV);
39          //color classification
40          Mat image3(imageinput.rows,imageinput.cols,CV_8UC3, Scalar::all(0));
41          for(int x=0;x<image3.cols;x++){
42              for(int y=0;y<image3.rows;y++){
43                  //RED
44                  if(Mpixel(imageH,x,y)>160 || Mpixel(imageH,x,y)<10){
45                  MpixelR(image3,x,y)=255;
46                  MpixelG(image3,x,y)=0;
47                  MpixelB(image3,x,y)=0;
48                  }
49                  //GREEN
50                  if(Mpixel(imageH,x,y)>45 && Mpixel(imageH,x,y)<65){
51                  MpixelR(image3,x,y)=0;
52                  MpixelG(image3,x,y)=255;
53                  MpixelB(image3,x,y)=0;
54                  }
55                  //YELLOW
56                  if(Mpixel(imageH,x,y)>25 && Mpixel(imageH,x,y)<45){
57                  MpixelR(image3,x,y)=255;
58                  MpixelG(image3,x,y)=255;
59                  MpixelB(image3,x,y)=0;
60                  }
61                  //ORANGE
62                  if(Mpixel(imageH,x,y)>=10 && Mpixel(imageH,x,y)<25){
63                  MpixelR(image3,x,y)=255;
64                  MpixelG(image3,x,y)=128;
65                  MpixelB(image3,x,y)=0;
66                  }
67                  //BLACK, low saturation
68                  if(Mpixel(imageS,x,y)<15){
69                      MpixelR(image3,x,y)=0;
70                      MpixelG(image3,x,y)=0;
71                      MpixelB(image3,x,y)=0;
72                  }
73              }
74          }
75          imshow("HSV",image3);
76          waitKey(0);
77  }
```

3 Exercise 3

Write a program that segments images by colour. Use a range of values that is equivalent to a cubic region in the RGB space (i.e., simple thresholds for each colour component). Test your code using the images *peppers.png* and *peppers2.png* to try to separate the green peppers from the red ones.

Answer: To show how the colours can be classified under RGB, an image was produced with the following thresholds:

- Red if $R > 100$ and $G < 100$
- Green if $R < 100$ and $G > 100$
- Yellow if $R < 200$ and $G > 200$
- Orange if $R > 240$ and $G < 200$

Listing 3: RGB classification

```
Mat imageinput;
int main( int argc, char** argv )
{
        namedWindow("Original",0);
        namedWindow("RGB classified",0);
        if (argc != 2) {printf("needs an input image\n");exit(0);}
        imageinput = imread( argv[1], 1);
        imshow("Original",imageinput);
        Mat image3(imageinput.rows,imageinput.cols,CV_8UC3, Scalar::all(0));
        for(int x=0;x<image3.cols;x++){
          for(int y=0;y<image3.rows;y++){
            //RED
            if(MpixelR(imageinput,x,y)>100 && MpixelG(imageinput,x,y)<100){
            MpixelR(image3,x,y)=255;
            MpixelG(image3,x,y)=0;
            MpixelB(image3,x,y)=0;
            }
            //GREEN
            if(MpixelG(imageinput,x,y)>100 && MpixelR(imageinput,x,y)<100){
            MpixelR(image3,x,y)=0;
            MpixelG(image3,x,y)=255;
            MpixelB(image3,x,y)=0;
            }
            //YELLOW
            if(MpixelR(imageinput,x,y)>200 && MpixelG(imageinput,x,y)>200){
            MpixelR(image3,x,y)=255;
            MpixelG(image3,x,y)=255;
            MpixelB(image3,x,y)=0;
            }
            //ORANGE
            if(MpixelR(imageinput,x,y)>240 && MpixelG(imageinput,x,y)<200){
```

```
32                  MpixelR(image3,x,y)=255;
33                  MpixelG(image3,x,y)=128;
34                  MpixelB(image3,x,y)=0;
35                  }
36              }
37          }
38          imshow("RGB classified",image3);
39          waitKey(0);
40  }
```

Figure 3: RGB classification.

4 Exercise 4

Write a program that segments skin colour using two different colour systems, RGB and HSV. Collect at least 10 different images from the Internet and test different parameters, until the segmentation is comparable to the results done by hand. Which colour system achieved the best results?

Answer: Use the camera programs available on Stream (camera_HSV_filter.c and camera_RGB_filter.c). Note that in RGB you need to adjust three parameters for skin colour, while in HSV the threshold for the value V can remain open (from 0 to 255). For that reason, HSV classification is more robust against changes in illumination than RGB classification, although the HSV classification looks a bit noisy.

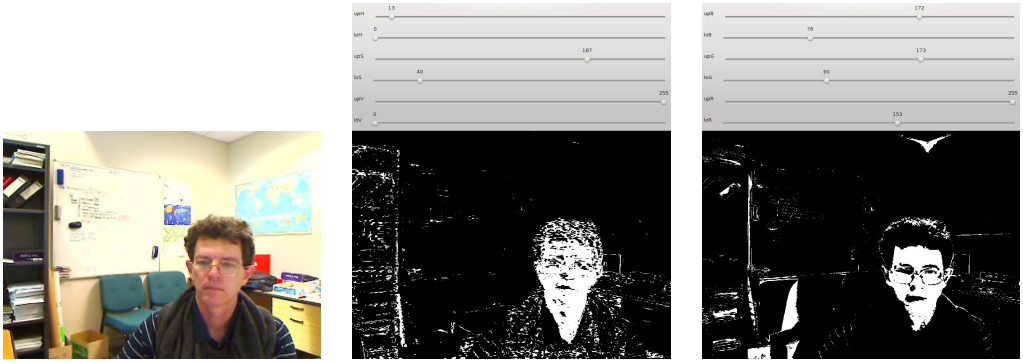

Figure 4: Using the camera to classify colour in HSV and RGB.