pyimagesearch
be awesome at building image search engines

Clever Girl: A Guide to Utilizing Color Histograms for Computer Vision and Image Search Engines

by **Adrian Rosebrock** on January 22, 2014 in **Image Descriptors**

> *It's…it's a histogram. – Dr. Grant*

Okay. So maybe that isn't the *exact* quote. But I think Dr. Grant would have been equally thrilled had he known the power of color histograms.

And more importantly, when the power goes out, *the histograms don't eat the tourists.*

So, what exactly is a histogram? A histogram represents the distribution of colors in an image. It can be visualized as a graph (or plot) that gives a high-level intuition of the intensity (pixel value) distribution. We are going to assume a RGB color space in this example, so these pixel values will be in the range of 0 to 255. If you are working in a different color space, the pixel range may be different.

When plotting the histogram, the X-axis serves as [illegible] bins, then we are effectively counting the number [illegible] use only 2 (equally spaced) bins, then we are cou[illegible]

`[0, 128)` or `[128, 255]`. The number of pixels binned to the X-axis value is then plotted on the Y-axis.

Looking for the source code to this post?
Jump right to the downloads section.

OpenCV and Python versions:
This example will run on **Python 2.7** and **OpenCV 2.4.X/OpenCV 3.0+**.

By simply examining the histogram of an image, you get a general understanding regarding the contrast, brightness, and intensity distribution.

This post will give you an OpenCV histogram exa

Application to Image Sea

In context of image search engines, histograms c
used to quantify an image and compare it to othe ge
search engines, we make the assumption that im lly
similar. I will talk more about this assumption in th
for the time being, let's go ahead and assume tha ar
content.

Comparing the "similarity" of color histograms car es
include: Euclidean, correlation, Chi-squared, inter
to use the Chi-squared distance, but the choice is
analyzed. No matter which distance metric you us
histograms.

Touring Jurassic Park

Let's imagine that we were along with Dr. Grant and company on their first Jurassic Park tour. We brought along our cell phone to document the entire experience (and let's also pretend that camera phones were a "thing" back then). Assuming we didn't pull a Dennis Nedry a have our face eaten by a Dilophosaurus, we could later download the pictures from our smartphones to our computers and compute histograms for each of the images.

At the very beginning of the tour we spent a lot of times in the labs, learning about DNA and witnessing the hatching of a baby velociraptor. These labs have a lot of "steel" and "gray" colors to them. Later on, we got into our jeeps and drove into the park. The park itself is a jungle — lots of green colors.

So based on these two color distributions, which one do you think the Dr. Grant image above is more similar to?

Well, we see that there is a fair amount of greenery in the background of the photo. In all likelihood, the color distribution of the Dr. Grant photo would be more "similar" to our pictures taken during the jungle tour vs. our pictures taken in the lab.

Using OpenCV to Compute Histograms

Now, let's start building some color histograms of

We will be using the `cv2.calcHist` function in to any code examples, let's quickly review the functi

```
cv2.calcHist(images, channels, mask,
```

1. **images:** This is the image that we want to c `[myImage]`.
2. **channels:** A list of indexes, where we specif histogram for. To compute a histogram of a g a histogram for all three red, green, and blue
3. **mask:** I haven't covered masking yet in this with the same shape as our original image, v pixels with a value greater than zero are incl allow us to only compute a histogram for a p a value of `None` for the mask.
4. **histSize:** This is the number of bins we want to use when computing a histogram. Again, this is a list, one for each channel we are computing a histogram for. The bin sizes do not all have to be the same. Here is an example of 32 bins for each channel: `[32, 32, 32]`.
5. **ranges:** The range of possible pixel values. Normally, this is `[0, 256]` for each channel, but if you are using a color space other than RGB (such as HSV), the ranges might be different.

Now that we have an understanding of the `cv2.calcHist` function, let's write some actual code.

Load Our Image — Python

```python
# import the necessary packages
from matplotlib import pyplot as plt
import numpy as np
import argparse
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True, help = "Path to the image")
args = vars(ap.parse_args())

# load the image and show it
image = cv2.imread(args["image"])
cv2.imshow("image", image)
```

This code isn't very exciting yet. All we are doing is importing the packages we will need, setting up an argument parser, and loading our image.

Computing a Grayscale Histogram using Python and OpenCV

```python
# convert the image to grayscale and create a histogram
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("gray", gray)
hist = cv2.calcHist([gray], [0], None, [256], [0, 256])
plt.figure()
plt.title("Grayscale Histogram")
plt.xlabel("Bins")
plt.ylabel("# of Pixels")
plt.plot(hist)
plt.xlim([0, 256])
```

Now things are getting a little more interesting. O colorspace to grayscale. Line 4 computes the act of the code up with the function documentation al grayscale image. A grayscale image has only one `channels`. We don't have a mask, so we set the histogram, and the possible values range from 0

A call to `plt.show()` displays:

Figure 1: Dr. Grant grayscale histogram.

Not bad. How do we interpret this histogram? Well, the bins (0-255) are plotted on the X-axis. And the Y-axis counts the number of pixels in each bin. The majority of pixels fall in the range of ~50 to ~125. Looking at the right tail of the histogram, we see very few pixels in the range 200 to 255. This means that there are very few "white" pixels in the image.

Now that we've seen at a grayscale histogram, let's look at what I call a "flattened" color histogram:

Computing a Flattened Color Histogram using Python and OpenCV

```python
# grab the image channels, initialize the tuple of colors,
# the figure and the flattened feature vector
chans = cv2.split(image)
colors = ("b", "g", "r")
plt.figure()
plt.title("'Flattened' Color Histogram")
plt.xlabel("Bins")
plt.ylabel("# of Pixels")
features = []

# loop over the image channels
for (chan, color) in zip(chans, colors):
    # create a histogram for the current
```

```
14      # concatenate the resulting histograms for each
15      # channel
16      hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
17      features.extend(hist)
18
19      # plot the histogram
20      plt.plot(hist, color = color)
21      plt.xlim([0, 256])
22
23  # here we are simply showing the dimensionality of the
24  # flattened color histogram 256 bins for each channel
25  # x 3 channels = 768 total values -- in practice, we would
26  # normally not use 256 bins for each channel, a choice
27  # between 32-96 bins are normally used, but this tends
28  # to be application dependent
29  print "flattened feature vector size: %d" % (np.array(features).flatten().shape)
```

There's definitely more code involved in computi
histogram. Let's tear this code apart and get a be

- **Lines 3-4:** The first thing we are going to do
green, and red. Normally, we read this is a re
image as a NumPy array in reverse order: B
tuple of strings representing the colors.
- **Lines 5-9:** Here we are just setting up our P
histograms.
- **Line 12:** Let's start looping over the channel
- **Line 16-17:** We are now computing a histogr
as computing a histogram for a single chann
color histogram to our features list.
- **Line 20:** Plot the histogram using the curren
- **Line 29:** Here we are just examining the sha
"flattened" histogram not because the (1) his
flatten() method. I call this a "flattened" histog
counts. Later, we explore multi-dimensional histograms (2D and 3D). A flattened histogram is simply the histogram *for each individual channel concatenated together*.

Now let's plot our color histograms:

Free 21-day crash course on computer vision & image search engines

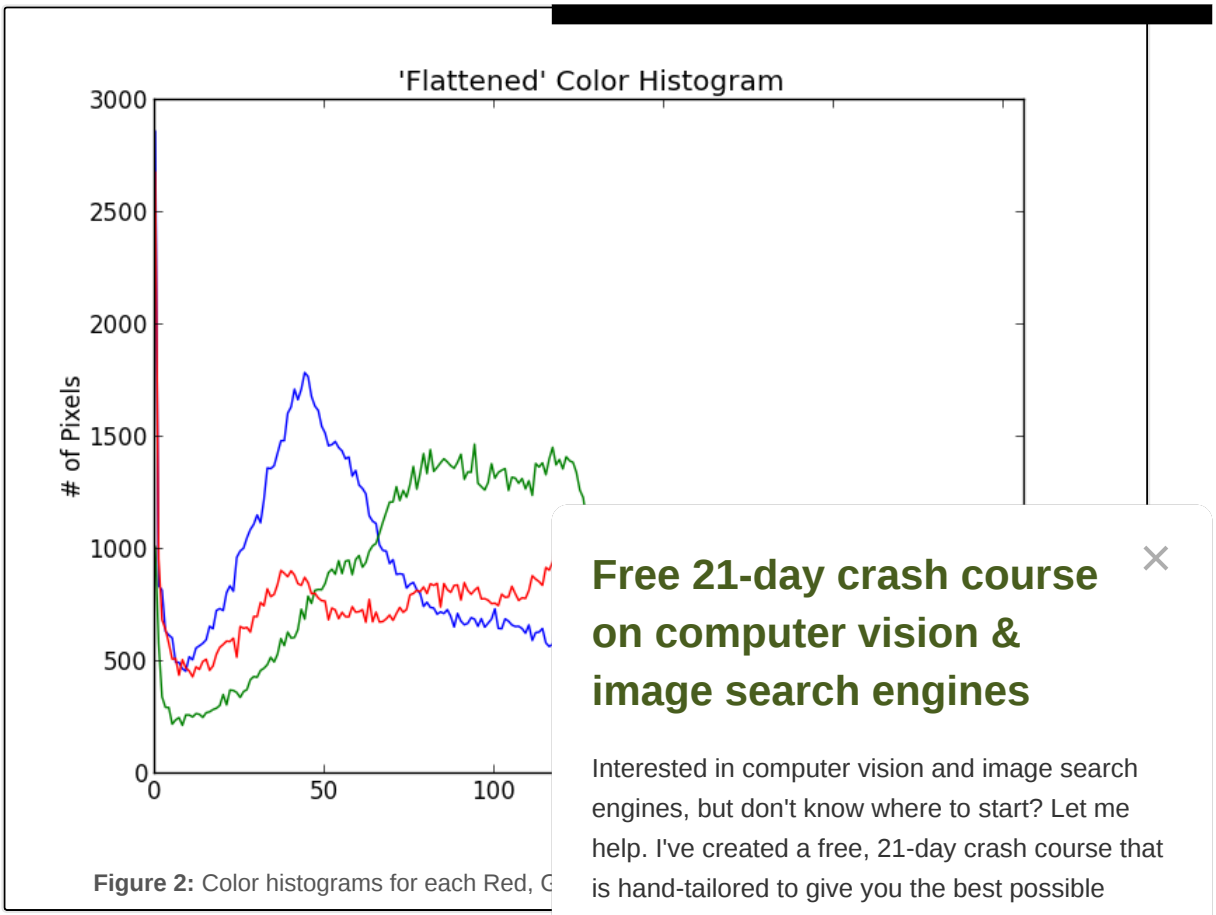


Figure 2: Color histograms for each Red, G

Awesome. That was pretty simple. What does thi rk
blue pixel values around bin #50. This range of b er
range of green pixels from bin #50 to #125 refers

Multi-dimensional Histog

Up until this point, we have computed a histogra
to multi-dimensional histograms and take into consideration two channels at a time.

The way I like to explain multi-dimensional histograms is to use the word **AND**. For example, we can ask a question such as
"how many pixels have a Red value of 10 **AND** a Blue value of 30?" How many pixels have a Green value of 200 **AND** a Red value of 130? By using the conjunctive **AND** we are able to construct multi-dimensional histograms.

It's that simple. Let's checkout some code to automate the process of building a 2D histogram:

Computing a 2D Color Histogram using Python and OpenCV — Python

```
# let's move on to 2D histograms -- I am reducing the
# number of bins in the histogram from 256 to 32 so we
# can better visualize the results
fig = plt.figure()

# plot a 2D color histogram for green and blue
ax = fig.add_subplot(131)
hist = cv2.calcHist([chans[1], chans[0]], [0, 1], None,
    [32, 32], [0, 256, 0, 256])
p = ax.imshow(hist, interpolation = "near
ax.set_title("2D Color Histogram for Gree
plt.colorbar(p)
```

```
13
14 # plot a 2D color histogram for green and red
15 ax = fig.add_subplot(132)
16 hist = cv2.calcHist([chans[1], chans[2]], [0, 1], None,
17     [32, 32], [0, 256, 0, 256])
18 p = ax.imshow(hist, interpolation = "nearest")
19 ax.set_title("2D Color Histogram for Green and Red")
20 plt.colorbar(p)
21
22 # plot a 2D color histogram for blue and red
23 ax = fig.add_subplot(133)
24 hist = cv2.calcHist([chans[0], chans[2]], [0, 1], None,
25     [32, 32], [0, 256, 0, 256])
26 p = ax.imshow(hist, interpolation = "nearest")
27 ax.set_title("2D Color Histogram for Blue and Red")
28 plt.colorbar(p)
29
30 # finally, let's examine the dimensionali
31 # the 2D histograms
32 print "2D histogram shape: %s, with %d va
33     hist.shape, hist.flatten().shape[0])
```

Yes, this is a fair amount of code. But that's only l or each combination of RGB channels: Red and Gre

Now that we are working with multi-dimensional h f bins we are using. In previous examples, I've use if we used a 256 bins for each dimension in a 2D h have 65,536 separate pixel counts. Not only is th applications using somewhere between 8 and 64 histograms. As **Lines 8-9** show, I am now using

The most important take away from this code car `cv2.calcHist` function. Here we see that we a Blue channels. And that's all there is to it.

So how is a 2D histogram stored in OpenCV? It's a 2D NumPy array. Since I used 32 bins for each channel, I now have a 32×32 histogram. We can treat this histogram as a feature vector simply by flattening it (**Line 33**). Flattening our histograms yields a list with 1024 values.

How do we visualize a 2D histogram? Let's take a look.

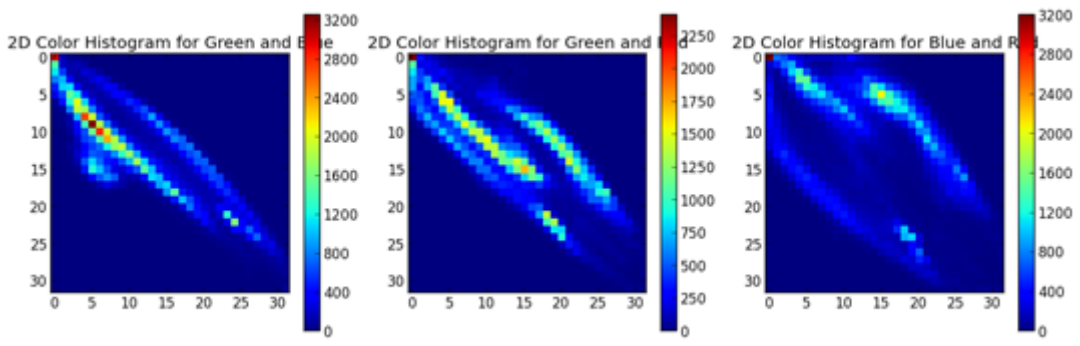


Figure 3: 2D color histograms for Dr. Grant.

In the above Figure, we see three graphs. The fir channels, the second for Green and Red, and the

low pixel counts, whereas shades of red represent large pixel counts (i.e. peaks in the 2D histogram). We can see such a peak in the Green and Blue 2D histogram (the first graph) when X=5 and Y=10.

Using a 2D histogram takes into account two channels at a time. But what if we wanted to account for *all three* RGB channels? You guessed it. We're now going to build a 3D histogram.

Computing a 3D Color Histogram using Python and OpenCV — Python

```
# our 2D histogram could only take into account 2 out
# of the 3 channels in the image so now let's build a
# 3D color histogram (utilizing all channels) with 8 bins
# in each direction -- we can't plot the 3D histogram, but
# the theory is exactly like that of a 2D histogram, so
# we'll just show the shape of the histogram
hist = cv2.calcHist([image], [0, 1, 2],
    None, [8, 8, 8], [0, 256, 0, 256, 0,
print "3D histogram shape: %s, with %d va
    hist.shape, hist.flatten().shape[0])
```

The code here is very simple — it's just an extens ee an 8x8x8 histogram for each of the RGB channel e that the shape is indeed `(8, 8, 8)` with 512 va vector can be done by simply flattening the array.

Color Spaces

The examples in this post have only explored the constructed for any color space in OpenCV. Discu post, but if you are interested, check out the docu

Drawbacks

Earlier in this post we made the assumption that semantically similar. For small, simple datasets, this may in fact be true. However, in practice, this assumption does not always hold.

Let's think about why this for.

For one, color histograms, by definition ignore both the shape and texture of the object(s) in the image. This means that color histograms have no concept of the shape of an object or the texture of the object. Furthermore, histograms also disregard any spatial information (i.e. where in the image the pixel value came from). An extension to the histogram, the color correlogram, can be used to encode a spatial relationship amongst pixels.

Let's think about Chic Engine, my visual fashion search engine iPhone app. I have different categories for different types of clothes, such as shoes and shirts. If I were using color histograms to describe a red shoe and a red shirt, the histogram would assume they were the same object. Clearly they are both red, but the semantics end there — they are simply not the same. Color histograms simply have no way to "model" what a shoe or a shirt is.

Finally, color histograms are sensitive to "noise", image was captured under and quantization erro se

limitations can potentially be mitigated by using a different color space than RGB (such as HSV or L*a*b*).

However, all that said, histograms are still widely used as image descriptors. They are dead simple to implement and very fast to compute. And while they have their limitations, they are very powerful when used correctly and in the right context.

What's next?

In this post I provided an OpenCV histogram example using Python, from start to finish.

Coming up Monday, we are going to build our first image search engine from start to finish using color histograms. Are you excited? I am. And maybe I'll even have a surprise for you that I've been building up around here. No guarantees though.

Until Monday!

Downloads:

If you would like to download the source code to this post, enter your email address in the form below. I'll also send you a **FREE 11-page** Image Search Engines, including **exclusive techniques** that I don't cover on the blog. Sound good? If so, enter your email address and I'll send you the code immediately!

Email address:

Your email address

DOWNLOAD THE CODE!

2d histogram, **3d histogram**, **color**, **histogram**, **rgb**

‹ Basic Image Manipulations in Python and OpenCV : Resizing (scaling), Rotating, and Cropping

Hobbits and Histograms – A How-To Guide to Building Your First Image Search Engine in Python ›

47 Responses to *Clever Girl: A Guide to Utilizing Color Histograms for Computer Vision and Image Search Engines*

mamachanko February 1, 2014 at 3:59 am #

I think you meant to say that the commo
[0, 256] when describing the calcHist funtion.

mamachanko February 1, 2014 at 4:01

Indeed, it seems I was wrong.

Adrian Rosebrock February 1, 2

This used to throw me for a loop
OpenCV and NumPy "gotchas" I've enco

So yes, the possible pixel values for each
However, when using calcHist (and NumPy's histogram method) the range becomes [0, 255). (Notice the parenthesis, not a bracket). It's an extremely subtle difference, and in most cases, it wouldn't "break" anything. But in order to capture the range [0, 255] with 256 bins, we have to specify [0, 256). It's definitely annoying.

Jeremy November 22, 2014 at 3:10 pm #

REPLY

I had a question on line 9: ap.add_argument("-i", "–image", required = True, help = "Path to the image")

If my path is: C:\Users\Jeremy\Documents\IPython Notebooks,
would my line 9 look like ap.add_argument("Ipython Notebooks", "face.jpeg", required = True, help = "Path to the image")?

Adrian Rosebrock November 23, 201

Hi Jeremy. If you are using an IPython Notebook you can skip the argument parsing entirely and use the path to your image directly. The argument parsing is only necessary if you are executing the script via command line.

Phil Gl October 31, 2015 at 2:46 am #

REPLY

I'm attempting to plot a 3D histogram. I thought it would be simple extension of the 2D examples you provide using something like:

hist = cv2.calcHist([img_patch],[0,1,2],None,[8,8,8],[0,256,0,256,0,256])

Where the Z axis would provide 'red' information a
respectively.

Not having much luck with it. Any guidance would

Shelly November 14, 2015 at 3:48 pm #

Did you have any luck with this? See

Mohanish December 15, 2015 at 9:10 pm #

hey, iam getting this error

```
1 "usage: mycv_3.py [-h] -i IMAGE
2 mycv_3.py: error: argument -i/--image i
```

iam really stucked in it…
help me out !!!

Adrian Rosebrock December 16, 2015 at 6:37 am #

REPLY

Hey Mohanish, you need to supply the `--image` switch, which is the path to the image you want to compute a histogram for:

```
$ python mycv_3.py --image path/to/my/image.jpg
```

I would suggest downloading the .zip of the source code to this post which includes the example image and example command to run.

Hamid December 16, 2015 at 4:54 am #

REPLY

Dear Adrian,
Thank you for providing awsome material around

histogram of images in a database, so I tried to append each of the feature list in another list as follow:

```
1 for (chan, color) in zip(chans, colors):
2                 hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
3                 features.extend(hist)
4
5         Matrix.append(features)
```

but when I write the martix as a csv file, I get a [ ] around each number in the csv file, do you have any idea how to work out this problem, as I need to read color histograms as an excel file in matlab later.

I truly appreciate your kind guidance and help.

Thank you.

Adrian Rosebrock December 16, 201

Hey Hamid, have you tried looking a
post demonstrates how to extract color histog
to a CSV file.

Stephen S Eacuellon June 23, 2016 at 11

Hi Adrian,

When I run the code with the image you provided

OpenCV Error: Assertion failed (size.width>0 &&
D:\Build\OpenCV\opencv-3.1.0\modules\highgui\s

Seems like openCV thinks the image size is 0? Strangely it runs with .png files correctly. Any idea what might cause an error like this?

Adrian Rosebrock June 23, 2016 at 1:04 pm #

REPLY

It's a strange error, but it seems like that your system is not configured to load JPEG files. If it .png files are working correctly, then you have the proper PNG libraries installed. But I'm guessing that the JPEG libraries are not installed/are not working with OpenCV.

Farah July 13, 2016 at 6:14 am #

REPLY

Hi Adrian

I wanted an equal density histogram for my research purpose. Is there some way of specifying the bins with calcHist rather than the size of the bin. I
to 10 then the second may be from 11to 16 and s

Adrian Rosebrock July 14, 2016 at 1:13 pm #

REPLY

If you're looking for non-equal sized bins, I probably wouldn't use the `cv2.calcHist` function for this. Instead, the NumPy's histogram function will give you the fine-grained control that you're looking for.

Jooyeon July 21, 2016 at 12:18 pm #

REPLY

Hello, Adrian!
I have downloaded the code and ran this code bu
plt" I have followed the tutorial from your blog pos
(http://www.pyimagesearch.com/2015/06/15/insta
matplotlib on the cv virtual environment.

It has this runtime error: RuntimeError: Python is
backend will not be able to function correctly if Py
documentation for more information on installing
reinstall Python as a framework, or try one of the
a virtual enviroment see 'Working with Matplotlib

I wonder how you solved this problem. Thank you

Adrian Rosebrock July 21, 2016 at 12

Hey Jooyeeon — this error happens
incompatibly with the virtual environments. The developers are aware of the problem, but I don't know of an exact fix for the newer versions of matplotlib. Instead, I would recommend installing a previous version of matplotlib:

```
$ pip install matplotlib==1.4.3
```

This should take care of the issue.

Jooyeon July 22, 2016 at 2:37 am #

REPLY

Thanks, Adrian. It works beautifully! Can't wait to dive into more openCV projects. I hope I can be an expert like you someday, sharing my knowledge with others to build awesome things!!

Adrian Rosebrock July 22, 2016 at 10:55 am #

REPLY

Awesome, I'm glad to hear it

AMAN KHATRI August 5, 2016 at 5:56 am #

REPLY

I have installed 1.4.3 version but the above mentioned error is still there for me.what can i do to remove the error?

Adrian Rosebrock August 7, 2016 at 8:20 am #

REPLY

You might want to double-check which version of matplotlib you are using and whether or not you are using a Pytho
virtual environments are independent
switched matplotlib versions for the ir

Devi September 19, 2016 at 3:59 am #

Thank you for sharing. Nicely explained

Adrian Rosebrock September 19, 201

Fantastic, I'm happy to hear the tuto

chaidev October 18, 2016 at 11:46 am #

Adrian, I just wanted to tell you how thoroughly impressed I am with this blog and with your commitment to keep answering comments on (older) posts. I'll be following along as i'm in looking to apply some CV in upcoming projects.
The threshold for following your tuts is low, yet the extra info along the way really helps build context for what we're actually doing.

thank you so much for your effort!

Adrian Rosebrock October 20, 2016 at 8:51 am #

REPLY

Thank you for the kind words Chaidev, it's comments like these that keep me coming back to answer comments on older posts 🙂

Dipti November 26, 2016 at 3:16 pm #

How to draw a histogram of an RGB ima

Adrian Rosebrock November 28, 2016 at 10:31 am #

REPLY

I would use `matplotlib` to draw the histograms. I cover computing and extracting histograms in detail inside my book, Practical Python and OpenCV.

Vishnu January 19, 2017 at 8:04 am #

REPLY

Hi Adrian!
So Im realtively new to python and Im not able to
what is the purpose of lines 8-10? What does it d

Adrian Rosebrock January 20, 2017 a

Hey Vishnu — I would suggest you s
arguments.

S Rank February 25, 2017 at 6:18 pm #

If anyone is getting the following error wh
10.11.5):

Terminating app due to uncaught exception 'NSIn
_setup:]: unrecognized selector sent to instance 0

This is what I did to resolve it:

pip install PyQT5

(at the very top of the code file)
import matplotlib
matplotlib.use("Qt5Agg")

I understand the problem is matplotlib relies on the OS for certain image processing/rendering functionality, and on OSX it tries to interface with the OS in an unsupported way. By specifying exactly how to interface with the OS, this problem is resolved.

Ctrl + F tags: MacOS, Apple

Adrian Rosebrock February 27, 2017 at 11:14 am #

REPLY

Thanks for sharing!

Jiayuanyuan March 1, 2017 at 9:23 am #

REPLY

hello,i create a code page to run this code,and i copy your code.But, when it is run,it will flash and exit,can't see any result.can you tell me why?

Adrian Rosebrock March 2, 2017 at 6:50 am #

REPLY

I would suggest you use the "Downloads" section of this tutorial to download the code for the post. Run my code, see if you get the correct result, and if so, compare the code to your own.

Xiaohua Peng April 14, 2017 at 4:39 am #

Hello, I also have some problems woth t
ap.add_argument("-i", "–image file/grant.jpg", requ
the "file/grant,jpg" is the exact place I put my jpg i
work, hope you can help me out, thanks.

Adrian Rosebrock April 16, 2017 at 8:

You should read up on command lin
to update the argument parsing code. The pro
line arguments via the terminal.

Qian April 26, 2017 at 9:31 am #

REPLY

Hi Adrian
I use the codes that was download by pyimagesearch,usage "python histograms.py –image grant.jpg" ,the result is just show two pictures (the raw and grayscale),I CAN'T SEE THE HISTOGRAM ,help me out .(ps. I have installed matplotlib.)

Adrian Rosebrock April 28, 2017 at 9:51 am #

REPLY

Hi Qian — I would suggest referring to this blog post on why plots may not show up.

Richie Close May 19, 2017 at 10:05 am #

REPLY

Thanks for the great tutorial! I've a related question. I'm an album of photos and I want to find the brightest image in the set, or a means to gene

which would represent an images histogram for example. I had tried cv2.calcHist but I'm struggling to pull it together into code. Any help appreciated!

Adrian Rosebrock May 21, 2017 at 5:18 am #

REPLY

Hi Richie — you're in luck, I'll be publishing a blog post on how to determine the "most colorful" and "least colorful" images in a dataset later this month. Be sure to keep your eye on the PyImageSearch blog!

Trackbacks/Pingbacks

Hobbits and Histograms - A How-To Guide to Buildin
PyImageSearch - January 27, 2014
[…] discussed the color histogram in my previous po
vision and image search engines. If you haven't read
and read it after […]
URL - January 29, 2014
… [Trackback]
[…] Read More here: pyimagesearch.com/2014/01/2
computer-vision-and-image-search-engines/ […]
Building an Image Search Engine: Indexing Your Dat
2014
[…] Color: Image descriptors that characterize the co
pixel intensities in each channel of the image. These
standard deviation, and skewness, along with color h
Building an Image Search Engine: Defining Your Sim y
17, 2014
[…] what it takes to build an image search engine. Start extracting features from images using simple color histograms. Then compare them using the distance functions discussed above. Note your […]
How To Describe and Quantify an Image Using Feature Vectors - March 3, 2014
[…] back to the Clever Girl: A Guide to Utilizing Color Histograms for Computer Vision and Image Search Engines and Hobbits and Histograms, we could also use a 3D color histogram to describe our […]
Building an Image Search Engine: Defining Your Image Descriptor (Step 1 of 4) - PyImageSearch - April 28, 2014
[…] the mean, standard deviation, and skew of each channel's pixel intensities. We could also use color histograms as we've seen in other blog posts. In color histograms are global image descriptors applied […]
Color Quantization with OpenCV using K-Means Clustering - PyImageSearch - July 7, 2014
[…] given 24-bit RGB image has 256 x 256 x 256 possible colors. And sure, we can build standard color histograms based on these intensity […]
How-To: 3 Ways to Compare Histograms using OpenCV and Python - PyImageSearch - July 14, 2014
[…] For more details on the cv2.calcHist function, definitely take a look at my guide to utilizing color histograms for computer vision and image search engines post. […]

Leave a Reply

Free 21-day crash course on computer vision & image search engines

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Resource Guide (it's totally free).

Click the button below to get my **free**
Uncover **exclusive techniques** that
engines of your own.

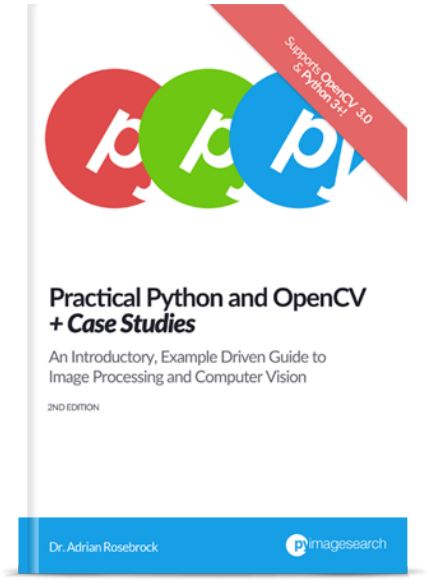
Downloa

Deep Learning for Computer Vision with Python Book

You're interested in deep learning and computer vision, *b*
book will teach you all you need to know about deep

I'm an entrepreneur and Ph.D who has launched two successful image search engines, ID My Pill and Chic Engine. I'm here to share my tips, tricks, and hacks I've learned along the way.

Learn computer vision in a single weekend.



Want to learn computer vision & OpenCV? I can teach yo o joke. My new book is your **guaranteed, quick-start guic** ? Click here to become a computer vision ninja.

CLICK HERE TO BECOME AN OPENCV NINJA

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Subscribe via RSS

Never miss a post! Subscribe to the PyImageSearch RSS Feed and keep up to date with my image search engine tutorials, tips, and tricks

POPULAR

Install OpenCV and Python on your Raspberry Pi 2 and B+
FEBRUARY 23, 2015

Home surveillance and motion detection with the Raspberry Pi, Python, OpenCV, and Dropbox
JUNE 1, 2015

Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3
APRIL 18, 2016

How to install OpenCV 3 on Raspbian Jessie
OCTOBER 26, 2015

Basic motion detection and tracking with Python and O
MAY 25, 2015

Free 21-day crash course on computer vision & image search engines

Accessing the Raspberry Pi Camera with OpenCV and Python
MARCH 30, 2015

Install OpenCV 3.0 and Python 2.7+ on Ubuntu
JUNE 22, 2015

Search

Search...

Free 21-day crash course on computer vision & image search engines