Experiment 1: Getting to grips with OpenCV

Adrian F. Clark

This first laboratory session gives you some familiarity with OpenCV. You will use it to calculate some statistics that describe an image. You will them calculate and draw a histogram and make a number of useful improvements to it which should help you with the second experiment.

Contents

1 Introduction

To perform this experiment, you will first need to download an unpack a a zip-file which contains the relevant software and test images. If you download the file into your "`Downloads`" directory, then the commands

```
mkdir vision-expt-01
cd vision-expt-01
unzip ~/Downloads/expt01.zip
```

should be enough to unpack everything you need in a useful place. It would be a good idea to review the files you have unpacked before proceeding.

2 Displaying images

Our Linux machines have on them an antique but astonishingly useful image display program known as `xv`. You can display a series of images with an incantation such as

```
  xv *.jpg
```

This displays the first image given on the command line and waits for your input. With the displayed image selected, the space bar moves you to the next image and the ⟨backspace⟩ key (which might be labelled ⟨delete⟩ on your keyboard) to the previous one. These features make reviewing a set of images both quick and easy to do, and with much less surrounding clutter than one experiences with an image editor.

However, `xv` can do more. Click the right mouse button and a dialogue box appears that lets you choose other images (double-click the filename), do simple operations such as rotating them, and so on. A second right-mouse click closes the dialogue box. You can select a region of the image with the mouse (left mouse click and drag), then type "`c`" to crop it from the image. If you type "`e`", a colour editor appears which allows you to perform surprisingly detailed modifications to the appearance of an image — "`Brite`" is useful on under-exposed images, for example.

If you want to use `xv` on your own Linux system, it is probably enough just to take a copy of it. The author has a working version for macOS too, though it requires Apple's (free) XQuartz X11 server to be installed. I do not believe it is available for Windows.

3 Summarizing images

The zip-file for this experiment contains a Python program called `summarize` which should be executable. You will run it with an incantation like

```
  ./summarize *.jpg
```

to work through all the JPEG-format files you have been provided with. However, don't do that just yet; instead, read through it to see what it does. When you are happy that you understand it, run it on three of the test images:

```
2006-10-23-001.jpg
2006-10-23-002.jpg
2006-10-23-003.jpg
```

and interpret the statistical values that are output as you look at them.

You will have seen that `summarize` not only outputs statistical values, it computes the histogram and outputs the values to files. The next step is to plot these.

4 Plotting the histograms

There are several ways in which you could plot the histograms. You could use `pylab`, part of Matplotlib, which is a plotting library directly callable from Python; or you could transfer the files into (say) Excel; or you could sketch them by hand on paper. However, the way that is probably most helpful for the future is to use Gnuplot, a general-purpose graph-plotting package available for pretty much every computing platform — the author even has it on his phone! Gnuplot expects the data it is to plot to be stored with an (x, y) pair on each line with white space separating the values, exactly the format in which `summarize` writes out values.

To run Gnuplot, it is only a matter of typing its name on the command line:

```
> gnuplot

        G N U P L O T
        Version 5.2 patchlevel 8    last modified 2019-12-01

        Copyright (C) 1986-1993, 1998, 2004, 2007-2019
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help FAQ"
        immediate help:   type "help"  (plot window: hit 'h')

Terminal type is now 'qt'
gnuplot>
```

The introductory message was produced on the author's Mac, so you will definitely see something a little different on a Linux box. In particular, the "terminal type" (the way it presents its output) will probably be different. However, the commands that you type and the results you obtain should be identical on all operating systems.

We start by defining the annotation of the plot and then tell Gnuplot to plot the actual data; you should make sure you understand what each of the following commands does.

```
unset key
set grid
set title "Histogram of 2006-10-23-001.jpg"
set xlabel "pixel value"
set ylabel "frequency"
plot "2006-10-23-001.jpg.dat"
```

The resulting plot will appear as a series of +-shaped points. This is acceptable but a histogram is normally drawn with vertical bars; the appropriate incantation is:

```
plot "2006-10-23-001.jpg.dat" with boxes
```

To restrict the range of the x-axis to 0–255, type

```
set xrange [0:255]
replot
```

You should save Gnuplot outputs for each of the test images in your record of the experiment. You can do this by taking a screenshot (using your phone or, better, using software) but the most elegant way is to use the `set terminal` and `set output` commands with appropriate parameters to choose the output filetype and output filename. Typing:

```
help set term
```

at the `gnuplot>` prompt will let you access the relevant online help. The author produces PDF output a lot (you'll find many examples in the lecture notes) using commands like

```
set term pdf
set output "myplot.pdf"
```

The first of these tells Gnuplot to generate PDF output while the second says where it should go.

Look at the histograms of the three test images and identify what characteristics determine whether the corresponding image is under-exposed, well-exposed or over-exposed. Discuss your findings with a demonstrator.

When you have worked through the complete script, you'll see how to produce multiple plots on a single set of axes. Have a go at reproducing the plot shown in Figure 1.

5 Improving the histogram routine

Display the image `m51.png`, an image of a Messier galaxy, with `xv`. It will initially appear more or less black, though there is some faint grey structure in it. Then bring up `xv`'s colour editor and click the `Random` button: a lot more structure should appear, albeit somewhat funky. It seems that this image has only a few low-valued grey levels occupied; in fact, that is not the whole truth as you'll see at the end of this experiment.

With only a few grey values being occupied, a significantly better way of visualizing the useful histogram is to find the minimum and

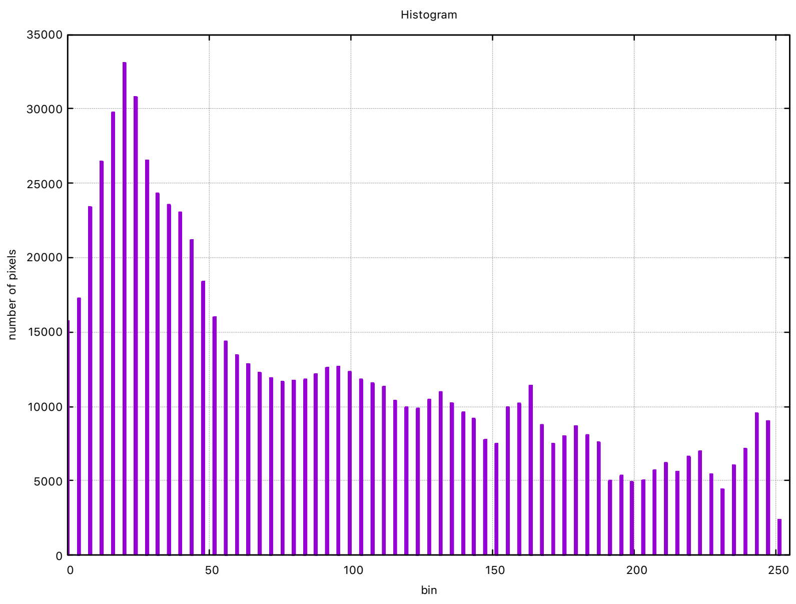

Figure 1: Histogram of the three images on the same axes

maximum values of the image, I_l and I_h say, and "contrast stretch" a value v in the image into he range 0–255:

$$255 \times \frac{v - I_l}{I_h - I_l}$$

If you find this difficult to understand, take a look at my lecture notes that introduce OpenCV, discuss it with your friends, or speak to a demonstrator.

The `summarize` program already shows you how to calculate I_l and I_h; and with these, you need make only a one-line change to the histogram routine to make it contrast-stretch the data. Plot the resulting histogram, taking care to produce sensible x-axis values.

In fact, 256 bins in a histogram is rather a large number; with this contrast-stretching in place, you should be able to reduce `MAXGREY` to 64, which is generally a more useful value.

6 Per-channel colour histograms

The histograms you have looked at to date plot the red, green and blue channels of an image together but it is normally more useful to histogram each of them separately. Modify the `histogram` routine so that:

- you pass in the channel of the image to be plotted;
- it saves its output in a filename which contains the channel number;

and then call the it for each of the three channels. Plot the result of processing the image `sx.jpg`, which I found somewhere on the University's website.

You can tell Gnuplot to plot multiple plots on the same axes by separating the files with commas, something like

```
plot "sx.jpg.c0" with boxes, "sx.jpg.c1" with boxes, "sx.jpg.c2" with boxes
```

We shall return to these per-channel histograms in a later experiment.

7 The whole truth about `m51.png`

If you display `m51.png` using `xv` and type "`i`" to bring up its information window, you'll see that it is identified as being a *16-bit* image. If you then use the author's EVE software on that image in a three-line program:

```
import eve
im = eve.image ("m51.png")
eve.statistics (im, output=True)
```

you'll see that the maximum value is 10106. Now,

$$\left\lfloor \frac{10106}{256} \right\rfloor = 39$$

so what OpenCV must be doing is reading in the 16-bit image and retaining only the top 8 bits of each pixel — but it isn't telling the user that it is doing so, which strikes the author as being very naughty. In fact, this propensity for doing things without telling the user was one of the prime motivators for writing EVE: the author wanted to check that OpenCV was genuinely doing what its documentation said it does. Having a good idea of what is going on, rather than what simply appears to be the case, has proven invaluable on a number of occasions — if you feel inclined, ask him to tell you about "the coral reef images."